System Requirements Specification Index

For

- Python Basics and NumPy, Pandas
- Usecase No 9
- 1.0

IIHT Pvt. Ltd.
fullstack@iiht.com

Use Case No 1: Pharmacy Inventory Management System (PharmacyManagementSystem .py)

Dataset

```
medicines = {
   101: {'name': 'Paracetamol', 'category': 'Painkiller', 'price': 10, 'stock': 100},
   102: {'name': 'Amoxicillin', 'category': 'Antibiotic', 'price': 50, 'stock': 25},
   103: {'name': 'Cetirizine', 'category': 'Antihistamine', 'price': 15, 'stock': 60},
   104: {'name': 'Ibuprofen', 'category': 'Painkiller', 'price': 20, 'stock': 80},
   105: {'name': 'Metformin', 'category': 'Antidiabetic', 'price': 30, 'stock': 40}
}
```

1Write a Python function to count the number of unique medicines available.
Define: count_number_of_unique_medicines()
The function should:

- Extract all unique medicine names from the dataset.
- Count and return the total number of unique medicines.

2Write a Python function to calculate the total stock value of all medicines.
Define: calculate_total_stock_value()
The function should:

- Compute the total stock value using Price * Stock Quantity for each medicine.
- Return the overall stock value.

3Write a Python function to find the medicine with the lowest stock quantity.
Define: find_medicine_with_lowest_stock()
The function should:

- Identify the medicine with the least stock availability.
- Return the medicine name along with the remaining stock quantity.
- Display all the values in the main function

Use CaseNo 2: Courier Tracking System (CourierTrackingSystem.py)

Dataset

```
shipments = {
   "TRK1001": {
      "sender": "Alice Johnson",
      "recipient": "Bob Smith",
      "current_location": "New York",
      "status": "In Transit"
   },
   "TRK1002": {
      "sender": "Charlie Brown",
      "recipient": "David Lee",
      "current_location": "Chicago",
      "status": "Delivered"
   },
   "TRK1003": {
      "sender": "Eve Miller",
      "recipient": "Frank White",
      "current_location": "San Francisco",
      "status": "Out for Delivery"
   },
```

```
    "TRK1004": {
        "sender": "George Harris",
        "recipient": "Helen Young",
        "current_location": "Los Angeles",
        "status": "In Transit"
    },
    "TRK1005": {
        "sender": "Ian Scott",
        "recipient": "Jane Clark",
        "current_location": "Boston",
        "status": "Delivered"
    }
}
```

1.Write a Python function to list all shipments with details.

Define: list_all_shipments()

The function should:

- Retrieve and return a list of all shipments.
- Include tracking number, sender, recipient, location, and status.

2 Define: check_shipment_status(tracking_number)

The function should:

- Tracking ID (TRK1003)
- Return the current status of (TRK1003) the shipment.
- If tracking number not found, return an appropriate message.

Use Case: Donation Management System (**donationmanagementsystem.py**)
Dataset

```
donations = {
    1: {'donor': 'Alice Johnson', 'amount': 100, 'date': '2025-02-25', 'purpose': 'Charity'},
    2: {'donor': 'Bob Smith', 'amount': 50, 'date': '2025-02-26', 'purpose': 'Education'},
    3: {'donor': 'Charlie Brown', 'amount': 200, 'date': '2025-02-25', 'purpose': 'Medical'},
    4: {'donor': 'David Lee', 'amount': 150, 'date': '2025-02-26', 'purpose': 'Charity'},
    5: {'donor': 'Frank White', 'amount': 120, 'date': '2025-02-27', 'purpose': 'Medical'}
}
```

1Write a Python function to write the file of donation made using the dataset
Define: save_static_donations():

- Write the program to save the dataset to the file

2Write a Python function to calculate the total amount donated by **frankwhite**

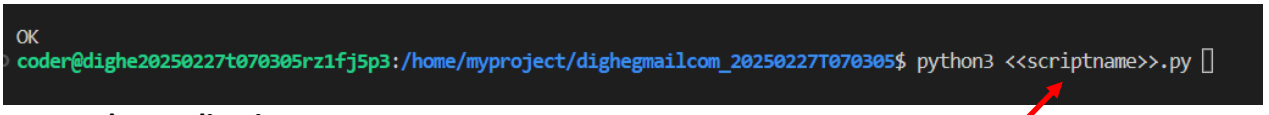
- Create a function to calculate donation made by **frankwhite**
- Calculate total donation made using the function calculate_total_donations():
- if '**frankwhite**' has made any donations. The function should perform a **case-insensitive check** on the donor's name

Execution Steps to Follow:

