Unit Testing Report

Please provide your GitHub repository link.

GitHub Repository URL: https://github.com/DaicosJ/Milestone2_Project.git

---

The testing report should focus solely on testing all the self-defined functions related to the five required features. There is no need to test the GUI components. Therefore, it is essential to decouple your code and separate the logic from the GUI-related code.

1. **Test Summary**

list all tested functions related to the five required features and the corresponding test functions designed to test those functions, for example:

Tested Functions	Test Functions
load_dataset(file_path)	test_load_dataset()
search_nutritional_values(df, protein)`	test_search_nutritional_values(sample_data)
plot_nutrition(df)	test_plot_nutrition(sample_data)`
reset_filters(df)	test_reset_filters(sample_data)
save_results_to_csv(df, file_path)	test_save_results_to_csv(sample_data)

---

2. **Test Case Details**

Test Case 1:

- **Test Function/Module**
  - `test_load_dataset()`
- **Tested Function/Module**
  - `load_dataset(file_path)`
- **Description**
  - Loads a CSV dataset into Dataframe to ensure valid dataframe contains necessary columns.
- **1) Valid Input and Expected Output**

Valid Input	**Expected Output**
`./Food-Nutrition_Dataset.csv`	`Dataframe with food column and data`

- **1) Code for the Test Function**

```python
def test_load_dataset():
    df = load_dataset('./Food_Nutrition_Dataset.csv')
    assert isinstance(df, pd.DataFrame), "Loaded data is not a DataFrame"
    assert not df.empty, "Loaded DataFrame is empty"
    assert 'food' in df.columns, "Column 'food' not found in the DataFrame"
```

- **2) Invalid Input and Expected Output**

Invalid Input	Expected Output
`add more cases in necessary`	`...`

- **2) Code for the Test Function**

```

```

Test Case 2:

- **Test Function/Module**
  - `test_search_nutritional_values(sample_data)`
- **Tested Function/Module**
  - `search_nutritional_values(df, protein)`
- **Description**
  - Filters the dataset based on protein content, returning rows that meet specified criteria
- **1) Valid Input and Expected Output**

Valid Input	Expected Output
`sample_data, protein=1`	`3 rows`
`sample_data, protein=0.5`	`3 rows`
`sample_data, protein=10`	`3 rows`
`sample_data, protein=60`	`3 rows`

- **1) Code for the Test Function**

```
def test_search_nutritional_values(sample_data):
    filtered_df = search_nutritional_values(sample_data, protein=1)
    assert len(filtered_df) == 3, "Expected 3 results for protein >= 1"

    filtered_df = search_nutritional_values(sample_data, protein=0.5)
    assert len(filtered_df) == 3, "Expected 3 results for protein >= 0.5"

    filtered_df = search_nutritional_values(sample_data, protein=10)
    assert len(filtered_df) == 1, "Expected 3 results for protein >= 10"

    filtered_df = search_nutritional_values(sample_data, protein=60)
    assert len(filtered_df) == 0, "Expected 3 results for protein >= 60"
```

- **2) Invalid Input and Expected Output**

Invalid Input	Expected Output
`add more cases in necessary`	`...`

- **2) Code for the Test Function**

```
def test_divide_invalid():
    with pytest.raises(ValueError) as exc_info:
        divide(10, 0)
    assert exc_info.type is ValueError
```

Test Case 3:

- **Test Function/Module**
  - `test_plot_nutrition(sample_data)`
- **Tested Function/Module**
  - `plot_nutrition(df)`
- **Description**
  - Generates a plot of nutritional values from Dataframe, Checks exceptions during plotting.
- **1) Valid Input and Expected Output**

Valid Input	Expected Output
`add more cases in necessary`	`...`

- **1) Code for the Test Function**

```
    try:
        plot_nutrition(sample_data)
    except Exception as e:
        pytest.fail(f"plot_nutrition raised as an exception: {e}")
```

- **2) Invalid Input and Expected Output**

Invalid Input	Expected Output
`divide(10, 0)`	`Handle Exception`
`add more cases in necessary`	`...`

- **2) Code for the Test Function**

```

```

Test Case 4:

- **Test Function/Module**

    - `test_reset_filters(sample_data)`
- **Tested Function/Module**
    - `reset_filters(df)`
- **Description**
    - Resets the aplied filters on the Dataframe and return original Dataframe.
- **1) Valid Input and Expected Output**

