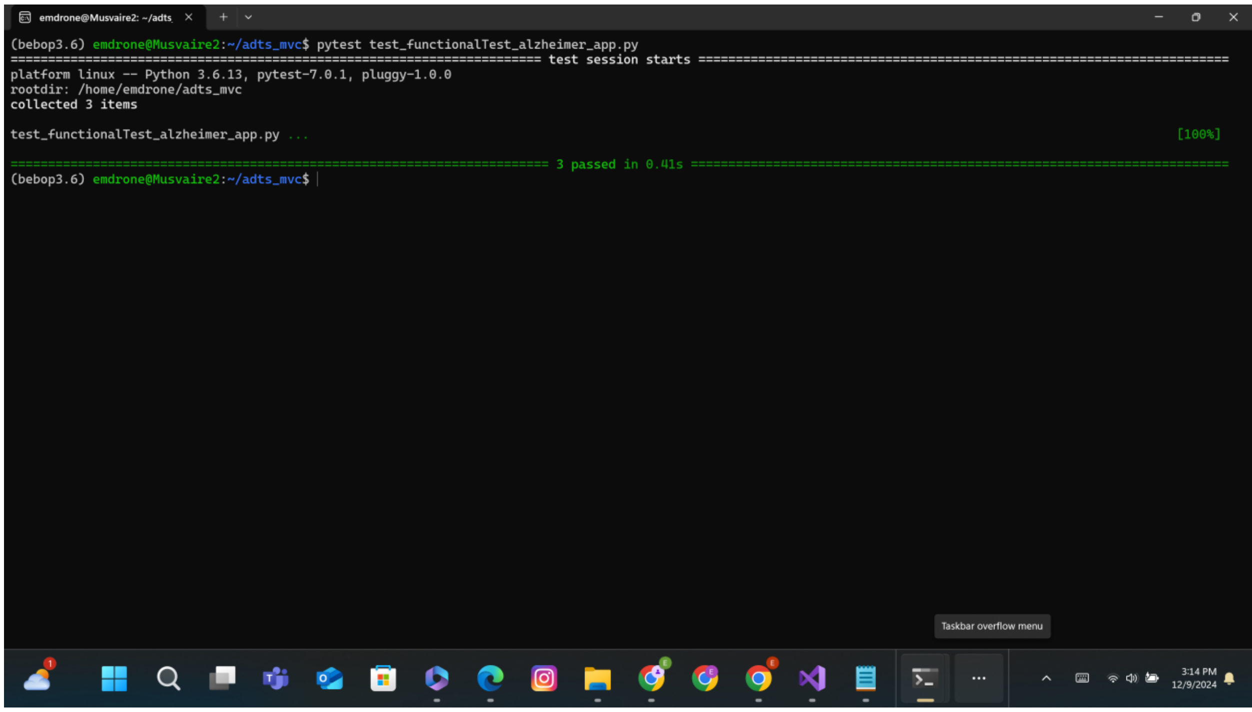
Implementation and Testing for Alzheimer's Disease Testing Software

Functional Testing

Purpose: To verify that the software functions according to the specified requirements.

- **Focus**: Ensures that each function of the software application operates in conformance with the requirement specification.
- **Scope**: Includes testing of APIs, databases, security, client/server applications, and other functionalities.
- **Examples**:
    - Verifying that a user can log in with valid credentials.
    - Checking that data is correctly saved to the database.
    - Ensuring that a user can successfully complete a transaction.



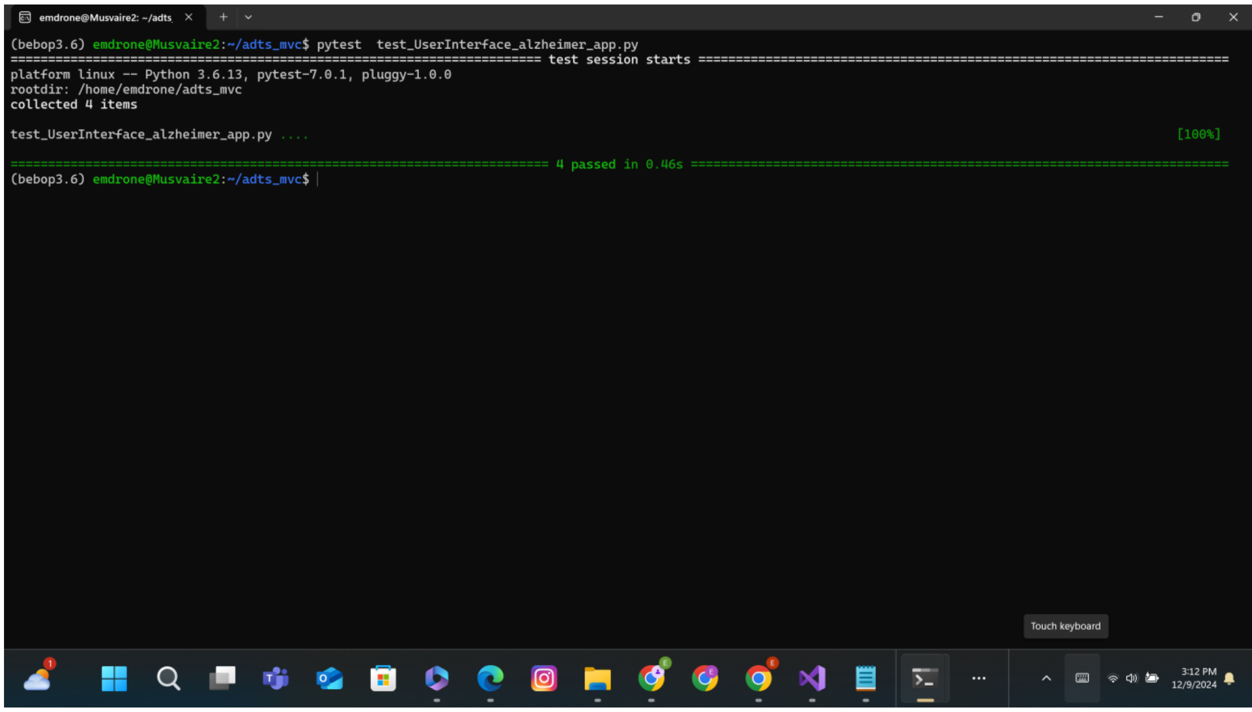
```
(bebop3.6) emdrone@Musvaire2:~/adts_mvc$ pytest test_functionalTest_alzheimer_app.py
===================================================================== test session starts ======================================================================
platform linux -- Python 3.6.13, pytest-7.0.1, pluggy-1.0.0
rootdir: /home/emdrone/adts_mvc
collected 3 items

test_functionalTest_alzheimer_app.py ...                                                                                                                  [100%]

====================================================================== 3 passed in 0.41s =======================================================================
(bebop3.6) emdrone@Musvaire2:~/adts_mvc$
```

User Interface (UI) Testing

Purpose: To ensure that the user interface of the application works as expected and provides a good user experience.

- **Focus**: Ensures that the graphical user interface meets the design specifications and is user-friendly.

- **Scope**: Includes testing of visual elements like buttons, menus, icons, and other graphical components.
- **Examples**:
  - Verifying that buttons are clickable and perform the correct actions.
  - Checking that text fields accept input and display the correct data.
  - Ensuring that the layout is consistent across different devices and screen sizes.

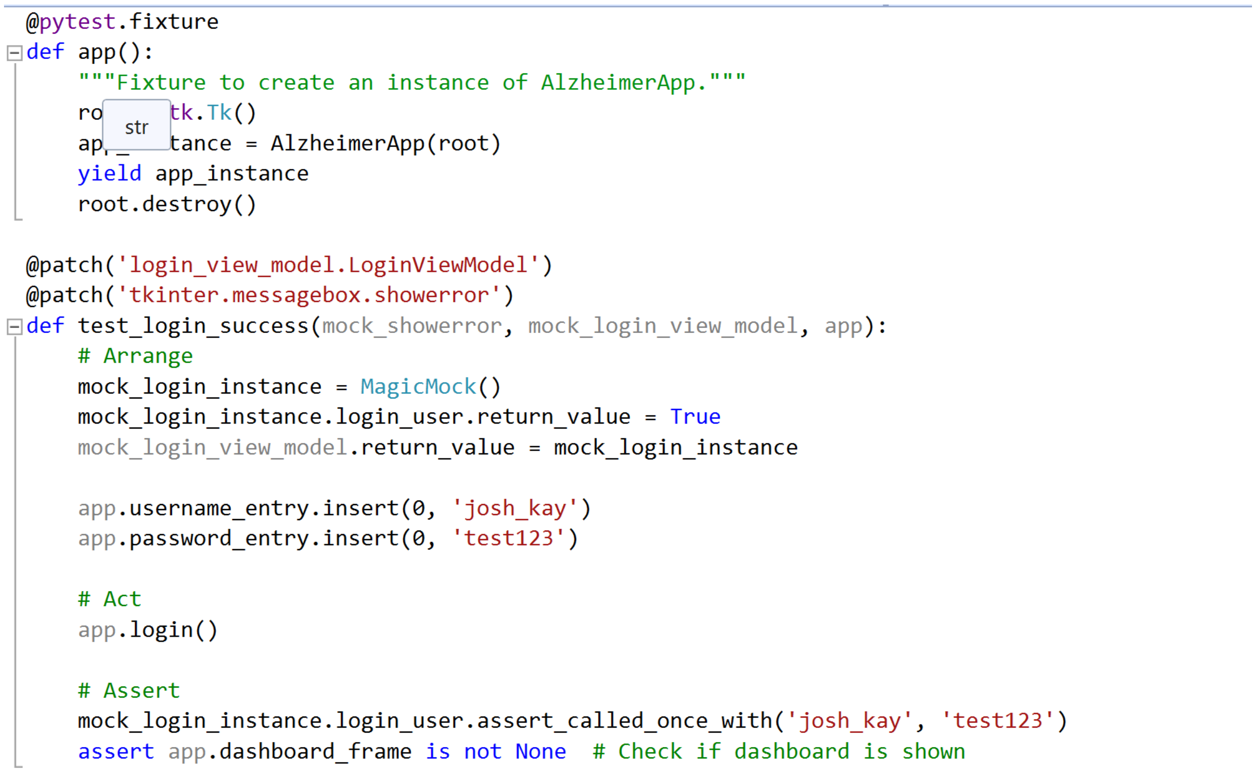


```
(bebop3.6) emdrone@Musvaire2:~/adts_mvc$ pytest  test_UserInterface_alzheimer_app.py
======================================================================== test session starts ========================================================================
platform linux -- Python 3.6.13, pytest-7.0.1, pluggy-1.0.0
rootdir: /home/emdrone/adts_mvc
collected 4 items

test_UserInterface_alzheimer_app.py ....                                                                                                                     [100%]

========================================================================= 4 passed in 0.46s =========================================================================
(bebop3.6) emdrone@Musvaire2:~/adts_mvc$
```

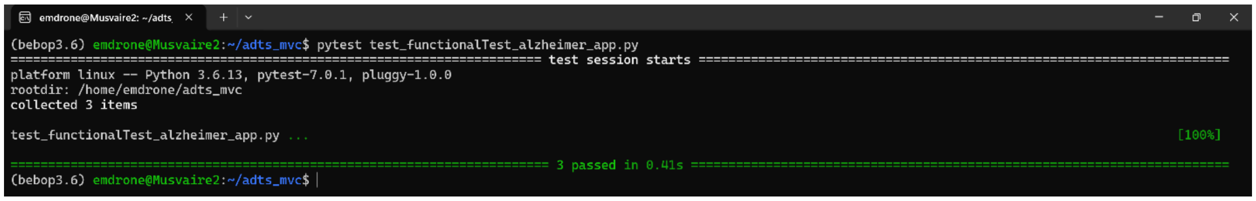
Key Differences

- **Objective**:
  - **Functional Testing**: Focuses on the functionality of the application.
  - **UI Testing**: Focuses on the look and feel of the application.
- **Level of Testing**:
  - **Functional Testing**: Can be performed at various levels, including unit, integration, system, and acceptance testing.
  - **UI Testing**: Primarily performed at the system and acceptance testing levels.
- **Tools**:
  - **Functional Testing**: Tools like Selenium, Postman, JUnit, and TestNG.
  - **UI Testing**: Tools like Selenium, QTP, and TestComplete.

