System Requirements Specification Index

For

Python Basics and NumPy, Pandas
UseCase No 7
1.0

IIHT Pvt. Ltd.
fullstack@iiht.com

Task 1

Hotel Management (`hotel_management.py`)

Function 1: `filter_hotels_by_rating(df, min_rating)`

Dataset:
[Hotel ID, Hotel Name, Rating, Price Per Night]

```
python
CopyEdit
hotel_data = pd.DataFrame({
    "Hotel ID": [1, 2, 3, 4, 5],
    "Hotel Name": ["Grand Stay", "Comfort Inn", "Luxury Palace", "Budget
Lodge", "Ocean View"],
    "Rating": [4.5, 3.8, 4.9, 3.2, 4.3],
    "Price Per Night": [2500, 1800, 4000, 1200, 3200]
})
```

Purpose:
Return all hotels whose rating is greater than or equal to the specified minimum.

Parameters:
• `df (pd.DataFrame)`: DataFrame of hotel records
• `min_rating (float)`: Minimum rating threshold

Returns:
• `pd.DataFrame`: Filtered DataFrame of matching hotels

Instructions:

1. Use a condition to filter `Rating` column.
2. Return only matching rows.

Function 2: `get_average_price(df)`

Purpose:
Calculate the average price per night across all hotels.

Parameters:
• `df (pd.DataFrame)`: DataFrame of hotel records

Returns:
• `float`: Rounded average of "Price Per Night" column

Instructions:

1. Access the `"Price Per Night"` column.
2. Use `.mean()` to calculate average.
3. Use `round(..., 2)` and return the result.

Task 2

Gym Membership Management (`membership_management.py`)

Function 1: `calculate_total_contribution(df, member_id)`

Dataset:
[Member ID, Name, Membership Type, Monthly Fee, Months Active]

```
python
CopyEdit
membership_data = pd.DataFrame({
    "Member ID": [101, 102, 103, 104, 105],
    "Name": ["Alice", "Bob", "Charlie", "Diana", "Ethan"],
    "Membership Type": ["Gold", "Silver", "Gold", "Bronze", "Silver"],
    "Monthly Fee": [1500, 1000, 1500, 800, 1000],
    "Months Active": [12, 8, 6, 3, 10]
})
```

Purpose:
Calculate the total revenue from a single member.

Parameters:
• `df (pd.DataFrame)`: Gym member dataset
• `member_id (int)`: Member ID to look up

Returns:
• `int`: Monthly Fee × Months Active

Instructions:

1. Filter the row with matching `member_id`.
2. Multiply `Monthly Fee` by `Months Active`.
3. Return the result as `int`.

Function 2: `count_members_per_type(df)`

Purpose:
Count the number of members under each membership type using a loop.

Parameters:
• `df (pd.DataFrame)`: Gym member dataset

Returns:
• `dict`: Dictionary of membership type → count

Instructions:

1. Initialize empty dictionary.
2. Loop through `"Membership Type"` column.
3. Use condition to increment counters.

Function 3: `is_alice_long_term(df)`

Purpose:
Check if Alice has more than 12 months of activity.

Parameters:
• `df (pd.DataFrame)`: Gym member dataset

Returns:
• `bool`: `True` if Alice has more than 12 months, else `False`

Instructions:

1. Filter the row where name is "Alice".
2. Compare `"Months Active"` with 12.
3. Return boolean result.

Task 3

Event Management (`event_management.py`)

Function 1: `get_total_people_registered(df)`

Dataset:
[Event ID, Event Name, Location, Registered People, Entry Fee]

```
python
CopyEdit
event_data = pd.DataFrame({
    "Event ID": [201, 202, 203, 204, 205],
    "Event Name": ["Tech Summit", "Music Fest", "Job Fair", "Food Carnival",
"Book Launch"],
    "Location": ["New York", "Los Angeles", "Chicago", "New York", "Boston"],
    "Registered People": [120, 200, 75, 180, 50],
    "Entry Fee": [50, 30, 0, 20, 10]
})
```

Purpose:
Compute the total number of people registered across all events.

