System Requirements Specification Index

For

Python Basics and NumPy, Pandas

Usecase No 4

1.0

IIHT Pvt. Ltd.
fullstack@iiht.com

Use Case: Blood Bank Management (BloodBankManagementSystem.py)

The dataset to be used in the format is

```
blood_groups=np.array(["A+", "A-", "B+", "B-", "O+", "O-", "AB+", "AB-"])
units_available=np.array ([10, 5, 8, 4, 15, 7, 6, 3])
```

1. **Write a Python program to manage blood bank inventory.**
   - Define a function **get_blood_bank_data()**.

2. **Write a Python program to add a new blood group to the inventory.**
   - Define a function **add_new_blood_group(blood_bank, blood_group, units)**.
   - The function should:
     - Accept a **DataFrame** (blood bank inventory), a **new blood group**, and its **unit count**.
     - Append the new blood group entry to the DataFrame.
     - Return the updated inventory.
   - Add a **new blood group (P+ with 10 units)** using **add_new_blood_group()**.

3. **Write a Python program to calculate the total number of blood units available.**
   - Define a function **get_total_units(blood_bank)**.
   - The function should:
     - Compute and return the **sum of all blood units** in the inventory.

Use Case: Food Delivery Management (OnlineFoodDeliverySystem.py)

In the source template file the value are updated in a wrong values user need to update the format to pass the testcase use the value in the document

1. **Write a Python program to manage a food menu and prices.**

Dataset provides are {
```
"Pizza": 8.50,
"Burger": 5.00,
"Pasta": 7.25,
"Fries": 3.50,
"Cola": 2.00
}
```
   - Define a function get_menu().
   - The function should:
     - Return a **dictionary** containing food items as **keys** and their prices as **values**.

2. **Write a Python program to calculate the total bill for a food order.**

   Create a function named calculate_bill() that takes two parameters:

   - orders: A list of tuples, where each tuple contains an item name and quantity
   - menu: A dictionary with menu items and their prices

   Initialize variables:

- total_bill: To track the total bill amount (start at 0)
- order_summary: An empty list to store order summary strings

Loop through each order (item, quantity):

- Check if the item exists in the menu
- If it does, calculate the item's total price (item_price * quantity)
- Add this amount to the total bill
- Add a formatted string to the order_summary list: "{item} x{quantity} = ${item_total:.2f}"

Return a tuple containing the total bill and order summary list

3. **Write a Python program to save order details to a file.**
   - Define a function **save_order(order_summary, total_bill, filename="food_orders.txt")**.
   - The function should:
     - Write the **order summary** and **total bill** to a text file.
     - Append new orders without overwriting existing data.
     - Return the filename after successful save.

Use Case 3: Employee Leave Management (EmployeeLeaveManagementSystem.py)

In the source template file the value are updated in a wrong format, wrong value user need to update the format to pass the testcase use only documentation values

Dataset to be used should be {

"E001": {"name": "John Doe", "leave_balance": 12},
"E002": {"name": "Alice Smith", "leave_balance": 10},
"E003": {"name": "Bob Johnson", "leave_balance": 8},
"E004": {"name": "Emma Davis", "leave_balance": 15},
"E005": {"name": "Michael Brown", "leave_balance": 5},

Leave request { ("E001", 3), -> leave request count
("E003", 2),
("E005", 4),
("E002", 1),
("E004", 5),
("E999", 3),
("E003", -2) }

1. **Write a Python program to manage employee leave balances (Dictionary format)**
   - Define a function **get_employee_data()**.
   - The function should:
     - Return a **dictionary** containing employee IDs as **keys**.
     - Each employee ID maps to a dictionary with **name** and **leave balance**.

2. **Write a Python program to process leave requests.(list format)**
   - Define a function **process_leave_requests(employees, leave_requests)**.
   - The function should:
     - Accept a **dictionary** of employees and a **list of leave requests** (employee

ID, leave days).

  - Validate if the **employee ID exists**.
  - Reject **negative leave requests**.
  - Approve leave if the **balance is sufficient** and update the leave balance.
  - Deny leave if the **balance is insufficient**.
  - Return a **list of messages** summarizing the leave request outcomes.

3. **Process and display the leave request summary.(list format)**
   - Display the **summary of leave approvals and rejections**.

Execution Steps to Follow:

1. All actions like build, compile, running application, running test cases will be through Command Terminal.

2. To open the command terminal the test takers, need to go to Application menu(Three horizontal lines at left top) -> Terminal -> New Terminal

3. This editor Auto Saves the code

4. 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.

5. 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 sametime it was stopped from the previous logout.