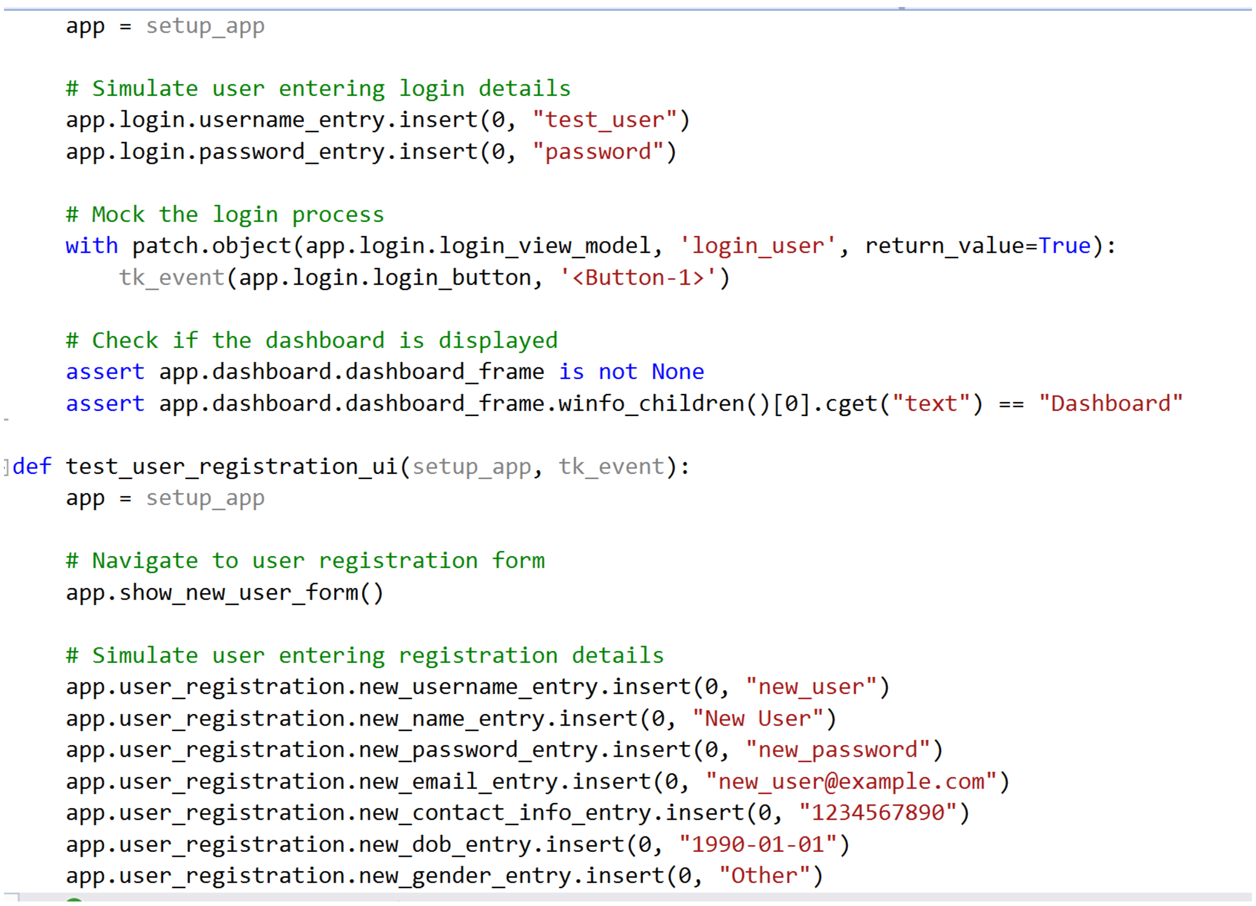
Both types of testing are crucial for delivering a high-quality software product. Functional testing ensures that the application works correctly, while UI testing ensures that it is user-friendly and visually appealing.



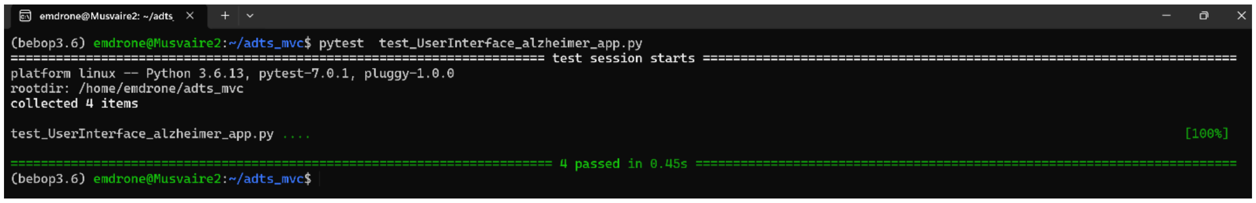
```
@pytest.fixture
def app():
    """Fixture to create an instance of AlzheimerApp."""
    ro        tk.Tk()
    ap _      tance = AlzheimerApp(root)
    yield app_instance
    root.destroy()

@patch('login_view_model.LoginViewModel')
@patch('tkinter.messagebox.showerror')
def test_login_success(mock_showerror, mock_login_view_model, app):
    # Arrange
    mock_login_instance = MagicMock()
    mock_login_instance.login_user.return_value = True
    mock_login_view_model.return_value = mock_login_instance

    app.username_entry.insert(0, 'josh_kay')
    app.password_entry.insert(0, 'test123')

    # Act
    app.login()

    # Assert
    mock_login_instance.login_user.assert_called_once_with('josh_kay', 'test123')
    assert app.dashboard_frame is not None  # Check if dashboard is shown
```



```
(bebop3.6) emdrone@Musvaire2:~/adts_mvc$ pytest test_functionalTest_alzheimer_app.py
======================================================================= test session starts ========================================================================
platform linux -- Python 3.6.13, pytest-7.0.1, pluggy-1.0.0
rootdir: /home/emdrone/adts_mvc
collected 3 items

test_functionalTest_alzheimer_app.py ...                                                                                                                     [100%]

======================================================================== 3 passed in 0.41s =========================================================================
(bebop3.6) emdrone@Musvaire2:~/adts_mvc$
```



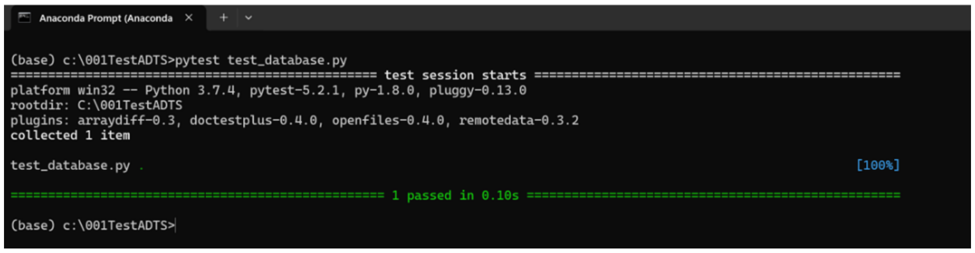
```python
    app = setup_app

    # Simulate user entering login details
    app.login.username_entry.insert(0, "test_user")
    app.login.password_entry.insert(0, "password")

    # Mock the login process
    with patch.object(app.login.login_view_model, 'login_user', return_value=True):
        tk_event(app.login.login_button, '<Button-1>')

    # Check if the dashboard is displayed
    assert app.dashboard.dashboard_frame is not None
    assert app.dashboard.dashboard_frame.winfo_children()[0].cget("text") == "Dashboard"

def test_user_registration_ui(setup_app, tk_event):
    app = setup_app

    # Navigate to user registration form
    app.show_new_user_form()

    # Simulate user entering registration details
    app.user_registration.new_username_entry.insert(0, "new_user")
    app.user_registration.new_name_entry.insert(0, "New User")
    app.user_registration.new_password_entry.insert(0, "new_password")
    app.user_registration.new_email_entry.insert(0, "new_user@example.com")
    app.user_registration.new_contact_info_entry.insert(0, "1234567890")
    app.user_registration.new_dob_entry.insert(0, "1990-01-01")
    app.user_registration.new_gender_entry.insert(0, "Other")
```



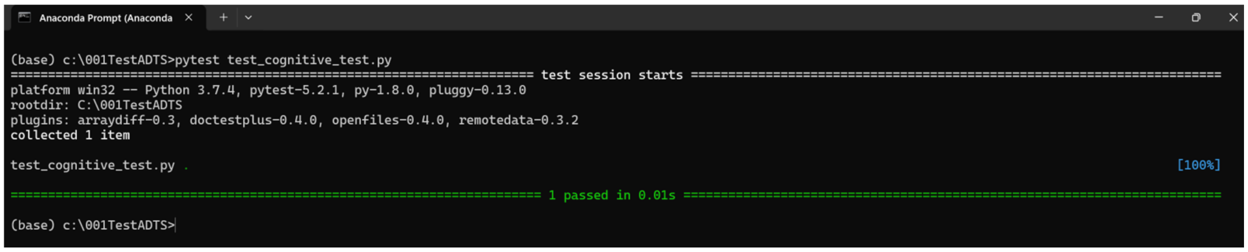
```
(bebop3.6) emdrone@Musvaire2:~/adts_mvc$ pytest  test_UserInterface_alzheimer_app.py
======================================================================== test session starts ========================================================================
platform linux -- Python 3.6.13, pytest-7.0.1, pluggy-1.0.0
rootdir: /home/emdrone/adts_mvc
collected 4 items

test_UserInterface_alzheimer_app.py ....                                                                                                                     [100%]

========================================================================= 4 passed in 0.45s =========================================================================
(bebop3.6) emdrone@Musvaire2:~/adts_mvc$
```

Test Types

1. **Unit Testing**: Validate individual components or functions.



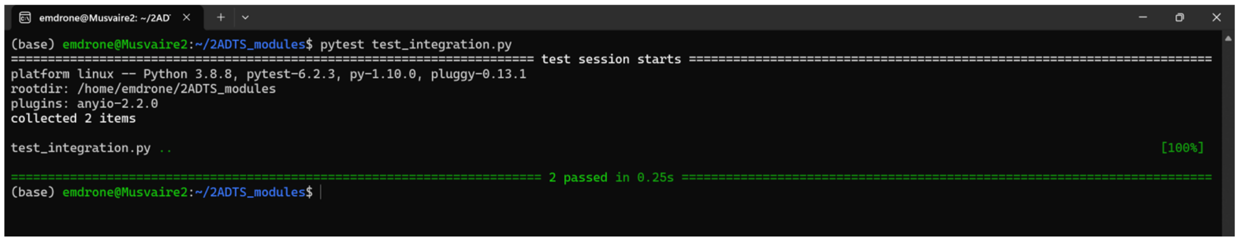
```
(base) c:\001TestADTS>pytest test_database.py
================================================= test session starts =================================================
platform win32 -- Python 3.7.4, pytest-5.2.1, py-1.8.0, pluggy-0.13.0
rootdir: C:\001TestADTS
plugins: arraydiff-0.3, doctestplus-0.4.0, openfiles-0.4.0, remotedata-0.3.2
collected 1 item

test_database.py .                                                                                               [100%]

================================================== 1 passed in 0.10s ==================================================

(base) c:\001TestADTS>
```



```
(base) c:\001TestADTS>pytest test_cognitive_test.py
======================================================================= test session starts ========================================================================
platform win32 -- Python 3.7.4, pytest-5.2.1, py-1.8.0, pluggy-0.13.0
rootdir: C:\001TestADTS
plugins: arraydiff-0.3, doctestplus-0.4.0, openfiles-0.4.0, remotedata-0.3.2
collected 1 item

test_cognitive_test.py .                                                                                                                                     [100%]

======================================================================== 1 passed in 0.01s =========================================================================

(base) c:\001TestADTS>
```

2. **Integration Testing**: Ensure that different modules or services interact correctly.

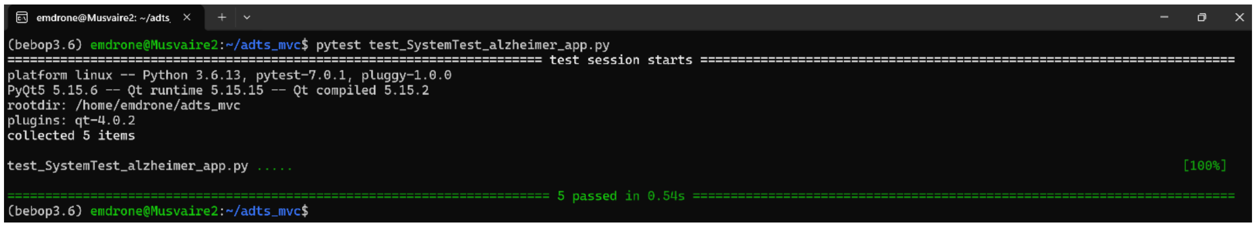
Integrating Testing



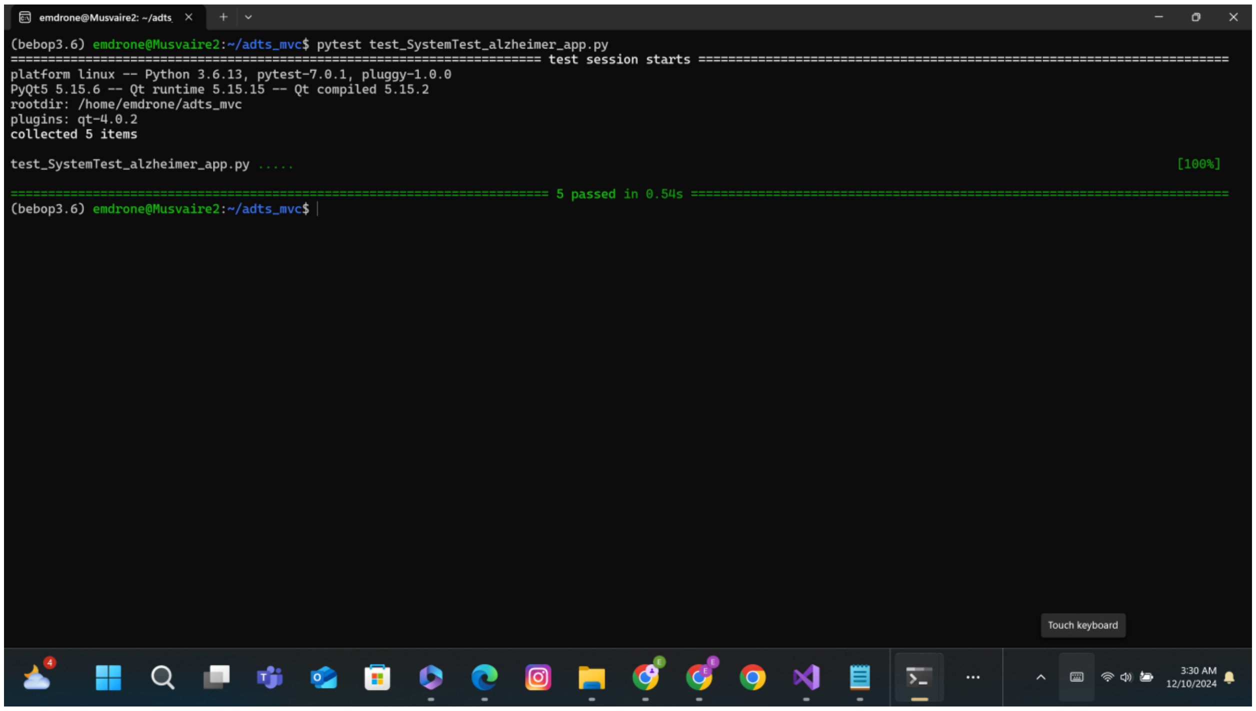
```
(base) emdrone@Musvaire2:~/2ADTS_modules$ pytest test_integration.py
======================================================================= test session starts ========================================================================
platform linux -- Python 3.8.8, pytest-6.2.3, py-1.10.0, pluggy-0.13.1
rootdir: /home/emdrone/2ADTS_modules
plugins: anyio-2.2.0
collected 2 items

test_integration.py ..                                                                                                                                       [100%]

======================================================================== 2 passed in 0.25s =========================================================================
(base) emdrone@Musvaire2:~/2ADTS_modules$
```

1. **System Testing**: Validate the complete and integrated software product.