Parameters:
• `df (pd.DataFrame)`: Event dataset

Returns:
• `int`: Total registered people

Instructions:

1. Use `.sum()` on `"Registered People"` column.
2. Return the result.

Function 2: `count_events_with_high_fees(df)`

Purpose:
Count how many events have entry fees greater than 10.

Parameters:
• `df (pd.DataFrame)`: Event dataset

Returns:
• `int`: Number of such events

Instructions:

1. Use condition on `"Entry Fee"` > 10.
2. Use `.shape[0]` to count matching rows.

Function 3: `count_events_in_newyork(df)`

Purpose:
Count how many events are happening in New York.

Parameters:
• `df (pd.DataFrame)`: Event dataset

Returns:
• `int`: Number of events in New York

Instructions:

1. Use condition on `"Location"` == `"New York"` (case-insensitive).
2. Use `.shape[0]` to get count.

Execution Steps to Follow:

- All actions like build, compile, running application, running test cases will be through the 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)
- 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 hotel.py
  python3 Event.py
  python3 Gym.py
- To run Test cases:
  python3 -m unittest

Screen shot to run the program

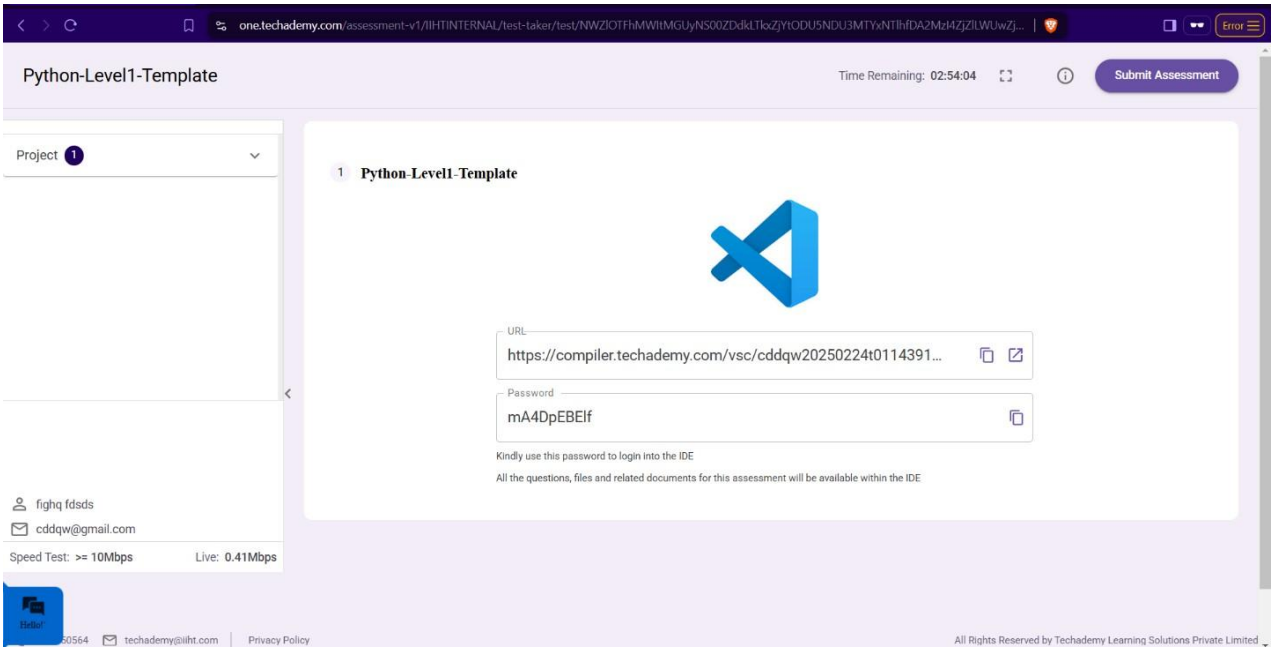
To run the application

python3 hotel.py
python3 Event.py
python3 Gym.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



- 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.