Unit Testing Report

Please provide your GitHub repository link.

GitHub Repository URL: https://github.com/JavaTheHut17/SoftTech_Group_06

1. Test Case Details

Test Case 1:

- **Test Function/Module**
  - `test_search_food_keyword_found()`
  - `test_search_food_keyword_partial_match()`
  - `test_search_food_keyword_not_found()`
  - `test_search_food_invalid_db()`
- **Tested Function/Module**
  - `search_food(keyword, data)`
- **Description**
  - test the search food items function on a variety of inputs and testcases.
- **1) Valid Input and Expected Output**

Valid Input	Expected Output
`search_food('watermelon', db)`	`['watermelon', 'watermelon seed kernels dried']`
`search_food('phosphorus', db)`	`['phosphorus']`

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

```
def test_search_food_keyword_found():
    expect_res = ['watermelon', 'watermelon seed kernels dried']

    func_res = search_food('watermelon', db)

    assert func_res == expect_res
```

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

Invalid Input	Expected Output
`search_food('slurpee', db)`	`'No matches found for keyword: slurpee"`
`search_food('cheese', 'Invalid_DB.csv')`	`"Error: Database file not found."`

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

```
def test_search_food_keyword_not_found():
    expect_res = "No matches found for keyword: slurpee"

    func_res = search_food('slurpee', db)

    assert func_res == expect_res

def test_search_food_invalid_db():
    expect_res = "Error: Database file not found."

    func_res = search_food('cheese', 'Invalid_DB.csv')

    assert func_res == expect_res
```

Test Case 2:

- **Test Function/Module**
  - `test_nutrition_breakdown_valid_food()`
  - `test_nutrition_breakdown_invalid_food()`
  - `test_nutrition_breakdown_invalid_db()`
- **Tested Function/Module**
  - `nutrition_breakdown(food_name, data)`
- **Description**
  - To test the nutritional breakdown based on the database
- **1) Valid Input and Expected Output**

Valid Input	Expected Output
`nutrition_breakdown('cream cheese', db)`	`"Nutritional breakdown for cream cheese displayed."`
`nutrition_breakdown('watermelon', db)`	`"Nutritional breakdown for watermelon displayed."`

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

```
def test_nutrition_breakdown_valid_food():
    result = nutrition_breakdown('cream cheese', db)
    assert result == "Nutritional breakdown for cream cheese displayed."
```

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

Invalid Input	Expected Output
`nutrition_breakdown('slurpee', db)`	`"Error: slurpee not found in the database."`

Invalid Input	Expected Output
`nutrition_breakdown('cream cheese', 'Invalid_DB.csv')`	`"Error: Database file not found."`

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

```
def test_nutrition_breakdown_invalid_food():
    result = nutrition_breakdown('slurpee', db)
    assert result == "Error: slurpee not found in the database."

def test_nutrition_breakdown_invalid_db():
    result = nutrition_breakdown('cream cheese', 'Invalid_DB.csv')
    assert result == "Error: Database file not found."
```

Test Case 3:

- **Test Function/Module**
  - `test_range_filter()`
  - `test_range_filter_no_res()`
  - `test_range_filter_invalid_name()`
  - `test_range_filter_invalid_value()`
  - `test_range_filter_invalid_value2()`
  - `test_range_filter_no_db()`
- **Tested Function/Module**
  - `range_filter(nutri_component, max_n_value, min_n_value, data)`
- **Description**
- Test the range filter function on a variety of inputs and testcases.
- **1) Valid Input and Expected Output**

Valid Input	Expected Output
`range_filter('Vitamin E', 0.08, 0.079, db)`	`[{'rice bowl with chicken, Vitamin E: 0.08'}, {'strawberry topping, Vitamin E: 0.079'}, ...]`
`range_filter('Vitamin D', 300, 230, db)`	`[]`