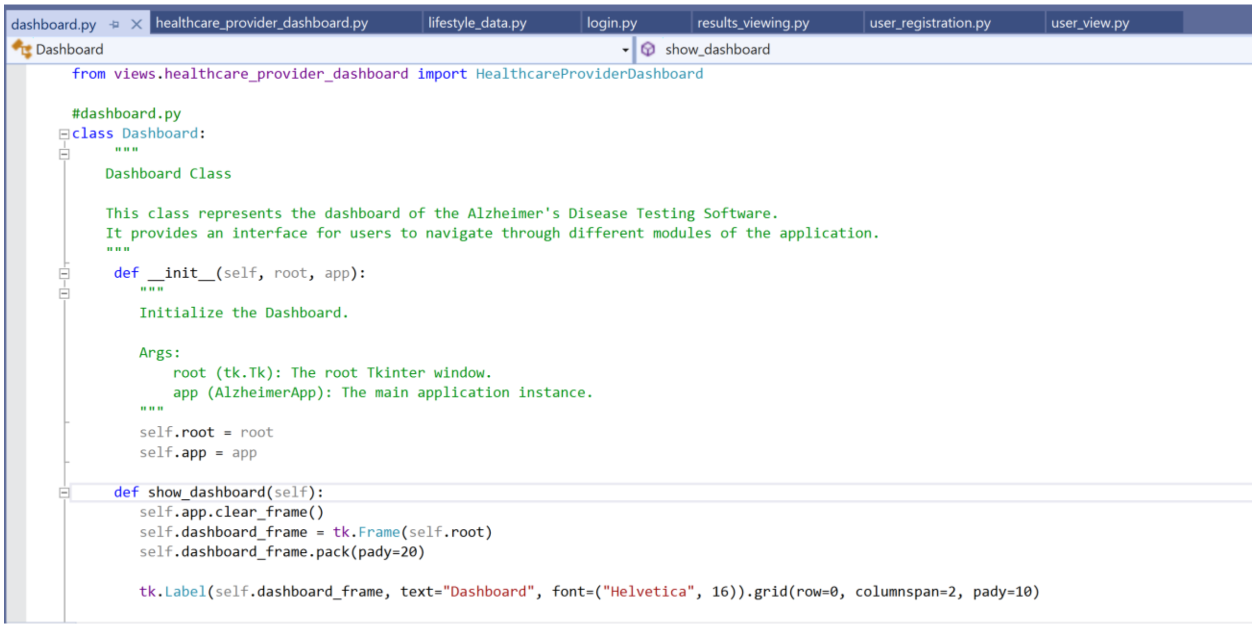
System testing



```
(bebop3.6) emdrone@Musvaire2:~/adts_mvc$ pytest test_SystemTest_alzheimer_app.py
======================================================================= test session starts ========================================================================
platform linux -- Python 3.6.13, pytest-7.0.1, pluggy-1.0.0
PyQt5 5.15.6 -- Qt runtime 5.15.15 -- Qt compiled 5.15.2
rootdir: /home/emdrone/adts_mvc
plugins: qt-4.0.2
collected 5 items

test_SystemTest_alzheimer_app.py .....                                                                                                                       [100%]

======================================================================== 5 passed in 0.54s =========================================================================
(bebop3.6) emdrone@Musvaire2:~/adts_mvc$
```



```
(bebop3.6) emdrone@Musvaire2:~/adts_mvc$ pytest test_SystemTest_alzheimer_app.py
=========================================================================== test session starts ===========================================================================
platform linux -- Python 3.6.13, pytest-7.0.1, pluggy-1.0.0
PyQt5 5.15.6 -- Qt runtime 5.15.15 -- Qt compiled 5.15.2
rootdir: /home/emdrone/adts_mvc
plugins: qt-4.0.2
collected 5 items

test_SystemTest_alzheimer_app.py .....                                                                                                                            [100%]

============================================================================ 5 passed in 0.54s ============================================================================
(bebop3.6) emdrone@Musvaire2:~/adts_mvc$
```

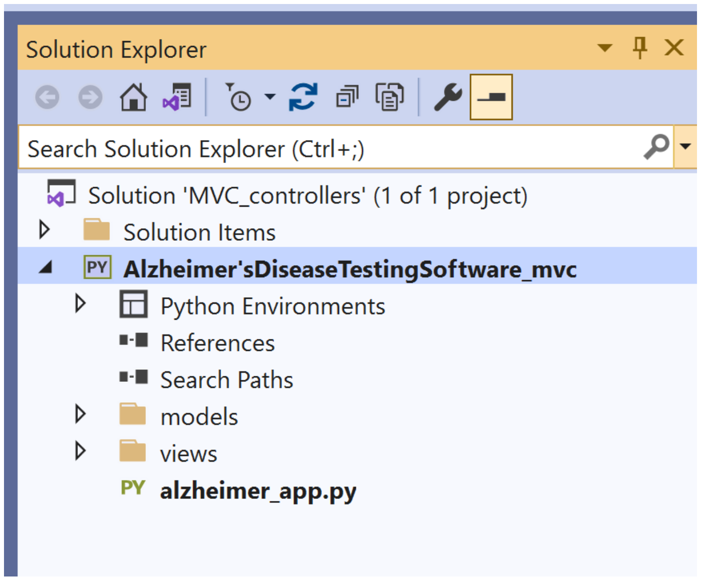


```
from views.healthcare_provider_dashboard import HealthcareProviderDashboard

#dashboard.py
class Dashboard:
    """
    Dashboard Class

    This class represents the dashboard of the Alzheimer's Disease Testing Software.
    It provides an interface for users to navigate through different modules of the application.
    """
     def __init__(self, root, app):
        """
        Initialize the Dashboard.

        Args:
            root (tk.Tk): The root Tkinter window.
            app (AlzheimerApp): The main application instance.
        """
        self.root = root
        self.app = app

     def show_dashboard(self):
        self.app.clear_frame()
        self.dashboard_frame = tk.Frame(self.root)
        self.dashboard_frame.pack(pady=20)

        tk.Label(self.dashboard_frame, text="Dashboard", font=("Helvetica", 16)).grid(row=0, columnspan=2, pady=10)
```

Modularity and Reusability

- **Modular Code**: The code is divided into classes and functions that handle specific tasks. This promotes modularity and makes the code easier to maintain and extend.

- **Reusability**: Components such as database connection methods and data processing functions are designed to be reusable.



```
Solution 'MVC_controllers' (1 of 1 project)
  Solution Items
  Alzheimer'sDiseaseTestingSoftware_mvc
    Python Environments
    References
    Search Paths
    models
    views
    alzheimer_app.py
```

Implementing a Model-View-Controller (MVC) architecture in your Alzheimer's Disease Testing Software helps separate concerns, making the code more modular, maintainable, and scalable. Here's how it is implemented.

1. **Model**

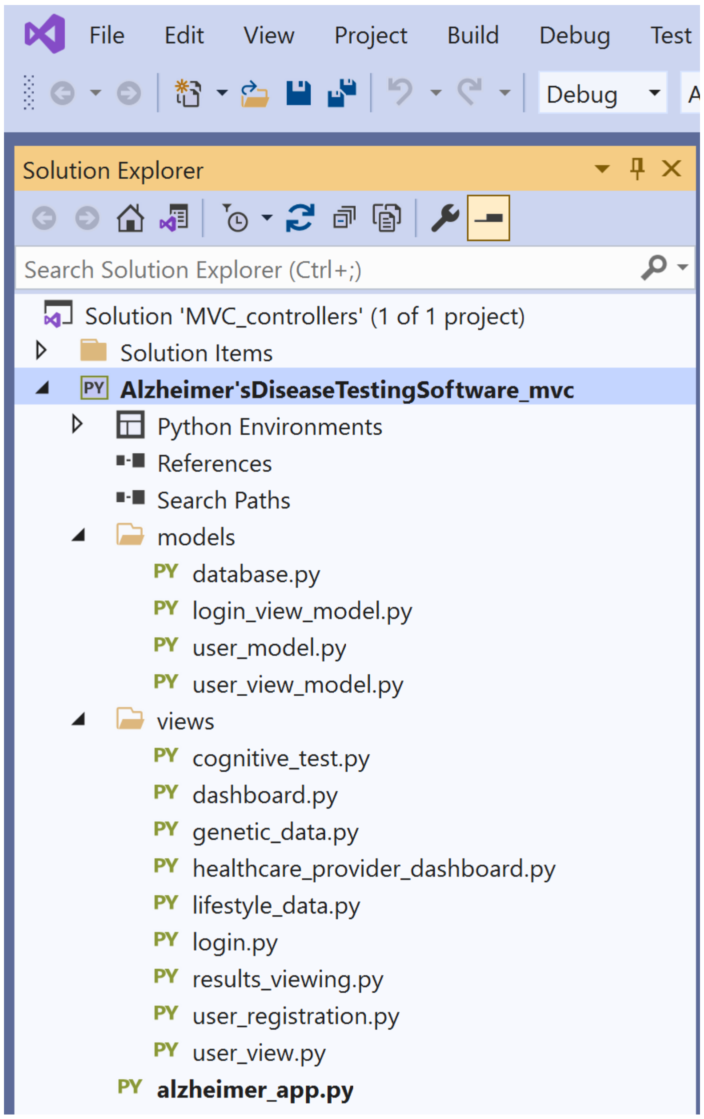
The Model represents the data and the business logic of the application. It interacts with the database and performs operations on the data.

2. View

The View is responsible for displaying the data to the user. It represents the UI components.

3. Controller

The Controller handles user input and updates the Model and View accordingly. It acts as an intermediary between the Model and the View.



```
Solution 'MVC_controllers' (1 of 1 project)
  Solution Items
  Alzheimer'sDiseaseTestingSoftware_mvc
    Python Environments
    References
    Search Paths
    models
      database.py
      login_view_model.py
      user_model.py
      user_view_model.py
    views
      cognitive_test.py
      dashboard.py
      genetic_data.py
      healthcare_provider_dashboard.py
      lifestyle_data.py
      login.py
      results_viewing.py
      user_registration.py
      user_view.py
    alzheimer_app.py
```

4. Modularity and Reusability

The Alzheimer's Disease Testing Software is designed with modularity and reusability in mind. Below are the eight modules, each with inline comments, docstrings.