Valid Input	Expected Output
`divide(10, 2)`	`5`
`divide(10, -2)`	`-5`
`add more cases in necessary`	`...`

- **1) Code for the Test Function**

```
def test_reset_filters(sample_data):
    reset_df = reset_filters(sample_data)
    pd.testing.assert_frame_equal(reset_df, sample_data, "reset_filters did not
return the original DataFrame")
```

- **2) Invalid Input and Expected Output**

Invalid Input	Expected Output
`divide(10, 0)`	`Handle Exception`

- **2) Code for the Test Function**

```

```

Test Case 5:

- **Test Function/Module**
    - `def test_save_results_to_csv(sample_data, test_file_path)`
- **Tested Function/Module**
    - `save_results_to_csv(df, file_path)`
- **Description**
    - Saves Dataframe to a CSV file and checks if the file is created and saved correctly.
- **1) Valid Input and Expected Output**

Valid Input	Expected Output
`sample_data, 'test_filtered_results.csv'`	`CSV file created with matching data`

- **1) Code for the Test Function**

```python
def test_save_results_to_csv(sample_data,):
    test_file_path = 'test_filtered_results.csv'
    save_results_to_csv(sample_data, test_file_path)

    assert os.path.exists(test_file_path), f"CSV file was not created at
{test_file_path}"
    if os.path.exists(test_file_path):
        os.remove(test_file_path)

    assert os.path.exists(test_file_path), "CSV file was not created"
    saved_df = pd.read_csv(test_file_path)
    pd.testing.assert_frame_equal(saved_df, sample_data, "Saved data does not
match the original DataFrame")

    os.remove(test_file_path)
```

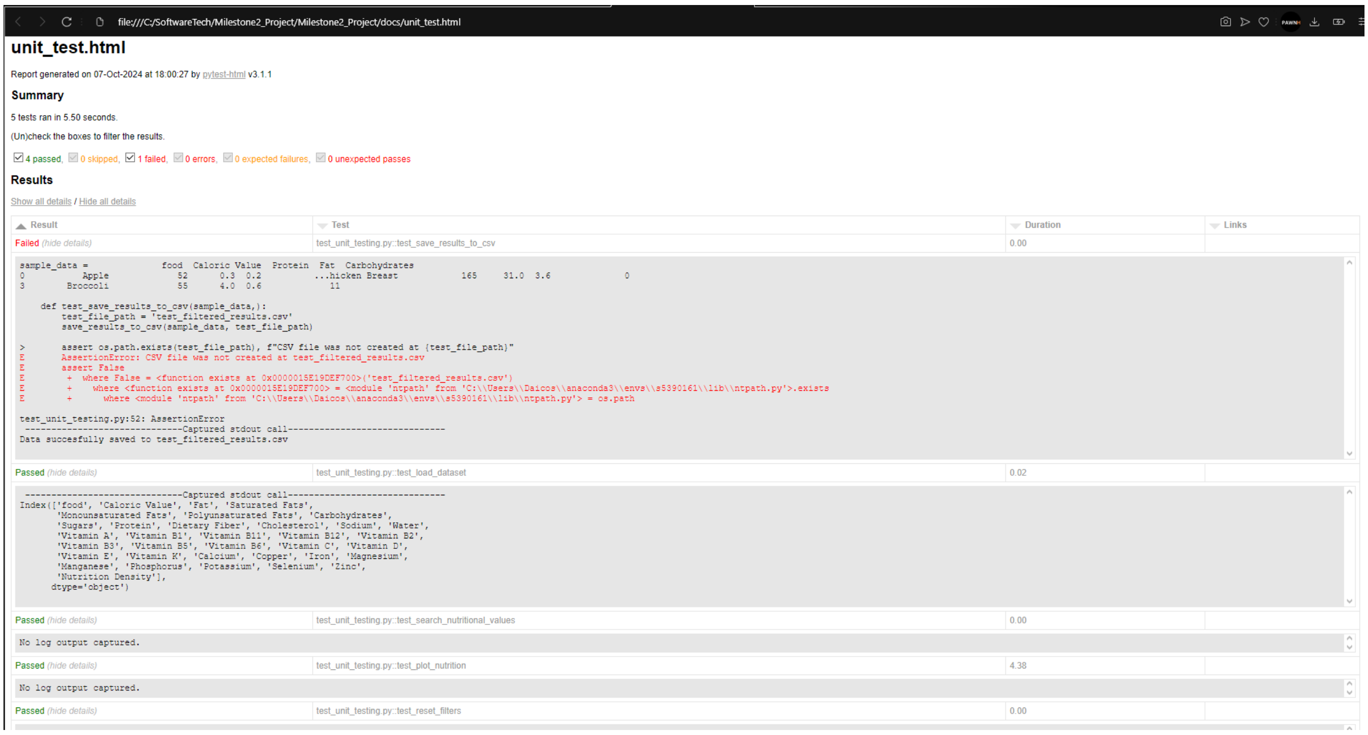
- **2) Invalid Input and Expected Output**