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

```
def test_range_filter():

  db = 'DataBase/Food_Nutrition_Dataset.csv'

  expect_res = [{'rice bowl with chicken, Vitamin E: 0.08'}, {'strawberry topping, Vitamin E: 0.079'},
                {'instant white rice raw, Vitamin E: 0.079'}, {'strawberries, Vitamin E: 0.079'},
                {'sage ground, Vitamin E: 0.08'}, {'navy beans raw, Vitamin E: 0.08'},
                {'navy beans cooked, Vitamin E: 0.08'}, {'veal thymus cooked, Vitamin E: 0.079'},
                {'fruit yogurt, Vitamin E: 0.079'}, {'instant cappuccino powder, Vitamin E: 0.08'},
```

```
                {'pineapple orange juice, Vitamin E: 0.079'}, {'chestnuts roasted, Vitamin E: 0.08'},
                {'hot chile pepper dried, Vitamin E: 0.079'}, {'dried tomatoes, Vitamin E: 0.079'},
                {'shiitake mushrooms cooked, Vitamin E: 0.08'}]

  func_res = range_filter('Vitamin E', 0.08, 0.079, db)

  assert func_res == expect_res

def test_range_filter_no_res():

  expect_res = []

  func_res = range_filter('Vitamin D', 300, 230, 'DataBase/Food_Nutrition_Dataset.csv')

  assert func_res == expect_res
```

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

Invalid Input	Expected Output
`range_filter('Vitamin Q', 300, 230, db)`	`"Error: Nutrient component not found in database"`
`range_filter('Vitamin E', '300', 230, db)`	`"Error: Min and Max value must be integer or float"`
`range_filter('Vitamin E', 300, '230', db)`	`"Error: Min and Max value must be integer or float"`
`range_filter('Vitamin E', 300, 230, 'DataBase')`	`"Error: DB not found."`

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

```
def test_range_filter_no_db():

  expect_res = print("Error: DB not found.")

  func_res = range_filter('Vitamin E', 300, 230, 'DataBase')

  assert func_res == expect_res

  def test_range_filter_invalid_value2():

    expect_res = print("Error: Min and Max value must be integer or float")

    func_res = range_filter('Vitamin E', 300, '230', 'DataBase/Food_Nutrition_Dataset.csv')

    assert func_res == expect_res
def test_range_filter_invalid_value():

  expect_res = print("Error: Min and Max value must be integer or float")

  func_res = range_filter('Vitamin E', '300', 230, 'DataBase/Food_Nutrition_Dataset.csv')
```

```
    assert func_res == expect_res

def test_range_filter_invalid_name():

    expect_res = print("Error: Nutrient component not found in database")

    func_res = range_filter('Vitamin Q', 300, 230, 'DataBase/Food_Nutrition_Dataset.csv')

    assert func_res == expect_res
```

Test Case 4:

- **Test Function/Module**
  - `test_high_med_low_filter_high()`
  - `test_high_med_low_filter_low()`
  - `test_high_med_low_filter_med()`
  - `test_high_med_low_invalid_nurti()`
  - `test_high_med_low_db_error()`
- **Tested Function/Module**
  - `high_med_low_filter('Vitamin Q', high=False, med=True, low=False, data='DataBase/Food_Nutrition_Dataset.csv')`
- **Description**
  - To test the high, med, low function of a variety of inputs, high, med, low.
- **1) Valid Input and Expected Output**

Valid Input	Expected Output
`high_med_low_filter('Vitamin D', high=True, med=False, low=False, data=db)`	`[{'pokeberry shoots raw, Vitamin D: 217.6'}, {'tomato juice, Vitamin D: 170.3'}, {'broccoli cooked, Vitamin D: 181.7'}, {'kohlrabi raw, Vitamin D: 164.3'}]`
`high_med_low_filter('Vitamin D', high=False, med=False, low=True, data=db)`	`[{'cream cheese, Vitamin D: 0.0'}, {'neufchatel cheese, Vitamin D: 0.0'},`
`high_med_low_filter('Vitamin D', high=False, med=True, low=False, data='DataBase/Food_Nutrition_Dataset.csv')`	`[{'orange, Vitamin D: 97.9'}, {'guava, Vitamin D: 125.6'}, ...`