1. Database Module

python

```python
import mysql.connector
from mysql.connector import Error

class Database:
    """Handles database connections and operations."""

    @staticmethod
    def connect_db():
        """
        Establishes a connection to the MySQL database.
        Returns:
            connection: A MySQL connection object or None if connection fails.
        """

        try:
            connection = mysql.connector.connect(
                host='localhost',
                database='alzheimer_testing',
                user='root',
                password='your_password'
            )
            if connection.is_connected():
                return connection
        except Error as e:
            print("Error while connecting to MySQL", e)

        return None
```

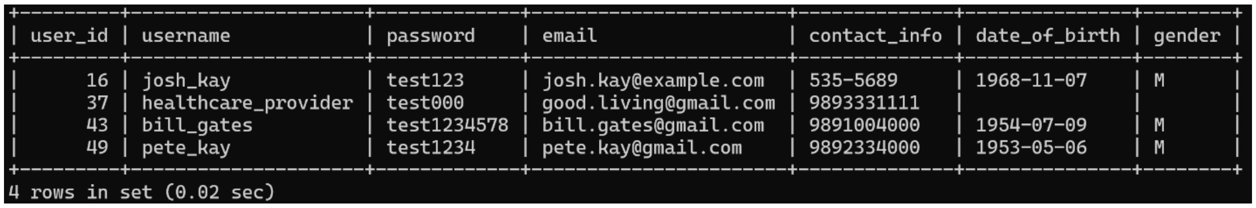
2. User Model Module

python

```
from database import Database  # Ensure this line is present


# user_model.py
class UserModel:
    """Represents a user in the system."""
    def __init__(self, user_id=None, username=None, name=None, password=None, email=None,
contact_info=None, date_of_birth=None, gender=None):
        """
        Initializes a user_model instance.

        Args:
            user_id: Unique identifier for the user.
            username: Username of the user.
            name: Full name of the user.
            password: User's password (should be hashed).
            email: User's email address.
            contact_info: User's contact information.
        """
        self.user_id = user_id
        self.username = username  # Added username field
        self.name = name
        self.password = password  # Ensure this is stored securely
        self.email = email
        self.contact_info = contact_info
        self.date_of_birth = date_of_birth
        self.gender = gender
    def register(self):
        """Registers a new user in the database."""
        connection = Database.connect_db()
        if connection:
            cursor = connection.cursor()
            cursor.execute("INSERT INTO USER (username, password, email, contact_info) 
VALUES (%s, %s, %s, %s)", 
                           (self.name, 'default_password', f'{self.name}@example.com', 
self.contact_info))
            connection.commit()
            self.user_id = cursor.lastrowid
            cursor.close()
            connection.close()


    @staticmethod
    def fetch_user_by_username(username):
        """
    Fetches a user from the database by their username
    Args:
        username (str): The username of the user to be fetched.
    Returns:
        UserModel or None: Returns a UserModel instance if found, otherwise None.
    """

        connection = Database.connect_db()
        user = None
        if connection:
            cursor = connection.cursor()
            cursor.execute("SELECT * FROM USER WHERE username = %s", (username,))
            user_data = cursor.fetchone()
            if user_data:
                user = UserModel(
                    user_id=user_data[0],
                    username=user_data[1],
                    name=user_data[2],
                    password=user_data[3],  # Assuming password is stored in the database
                    email=user_data[4],
                    contact_info=user_data[5],
                    date_of_birth=user_data[6],
                    gender=user_data[7]
                )
            cursor.close()
            connection.close()

        return user
```

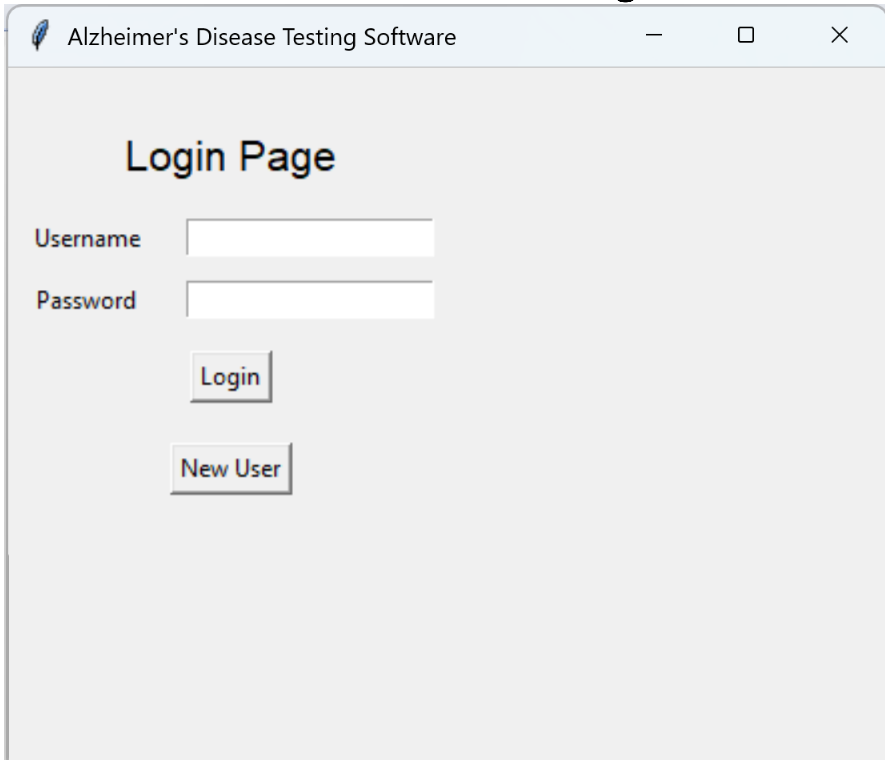
3. Login ViewModel Module

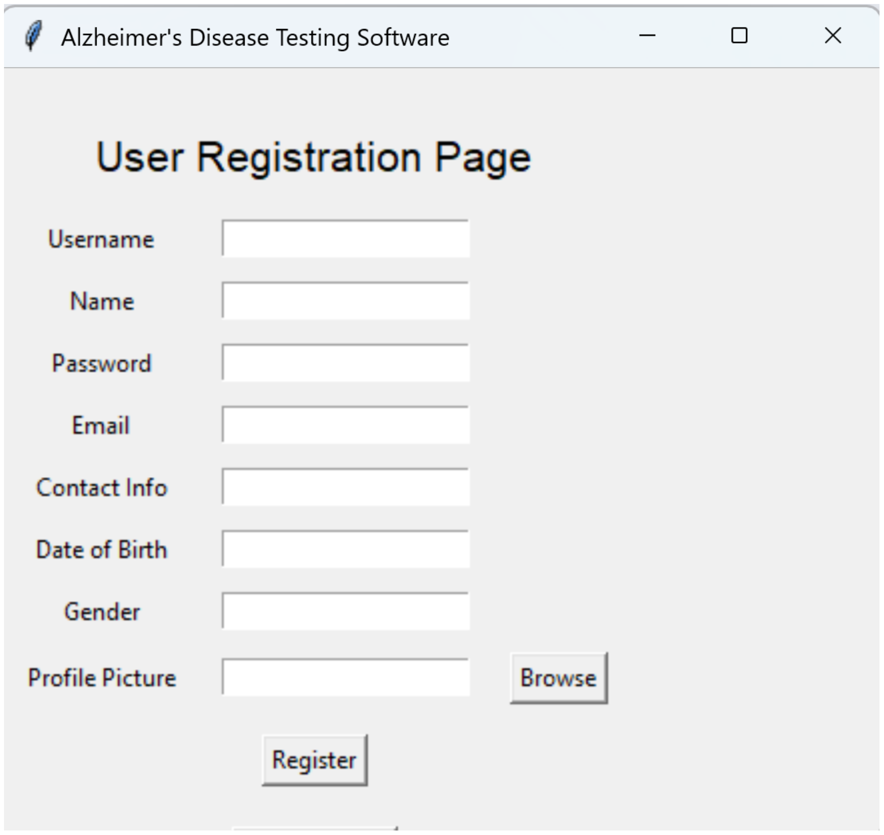
python
```
from user_model import UserModel

# login_view_model.py
class LoginViewModel:
    """Handles user login logic."""

    def __init__(self):
        self.user = None

    def login_user(self, username, password):
        """
        Authenticates a user based on username and password.

        Args:
            username: The username of the user.
            password: The password of the user.

        Returns:
            bool: True if login is successful, False otherwise.
        """

        self.user = UserModel.fetch_user_by_username(username)  # Implement this method
in UserModel
        if self.user and self.user.password == password:  # Assuming password is stored
securely
            return True
        return False


mysql> SELECT * FROM user;
|
```



```
+---------+----------------------+--------------+----------------------+--------------+---------------+--------+
| user_id | username             | password     | email                | contact_info | date_of_birth | gender |
+---------+----------------------+--------------+----------------------+--------------+---------------+--------+
|      16 | josh_kay             | test123      | josh.kay@example.com | 535-5689     | 1968-11-07    | M      |
|      37 | healthcare_provider  | test000      | good.living@gmail.com| 9893331111   |               |        |
|      43 | bill_gates           | test1234578  | bill.gates@gmail.com | 9891004000   | 1954-07-09    | M      |
|      49 | pete_kay             | test1234     | pete.kay@gmail.com   | 9892334000   | 1953-05-06    | M      |
+---------+----------------------+--------------+----------------------+--------------+---------------+--------+
4 rows in set (0.02 sec)
```

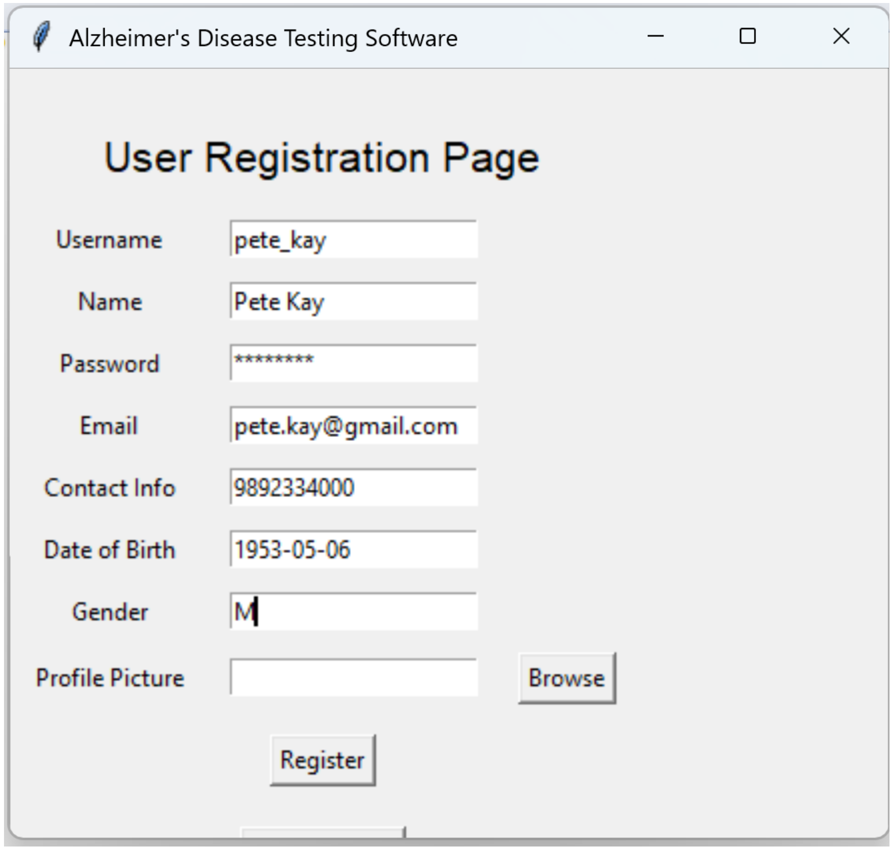
Alzheimer's Disease Testing Software User Interface




Alzheimer's Disease Testing Software

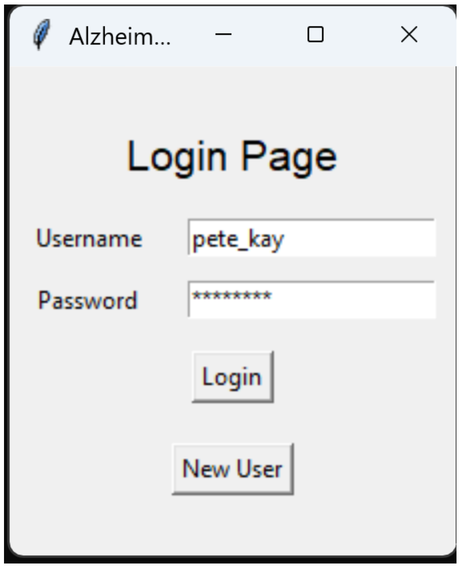
User Registration Page

Username	
Name	
Password	
Email	
Contact Info	
Date of Birth	
Gender	
Profile Picture	


Alzheimer's Disease Testing Software

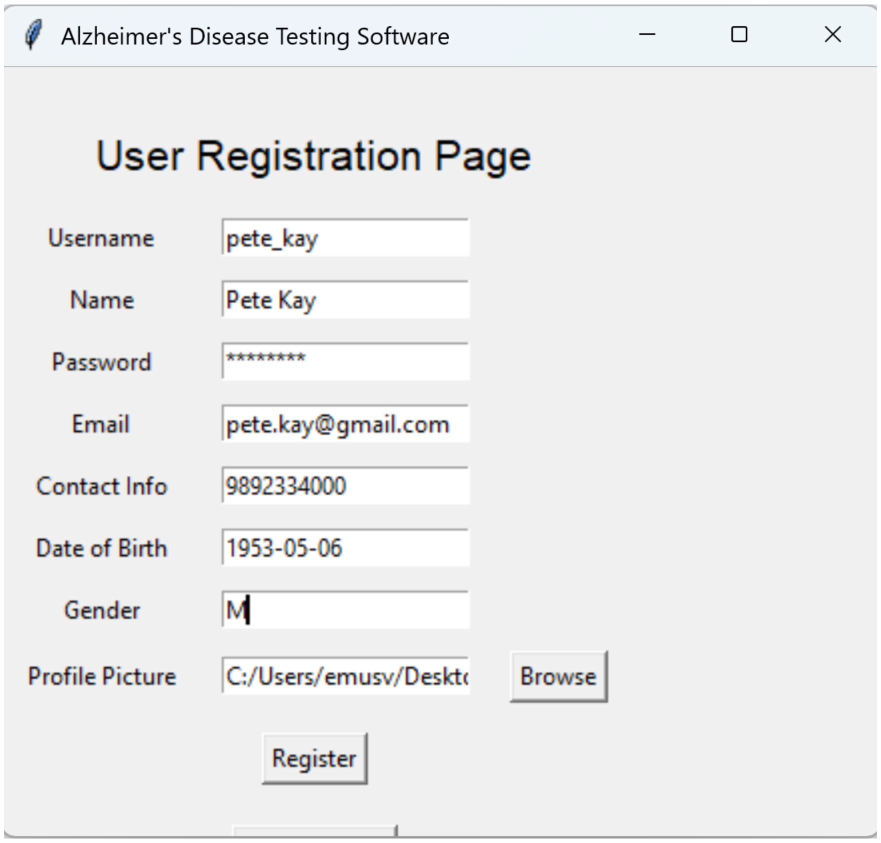
User Registration Page

Username	pete_kay
Name	Pete Kay
Password	********
Email	pete.kay@gmail.com
Contact Info	9892334000
Date of Birth	1953-05-06
Gender	M
Profile Picture	