- All actions like build, compile, running application, running test cases will be through Command Terminal.
- To open the command terminal the test takers, need to go to Application menu (Three horizontal lines at left top) -> Terminal -> New Terminal
- This editor Auto Saves the code
- If you want to exit(logout) and continue the coding later anytime (using Save & Exit option on Assessment Landing Page) then you need to use **CTRL+Shift+B** -command compulsorily on code IDE. This will push or save the updated contents in the internal git/repository. Else the code will not be available in the next login.
- These are time bound assessments the timer would stop if you logout and while logging in back using the same credentials the timer would resume from the same time it was stopped from the previous logout.
- To setup environment:
  You can run the application without importing any packages
- To launch application:
  python3 CourierTrackingSystem.py
  python3 PharmacyManagementSyste.py
  python3 donationmanagementsystem.py
- To run Test cases:
  python3 -m unittest

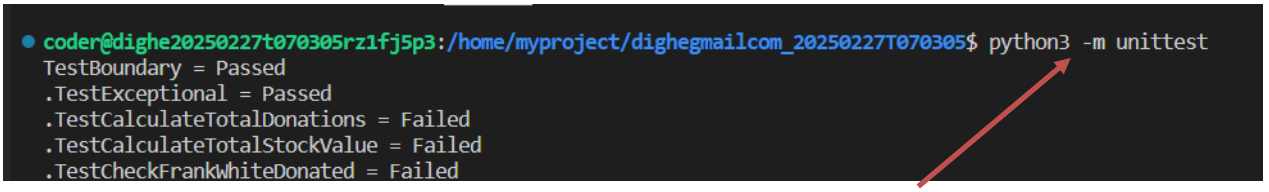
- Before Final Submission also, you need to use **CTRL+Shift+B** - command compulsorily on code IDE, before final submission as well. This will push or save the updated contents in the internal git/repository, and will be used to evaluate the code quality.

Screen shot to run the program



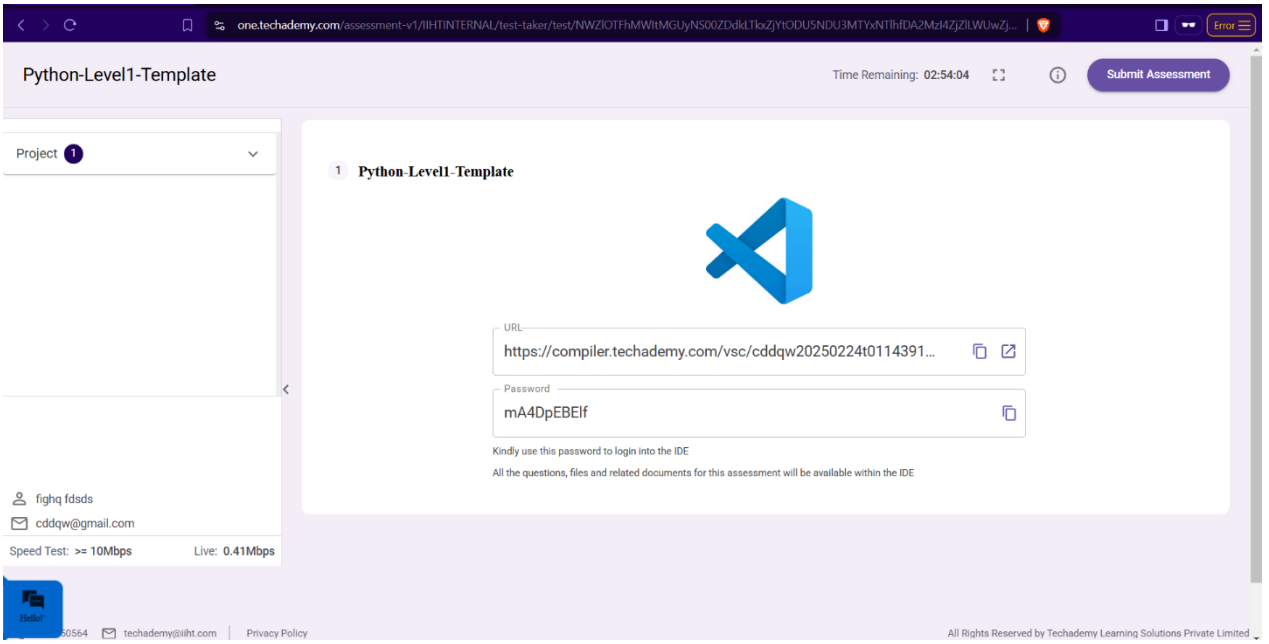
To run the application

python3 CourierTrackingSystem.py
python3 PharmacyManagementSystem.py
python3 donationmanagementsystem.py



To run the testcase

python3 -m unittest



- **Once you are done with development and ready with submission, you may navigate to the previous tab and submit the workspace. It is mandatory to click on “Submit Assessment” after you are done with code.**