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

```
def test_high_med_low_filter_high():

    expect_res = [{'pokeberry shoots raw, Vitamin D: 217.6'}, {'tomato juice, Vitamin D: 170.3'},
                  {'broccoli cooked, Vitamin D: 181.7'}, {'kohlrabi raw, Vitamin D: 164.3'}]
```

```
  func_res = high_med_low_filter('Vitamin D', high=True, med=False, low=False, data=db)

  assert func_res == expect_res


def test_high_med_low_filter_low():

  expect_res = [{'cream cheese, Vitamin D: 0.0'}, {'neufchatel cheese, Vitamin D: 0.0'}, {'requeijao
  func_res =high_med_low_filter('Vitamin D', high=False, med=False, low=True, data=db)

  assert func_res == expect_res


def test_high_med_low_filter_med():

  expect_res = [{'orange, Vitamin D: 97.9'}, {'guava, Vitamin D: 125.6'}, {'mango, Vitamin D: 122.3'}

  func_res = high_med_low_filter('Vitamin D', high=False, med=True, low=False, data='DataBase/Food_Nu

  assert func_res == expect_res
```

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

Invalid Input	Expected Output
`high_med_low_filter('Vitamin Q', high=False, med=True, low=False, data='DataBase/Food_Nutrition_Dataset.csv')`	`Error: Nutrient component not found in database`
`high_med_low_filter('Vitamin D', high=False, med=True, low=False, data='DataBase/Food_Nutrition_Datasets.csv')`	`Error: DB not found.`

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

```
def test_high_med_low_invalid_nurti():

  expect_res = print("Error: Nutrient component not found in database")

  func_res = high_med_low_filter('Vitamin Q', high=False, med=True, low=False, data='DataBase/Food_Nu

  assert func_res == expect_res

def test_high_med_low_db_error():

  expect_res = print("Error: DB not found.")

  func_res = high_med_low_filter('Vitamin D', high=False, med=True, low=False, data='DataBase/Food_Nu

  assert func_res == expect_res
```

Test Case 5:

- **Test Function/Module**
  - `test_high_low_filter_db_error()`
  - `test_high_low_filter_low()`
  - `test_high_low_filter_high()`
  - `test_high_low_filter()`
  -
- **Tested Function/Module**
  - `high_low_filter('Vitamin D', high = False, low = True, data = 'DataBase/Food_Nutrition_Dataset.csv')`
- **Description**
  - High low filter to retrieve the highest and lowest values from the database.
- **1) Valid Input and Expected Output**

Valid Input	Expected Output
`high_low_filter('Vitamin D', high = False, low = True, data = 'DataBase/Food_Nutrition_Dataset.csv')`	`[{'cream cheese, Vitamin D: 0.0'}, {'fruit flavored water, Vitamin D: 0.0'}, {'table water, Vitamin D: 0.0'}, ...`
`test_high_low_filter_high()`	`[{'pokeberry shoots raw, Vitamin D: 217.6'}, {'broccoli cooked, Vitamin D: 181.7'}, {'tomato juice, Vitamin D: 170.3'}, ... }]`
`add more cases in necessary`	`...`

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

```
def test_high_low_filter_high():
  func_res = high_low_filter('Vitamin D', high = True, low = False, data = 'DataBase/Food_Nutrition_Da
  expect_res = [{'pokeberry shoots raw, Vitamin D: 217.6'}, {'broccoli cooked, Vitamin D: 181.7'}, {'
  assert func_res == expect_res
def test_high_low_filter_low():
  func_res = high_low_filter('Vitamin D', high = False, low = True, data = 'DataBase/Food_Nutrition_Da
  expect_res = [{'cream cheese, Vitamin D: 0.0'}, {'fruit flavored water, Vitamin D: 0.0'}, {'table wa
  assert func_res == expect_res
```