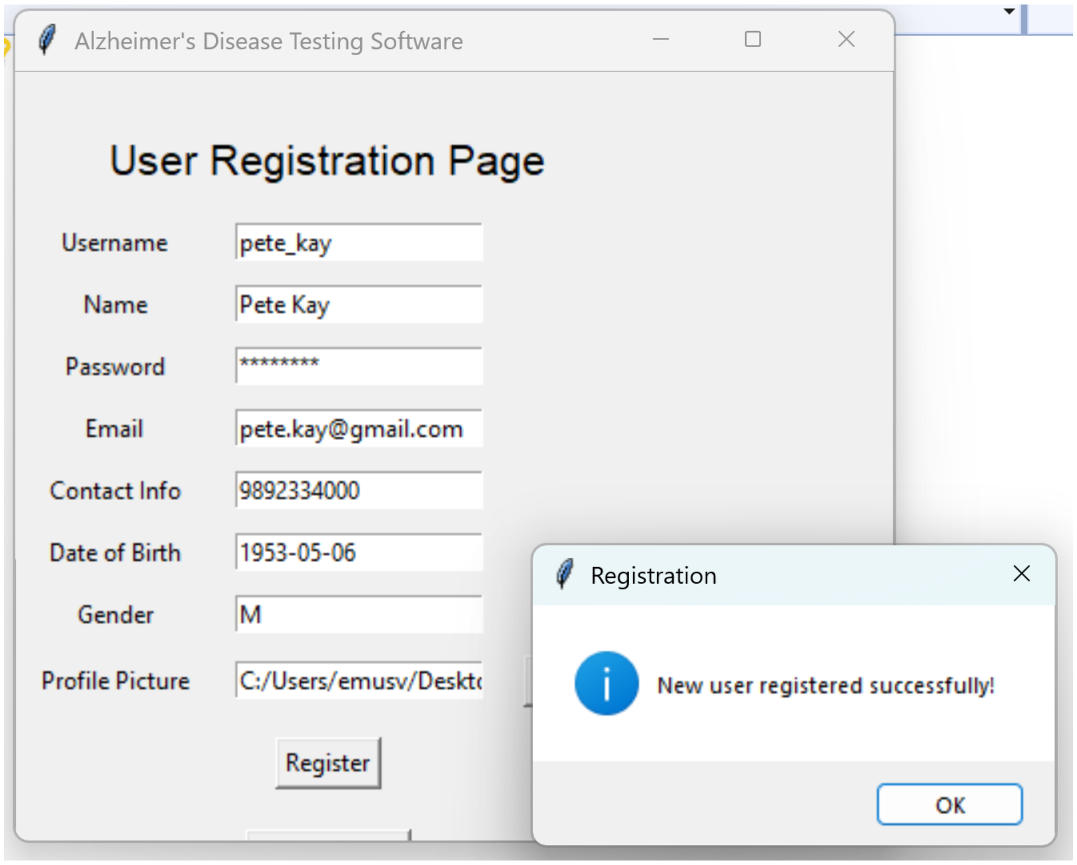
Login Page

Username	pete_kay
Password	********



User Registration Page

Username	pete_kay
Name	Pete Kay
Password	********
Email	pete.kay@gmail.com
Contact Info	9892334000
Date of Birth	1953-05-06
Gender	M
Profile Picture	C:/Users/emusv/Deskt



Alzheimer's Disease Testing Software

User Registration Page

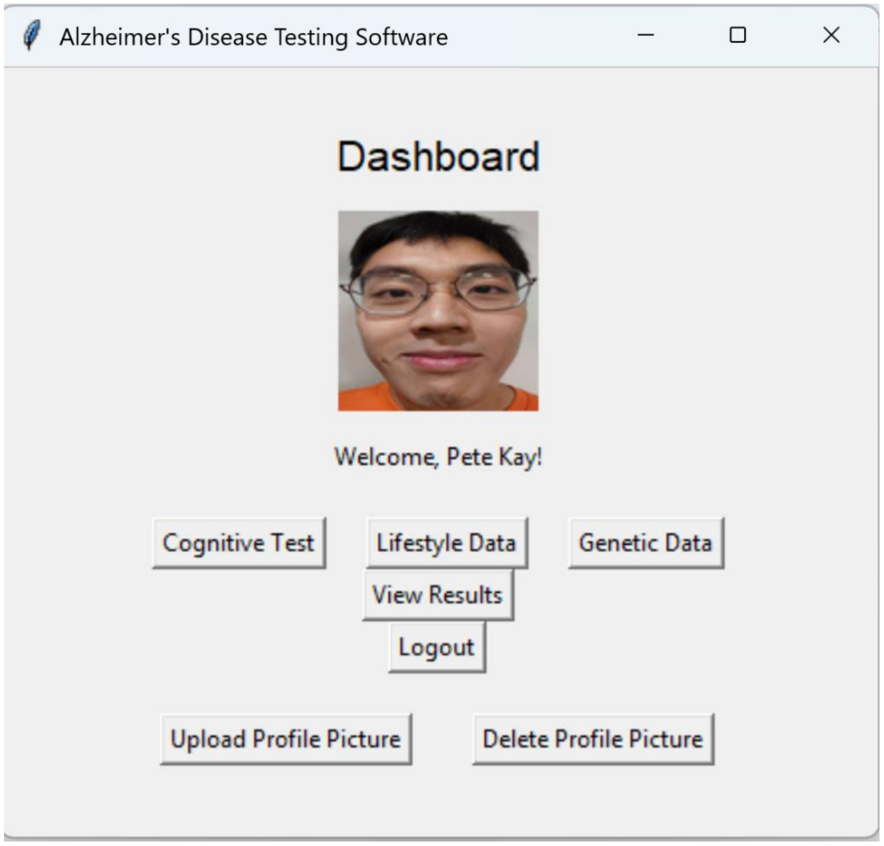
Username	pete_kay
Name	Pete Kay
Password	********
Email	pete.kay@gmail.com
Contact Info	9892334000
Date of Birth	1953-05-06
Gender	M
Profile Picture	C:/Users/emusv/Deskt

Register

Registration

New user registered successfully!

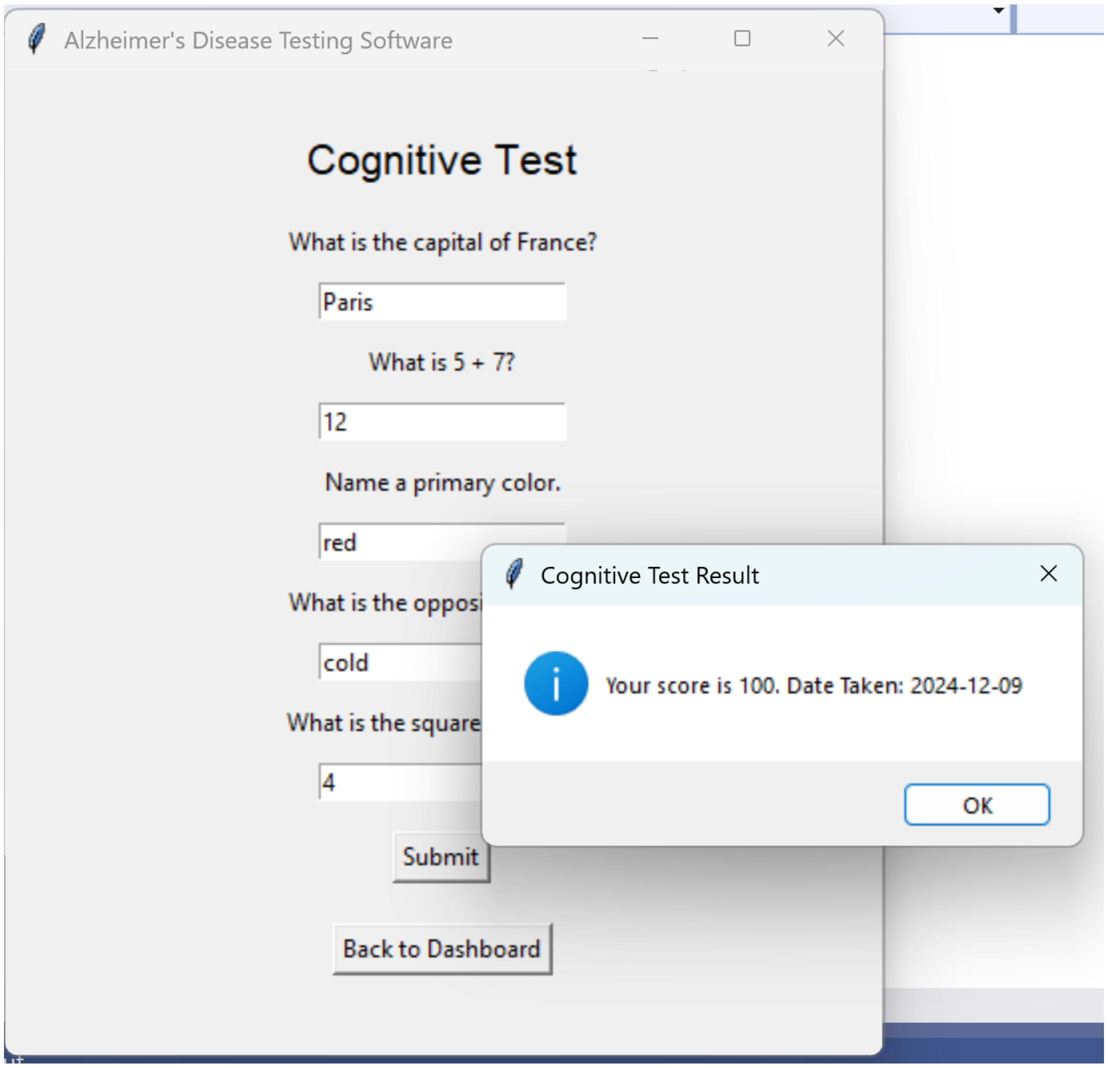
OK



Alzheimer's Disease Testing Software

Dashboard

Welcome, Pete Kay!


Alzheimer's Disease Testing Software

Cognitive Test

What is the capital of France?

Paris

What is 5 + 7?

12

Name a primary color.

red

What is the opposi

cold

What is the square

4

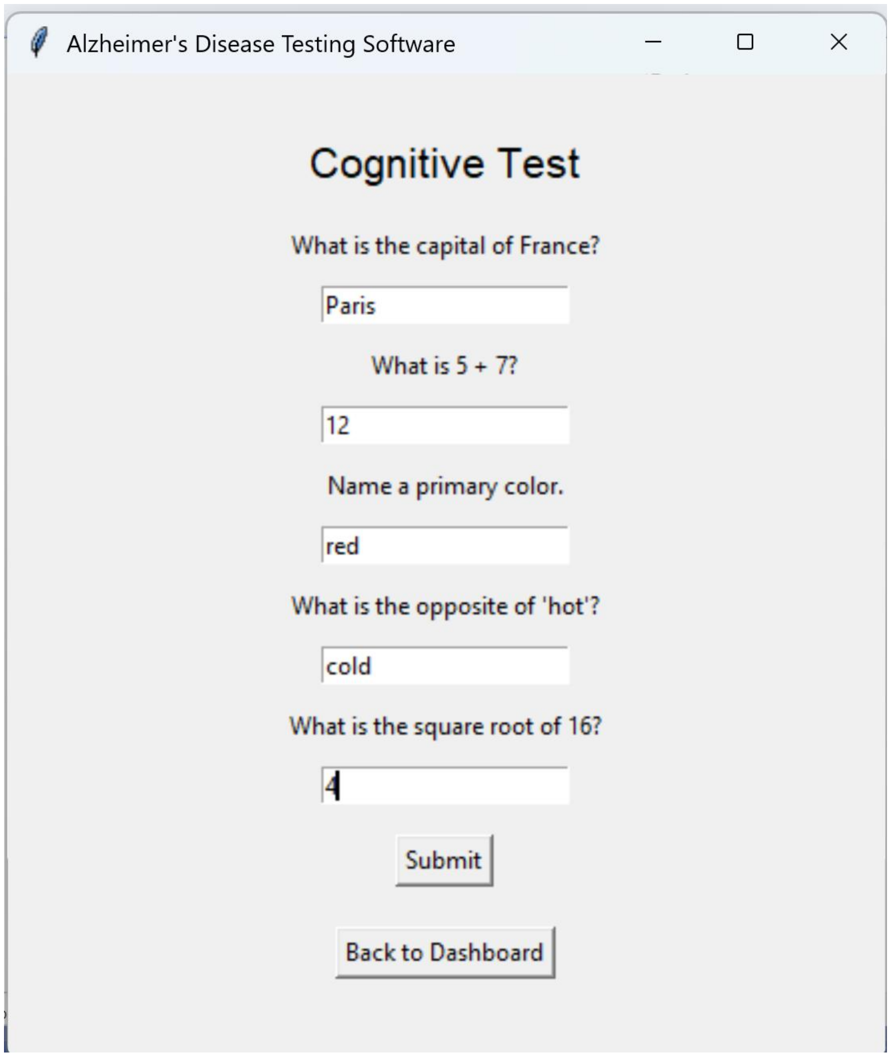
Submit

Back to Dashboard

Cognitive Test Result

Your score is 100. Date Taken: 2024-12-09

OK


Alzheimer's Disease Testing Software

Cognitive Test

What is the capital of France?