Invalid Input	Expected Output
`divide(10, 0)`	`Handle Exception`
`add more cases in necessary`	`...`

- **2) Code for the Test Function**

```
```

3. **Testing Report Summary**



file:///C:/SoftwareTech/Milestone2_Project/Milestone2_Project/docs/unit_test.html

unit_test.html

Report generated on 07-Oct-2024 at 18:00:27 by pytest-html v3.1.1

Summary

5 tests ran in 5.50 seconds.

(Un)check the boxes to filter the results.

☑ 4 passed, ☑ 0 skipped, ☑ 1 failed, ☑ 0 errors, ☑ 0 expected failures, ☑ 0 unexpected passes

Results

Show all details / Hide all details

Result	Test	Duration	Links
Failed (hide details)	test_unit_testing.py::test_save_results_to_csv	0.00	

```
sample_data =             food  Caloric Value  Protein  Fat  Carbohydrates
0         Apple            52      0.3  0.2          ...hicken Breast          165     31.0  3.6            0
3       Broccoli           55      4.0  0.6             11

    def test_save_results_to_csv(sample_data,):
        test_file_path = 'test_filtered_results.csv'
        save_results_to_csv(sample_data, test_file_path)

>       assert os.path.exists(test_file_path), f"CSV file was not created at {test_file_path}"
E       AssertionError: CSV file was not created at test_filtered_results.csv
E       assert False
E        +  where False = <function exists at 0x0000015E19DEF700>('test_filtered_results.csv')
E        +    where <function exists at 0x0000015E19DEF700> = <module 'ntpath' from 'C:\\Users\\Daicos\\anaconda3\\envs\\s5390161\\lib\\ntpath.py'>.exists
E        +      where <module 'ntpath' from 'C:\\Users\\Daicos\\anaconda3\\envs\\s5390161\\lib\\ntpath.py'> = os.path

test_unit_testing.py:52: AssertionError
 ------------------------------Captured stdout call------------------------------
Data succesfully saved to test_filtered_results.csv
```

Result	Test	Duration	Links
Passed (hide details)	test_unit_testing.py::test_load_dataset	0.02	

```
 ------------------------------Captured stdout call------------------------------
Index(['food', 'Caloric Value', 'Fat', 'Saturated Fats',
       'Monounsaturated Fats', 'Polyunsaturated Fats', 'Carbohydrates',
       'Sugars', 'Protein', 'Dietary Fiber', 'Cholesterol', 'Sodium', 'Water',
       'Vitamin A', 'Vitamin B1', 'Vitamin B11', 'Vitamin B12', 'Vitamin B2',
       'Vitamin B3', 'Vitamin B5', 'Vitamin B6', 'Vitamin C', 'Vitamin D',
       'Vitamin E', 'Vitamin K', 'Calcium', 'Copper', 'Iron', 'Magnesium',
       'Manganese', 'Phosphorus', 'Potassium', 'Selenium', 'Zinc',
       'Nutrition Density'],
      dtype='object')
```

Result	Test	Duration	Links
Passed (hide details)	test_unit_testing.py::test_search_nutritional_values	0.00	

```
No log output captured.
```

Result	Test	Duration	Links
Passed (hide details)	test_unit_testing.py::test_plot_nutrition	4.38	

```
No log output captured.
```

Result	Test	Duration	Links
Passed (hide details)	test_unit_testing.py::test_reset_filters	0.00	