6. To setup environment:

You can run the application without importing any packages

7. To launch application:

python3 BloodBankManagementSystem.py

python3 OnlineFoodDeliverySystem.py

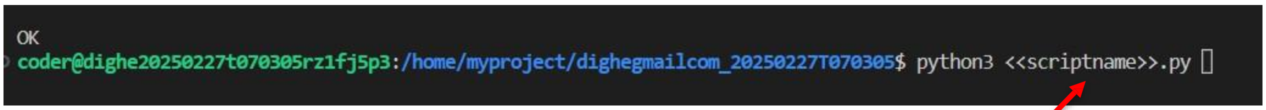
python3 EmployeeLeaveManagementSystem.py

To run Test cases:

python3 -m unittest

8. 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



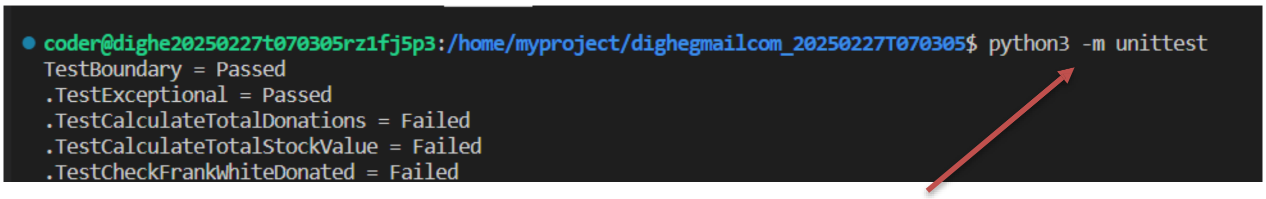
```
OK
coder@dighe20250227t070305rz1fj5p3:/home/myproject/dighegmailcom_20250227T070305$ python3 <<scriptname>>.py
```

To run the application

python3 BloodBankManagementSystem.py

python3 OnlineFoodDeliverySystem.py

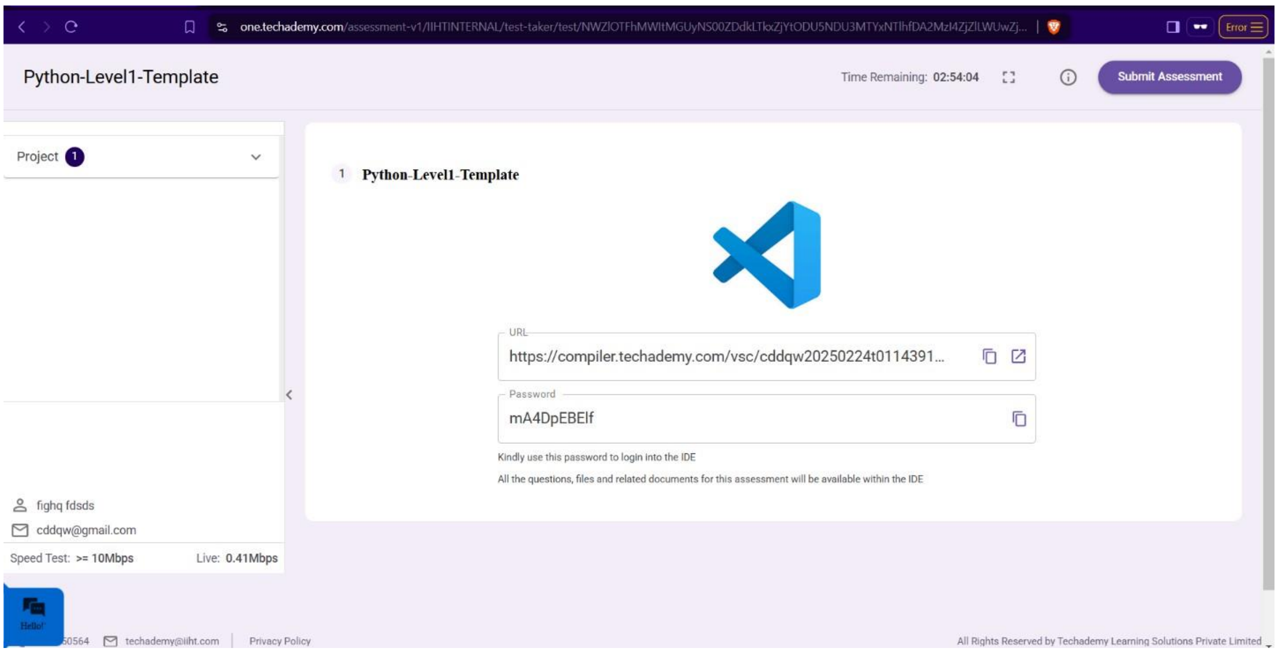
python3 EmployeeLeaveManagementSystem.py



```
● coder@dighe20250227t070305rz1fj5p3:/home/myproject/dighegmailcom_20250227T070305$ python3 -m unittest
  TestBoundary = Passed
  .TestExceptional = Passed
  .TestCalculateTotalDonations = Failed
  .TestCalculateTotalStockValue = Failed
  .TestCheckFrankWhiteDonated = Failed
```

To run the testcase

- **python3 -m unittest**



one.techademy.com/assessment-v1/IIHTINTERNAL/test-taker/test/NWZlOTFhMWItMGUyNS00ZDdkLTkxZjYtODU5NDU3MTYxNTlhfDA2Mzl4ZjZlLWUwZj...

Python-Level1-Template

Time Remaining: 02:54:04

Submit Assessment

Project 1

1 Python-Level1-Template

URL
https://compiler.techademy.com/vsc/cddqw20250224t0114391...

Password
mA4DpEBElf

Kindly use this password to login into the IDE

All the questions, files and related documents for this assessment will be available within the IDE

fighq fdsds
cddqw@gmail.com
Speed Test: >= 10Mbps Live: 0.41Mbps

60564 techademy@iiht.com | Privacy Policy

All Rights Reserved by Techademy Learning Solutions Private Limited

9. 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.

-----x-----