Paris

What is 5 + 7?

12

Name a primary color.

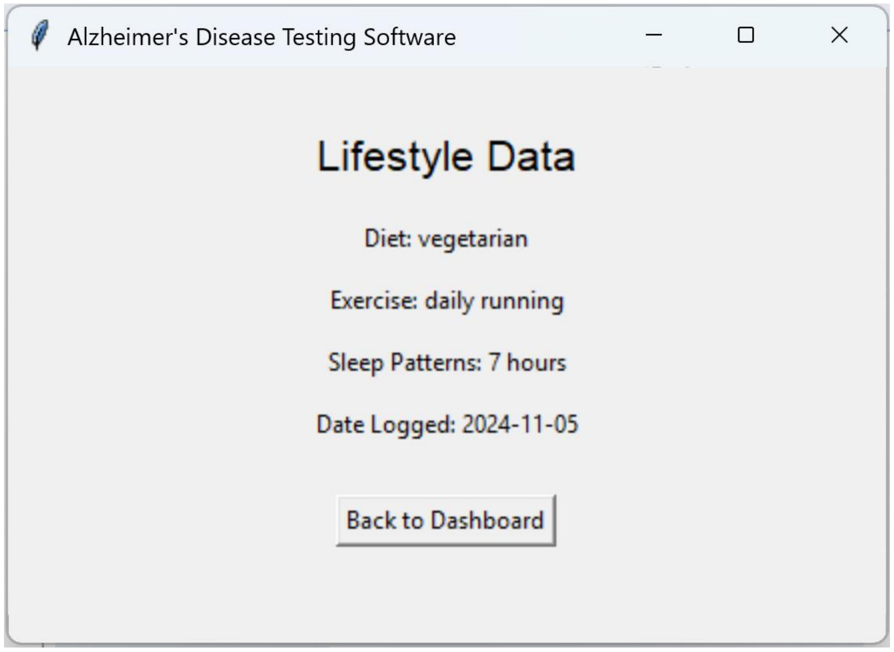
red

What is the opposite of 'hot'?

cold

What is the square root of 16?