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

Invalid Input	Expected Output
`high_low_filter('Vitamin P', high = True, low = False, data = 'DataBase/Food_Nutrition_Dataset.csv')`	`"Error: Nutrient component not found in database"`
`high_low_filter('Vitamin D', high = False, low = True, data = 'DataBase/Food_Nutrition_Datasets.csv)`	`"Error: DB not found."`

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

```
def test_high_low_filter():
  func_res = high_low_filter('Vitamin P', high = True, low = False, data = 'DataBase/Food_Nutrition_Da
  expect_res = print("Error: Nutrient component not found in database")
  assert func_res == expect_res
def test_high_low_filter_db_error():
  expect_res = print("Error: DB not found.")
  func_res = high_low_filter('Vitamin D', high = False, low = True, data = 'DataBase/Food_Nutrition_Da
  assert func_res == expect_res
```

3. Testing Report Summary

unit_test.html

Report generated on 06-Oct-2024 at 15:46:07 by pytest-html v3.1.1

Environment

Packages	{"pluggy": "1.0.0", "py": "1.11.0", "pytest": "7.1.2"}
Platform	Darwin-23.6.0-x86_64-i386-64bit
Plugins	{"cov": "4.1.0", "html": "3.1.1", "metadata": "1.11.0", "mock": "3.10.0"}
Python	3.7.16

Summary

22 tests ran in 5.97 seconds.

(Un)check the boxes to filter the results.

☑ 22 passed, ☑ 0 skipped, ☑ 0 failed, ☑ 0 errors, ☑ 0 expected failures, ☑ 0 unexpected passes

Results

Show all details / Hide all details

Result	Test	Duration	Links
Passed *(show details)*	test_all_functions.py::test_search_food_keyword_found	0.00	
Passed *(show details)*	test_all_functions.py::test_search_food_keyword_partial_match	0.00	
Passed *(show details)*	test_all_functions.py::test_search_food_keyword_not_found	0.00	
Passed *(show details)*	test_all_functions.py::test_search_food_invalid_db	0.00	
Passed *(show details)*	test_all_functions.py::test_nutrition_breakdown_valid_food	3.81	
Passed *(show details)*	test_all_functions.py::test_nutrition_breakdown_invalid_food	0.01	
Passed *(show details)*	test_all_functions.py::test_nutrition_breakdown_invalid_db	0.00	
Passed *(show details)*	test_all_functions.py::test_range_filter	0.01	
Passed *(show details)*	test_all_functions.py::test_range_filter_no_res	0.00	
Passed *(show details)*	test_all_functions.py::test_range_filter_invalid_name	0.00	
Passed *(show details)*	test_all_functions.py::test_range_filter_invalid_value	0.00	
Passed *(show details)*	test_all_functions.py::test_range_filter_invalid_value2	0.00	
Passed *(show details)*	test_all_functions.py::test_range_filter_no_db	0.00	
Passed *(show details)*	test_all_functions.py::test_high_med_low_filter_high	0.00	
Passed *(show details)*	test_all_functions.py::test_high_med_low_filter_low	0.03	
Passed *(show details)*	test_all_functions.py::test_high_med_low_filter_med	0.00	
Passed *(show details)*	test_all_functions.py::test_high_med_low_invalid_nurti	0.00	
Passed *(show details)*	test_all_functions.py::test_high_med_low_db_error	0.00	
Passed *(show details)*	test_all_functions.py::test_high_low_filter_db_error	0.00	
Passed *(show details)*	test_all_functions.py::test_high_low_filter_low	0.01	
Passed *(show details)*	test_all_functions.py::test_high_low_filter_high	0.00	
Passed *(show details)*	test_all_functions.py::test_high_low_filter	0.00	