4


Alzheimer's Disease Testing Software

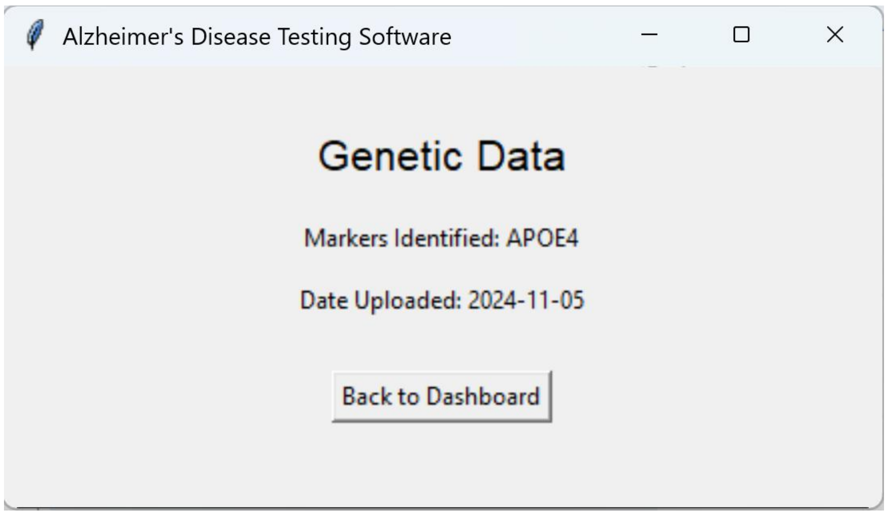
Lifestyle Data

Diet: vegetarian

Exercise: daily running

Sleep Patterns: 7 hours

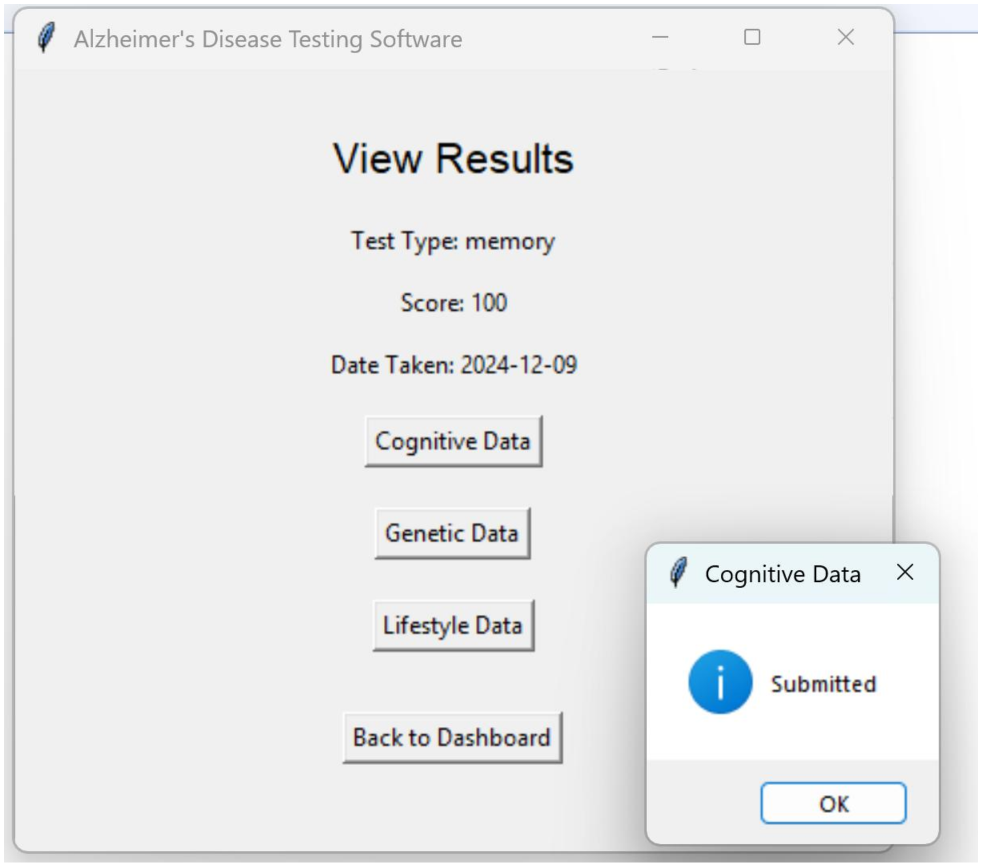
Date Logged: 2024-11-05


Alzheimer's Disease Testing Software

Genetic Data

Markers Identified: APOE4

Date Uploaded: 2024-11-05



Alzheimer's Disease Testing Software

View Results

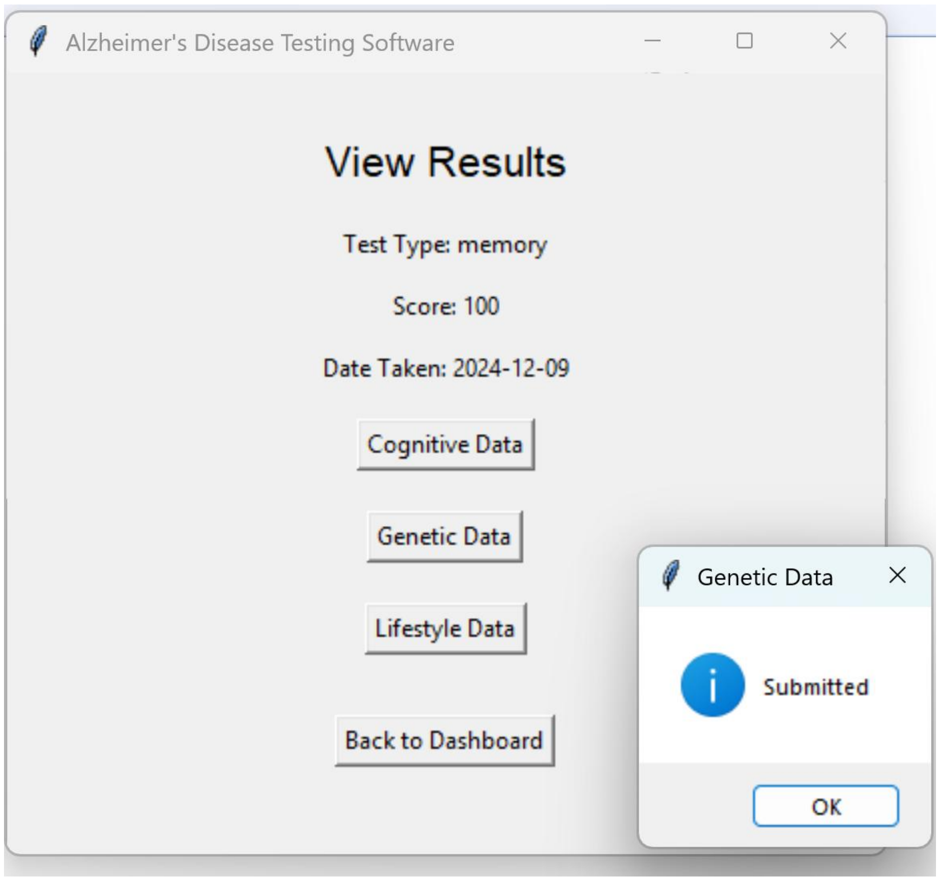
Test Type: memory

Score: 100

Date Taken: 2024-12-09

Cognitive Data

Submitted



Alzheimer's Disease Testing Software

View Results

Test Type: memory

Score: 100

Date Taken: 2024-12-09

Cognitive Data

Genetic Data

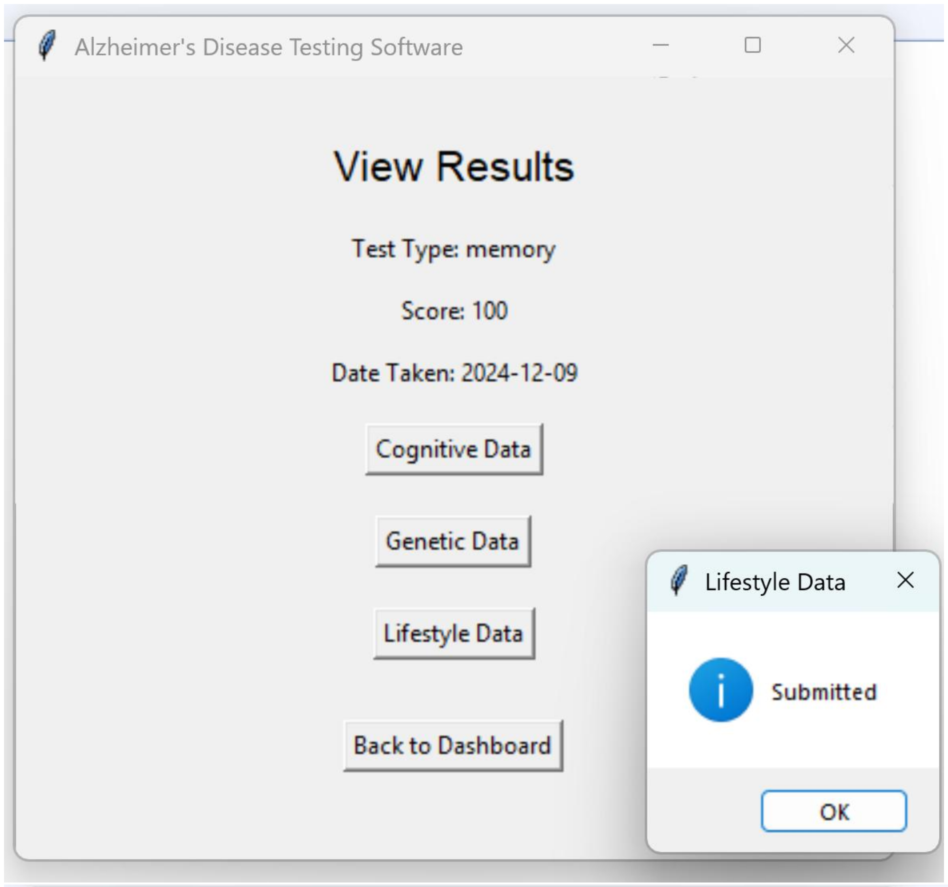
Lifestyle Data

Back to Dashboard

Genetic Data

Submitted

OK


Alzheimer's Disease Testing Software

View Results

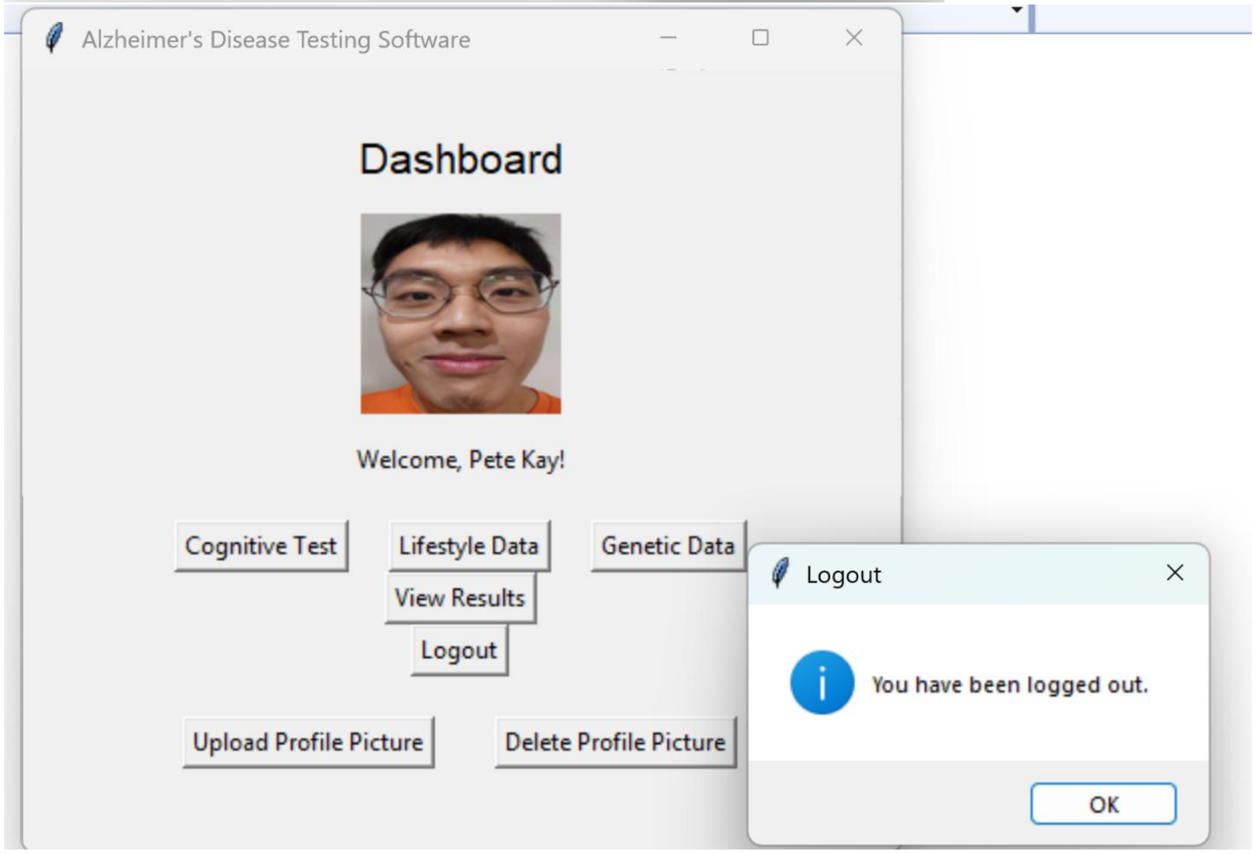
Test Type: memory

Score: 100

Date Taken: 2024-12-09

Lifestyle Data

Submitted

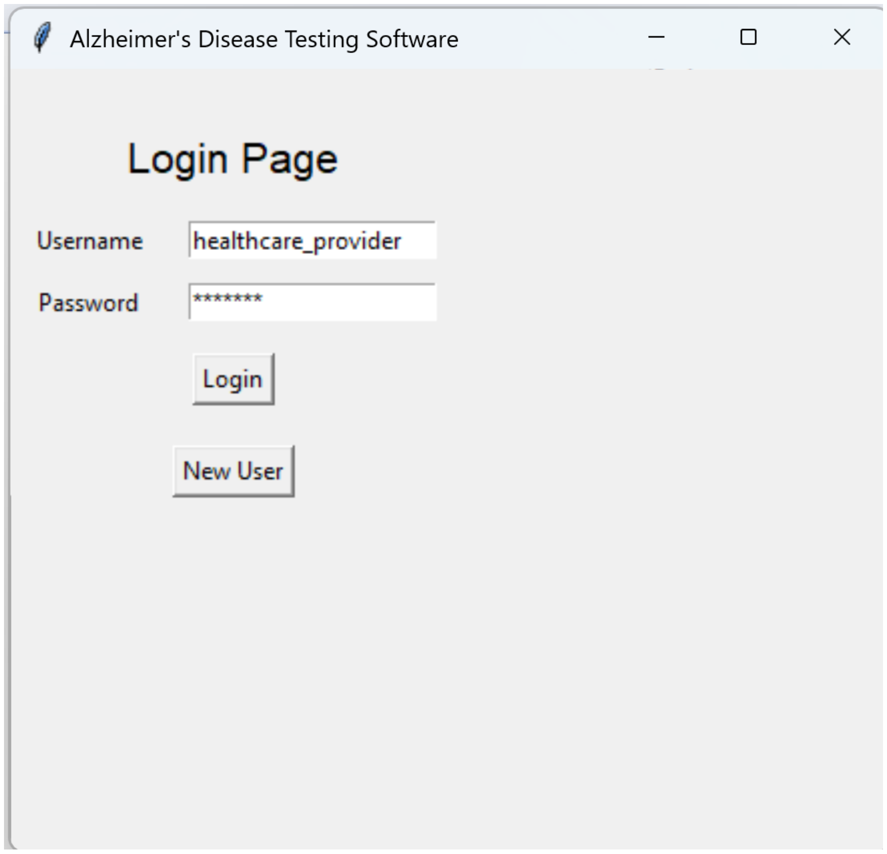

Alzheimer's Disease Testing Software

Dashboard

Welcome, Pete Kay!

Logout

You have been logged out.


Alzheimer's Disease Testing Software

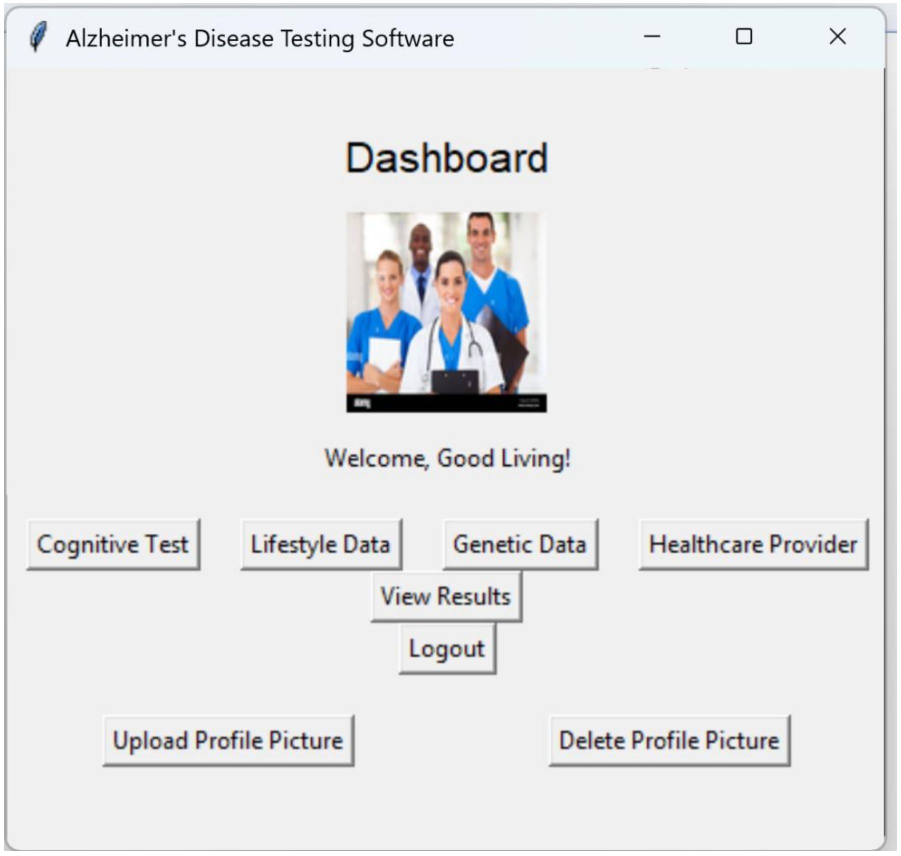
Login Page

Username healthcare_provider

Password *******

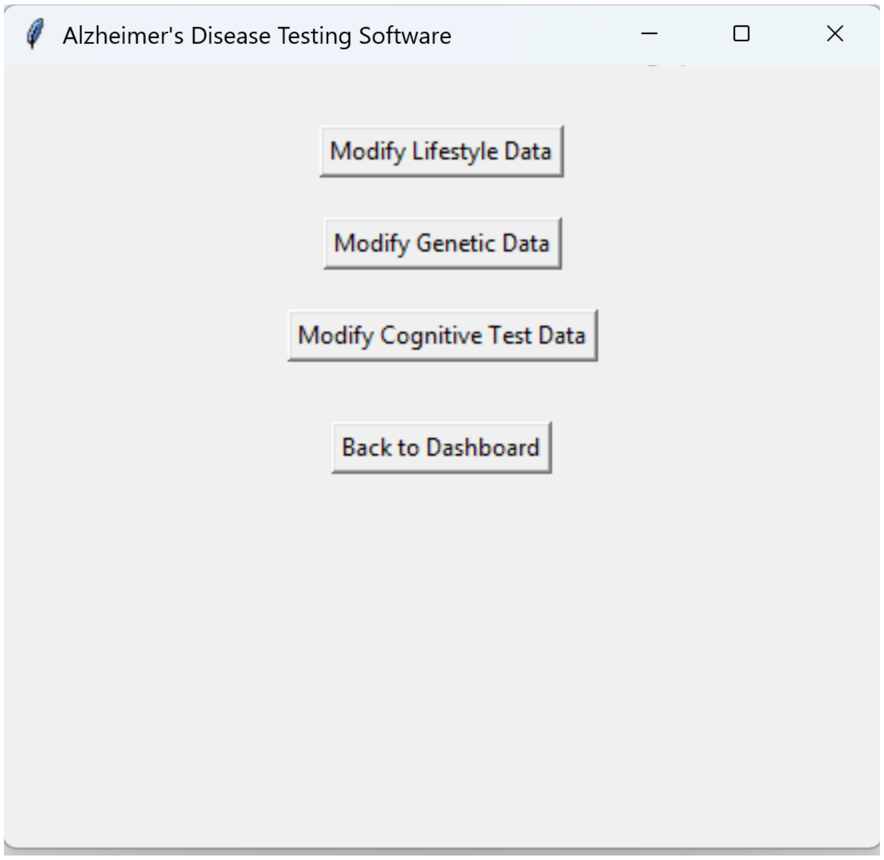
Login

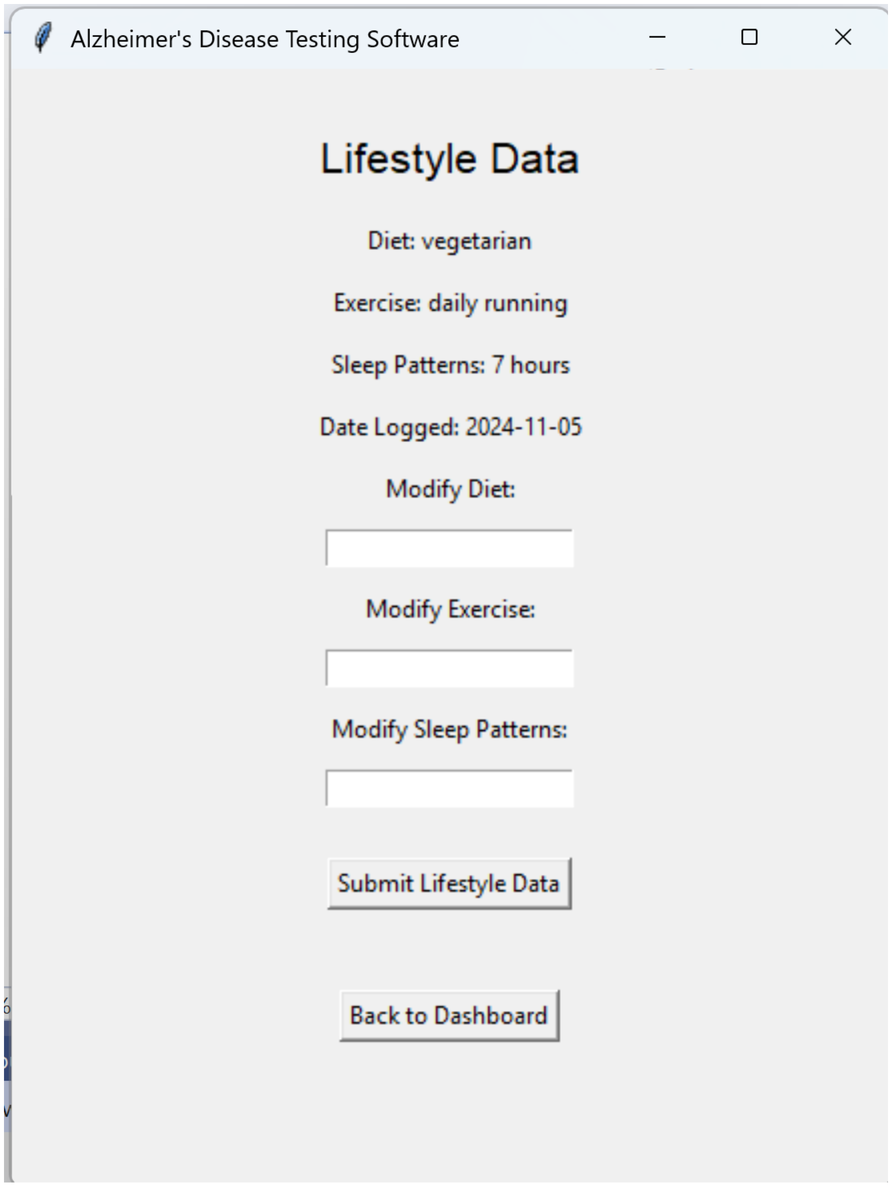
New User


Alzheimer's Disease Testing Software

Dashboard

Welcome, Good Living!


Alzheimer's Disease Testing Software
Modify Lifestyle Data
Modify Genetic Data
Modify Cognitive Test Data
Back to Dashboard


Alzheimer's Disease Testing Software

Lifestyle Data

Diet: vegetarian

Exercise: daily running

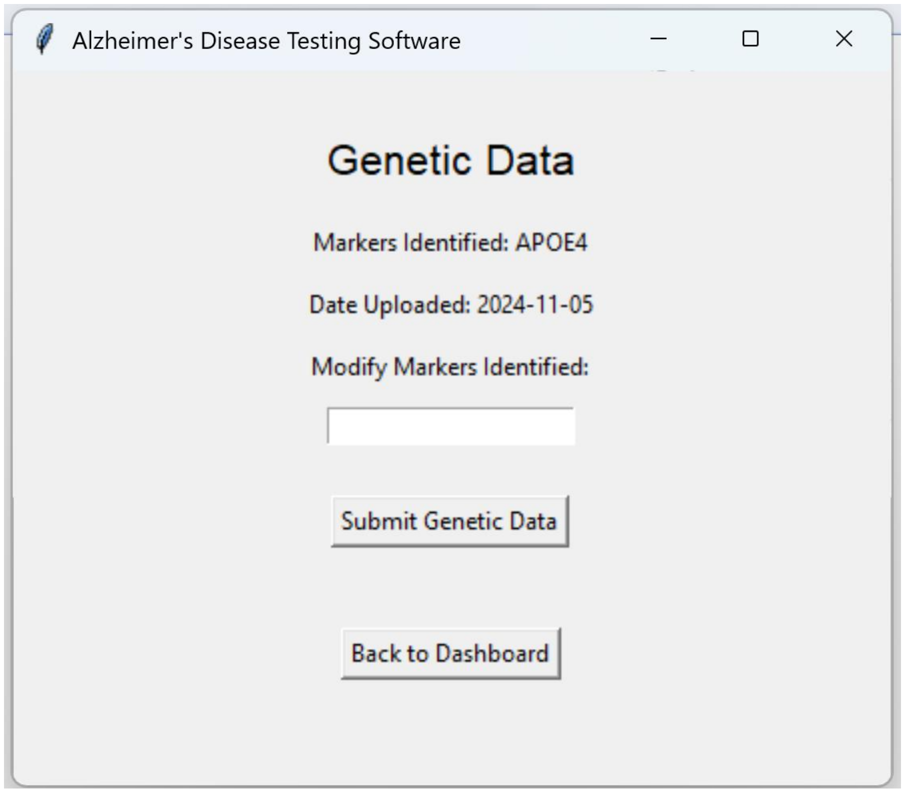
Sleep Patterns: 7 hours

Date Logged: 2024-11-05

Modify Diet:

Modify Exercise:

Modify Sleep Patterns:



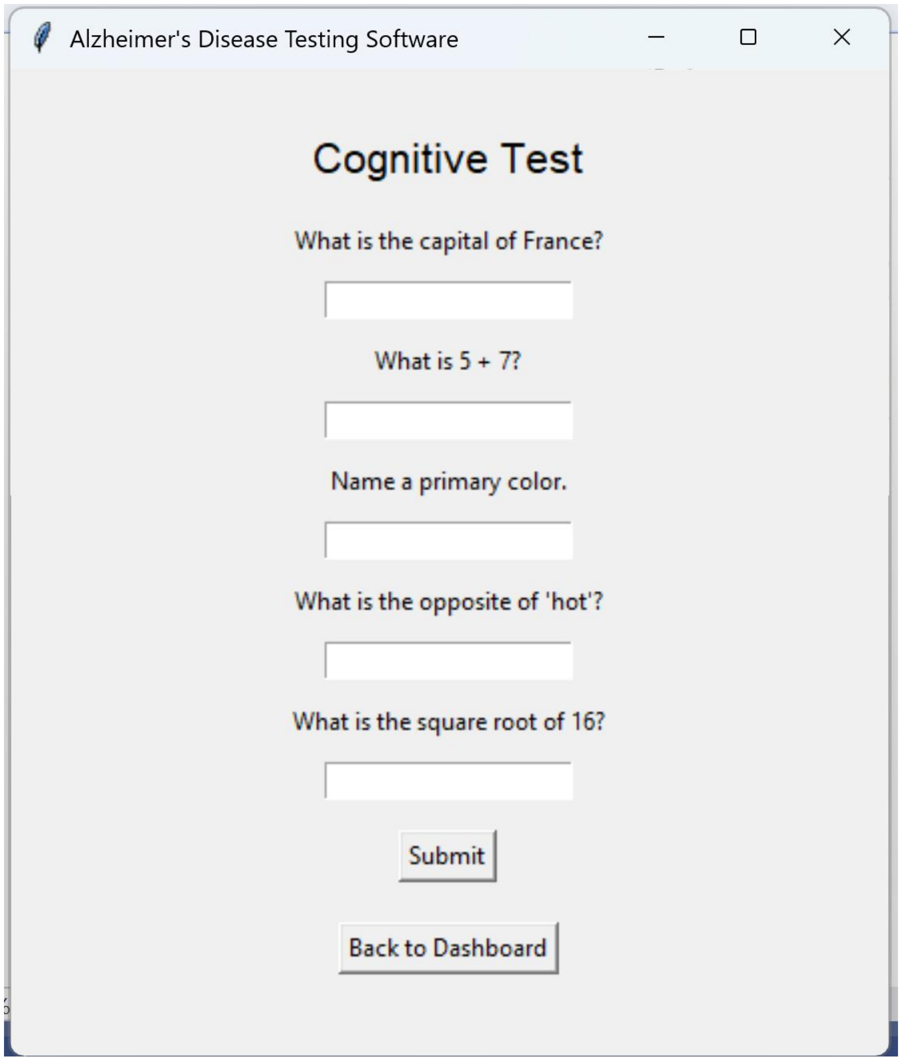
Alzheimer's Disease Testing Software

Genetic Data

Markers Identified: APOE4

Date Uploaded: 2024-11-05

Modify Markers Identified:



Alzheimer's Disease Testing Software

Cognitive Test

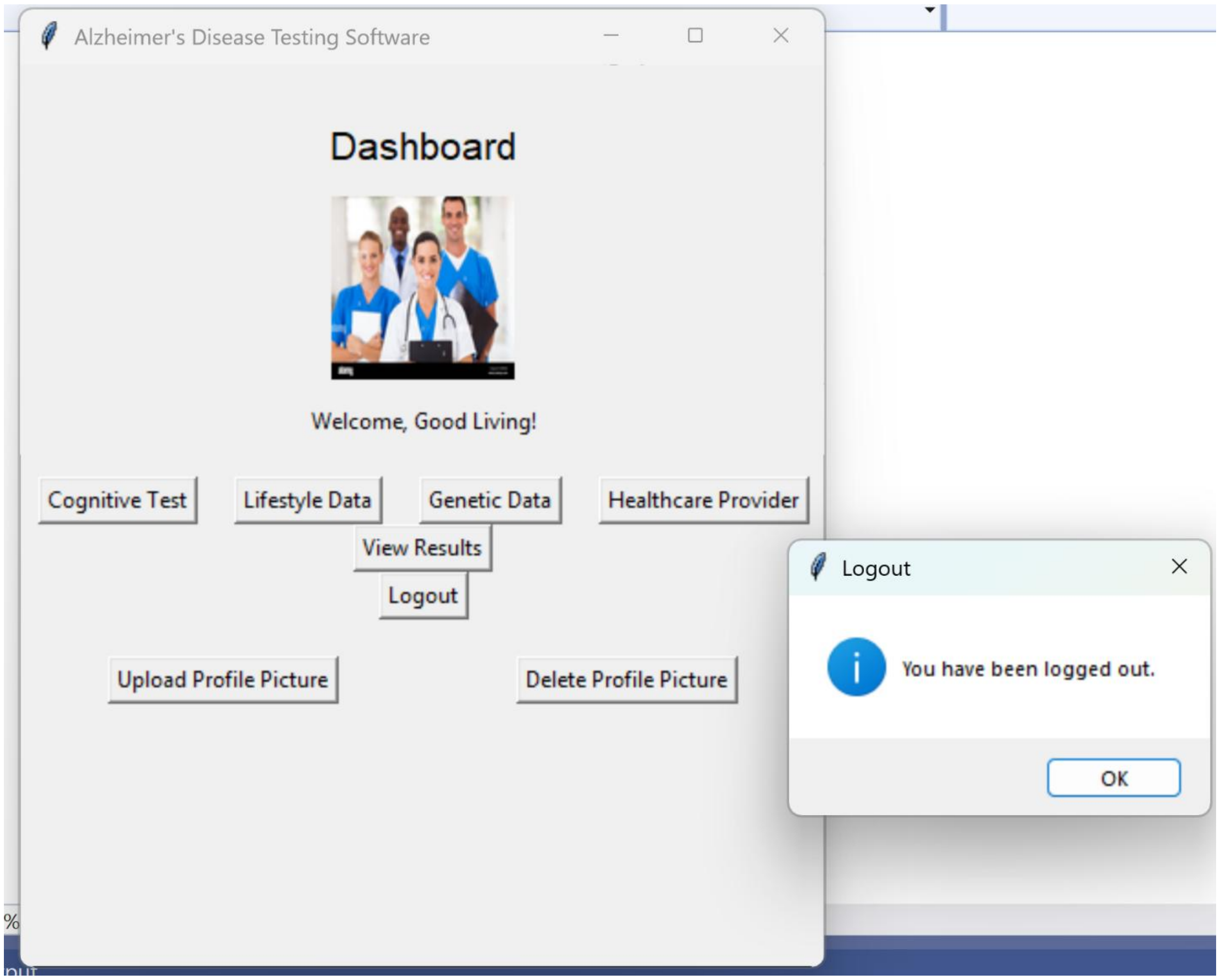
What is the capital of France?

What is 5 + 7?

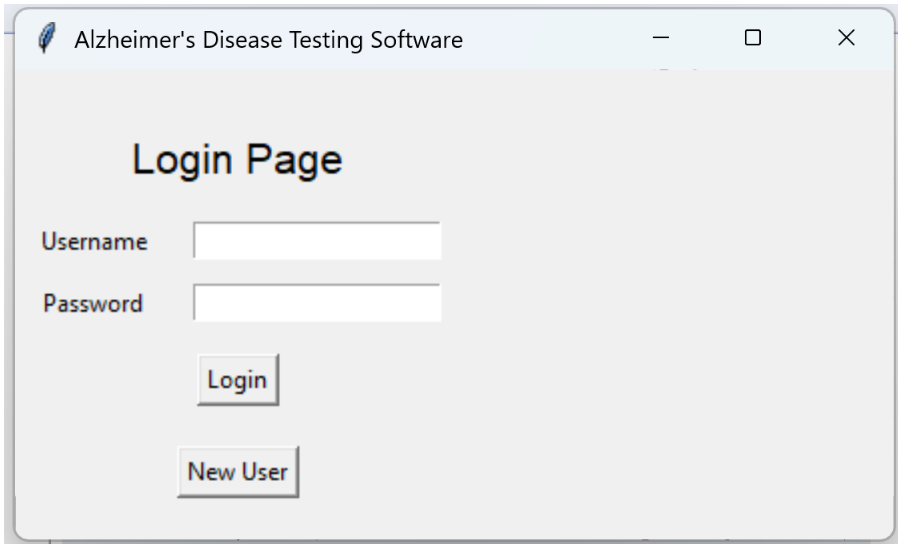
Name a primary color.

What is the opposite of 'hot'?

What is the square root of 16?



You have been logged out.


Alzheimer's Disease Testing Software
Login Page
Username
Password
Login
New User