System RequirementsSpecification Index

For

Python Basics and NumPy, Pandas
Usecase No 8
1.0

IIHT Pvt. Ltd.

fullstack@iiht.com

Usecase No 1 Use Case: Student Attendance Management System (student.py)

Dataset

```
students = {
   101: {'name': 'Alice Johnson', 'class': '10th Grade'},
   102: {'name': 'Bob Smith', 'class': '10th Grade'},
   103: {'name': 'Charlie Brown', 'class': '10th Grade'},
   104: {'name': 'David Lee', 'class': '10th Grade'},
   105: {'name': 'Eve Miller', 'class': '10th Grade'}
}
```

Attendance Records (student_id: list of (date, status) tuples):

python

CopyEdit

```
attendance = {
   101: [('2025-02-25', 'Absent'), ('2025-02-26', 'Absent'), ('2025-02-27', 'Absent')],
   102: [('2025-02-25', 'Absent'), ('2025-02-26', 'Present'), ('2025-02-27', 'Present')],
   103: [('2025-02-25', 'Present'), ('2025-02-26', 'Absent'), ('2025-02-27', 'Present')],
   104: [('2025-02-25', 'Present'), ('2025-02-26', 'Present'), ('2025-02-27', 'Present')],
   105: [('2025-02-25', 'Absent'), ('2025-02-26', 'Present'), ('2025-02-27', 'Absent')]
}
```

1 Write a Python function to count the number of unique attendance days.
Define: count_number_of_days_using_dates()
The function should:

- Identify the unique dates in the attendance dataset.
- Print the total number of unique dates.

2Write a Python function to find the student with the most absent days.
Define: find_most_absent_student()
The function should:

- Count the number of absences for each student.
- Identify and print the student with the highest number of absent days.

Use Case No 2: Flight Management System (FlightReservationSystem.py)
Dataset

```
flights_data = {
   "Flight Number": ["AI101", "BA202", "DL303"],
   "Airline": ["Air India", "British Airways", "Delta Airlines"],
   "Total Seats": [150, 180, 200],
   "Booked Seats": [120, 160, 190],
   "Ticket Price": [8000, 12000, 10000]  # Price per ticket in ₹
  }
```

1.Write a Python function to list all flights with details.
Define: list_all_flights()
The function should:

- Display all flight details in a tabular format using Pandas DataFrame.
- Print the list of available flights.

2Write a Python function to check available seats for a given flight.
Define: available_seats_for_flight(flight_number)
The function should:

- Retrieve total and booked seats for a specific flight.
- Calculate the number of available seats using NumPy.
- Return the available seat count or an appropriate message if the flight is not found.

3Write a Python function to calculate the total revenue generated from all flights.
Define: total_revenue_for_all_flights()
The function should:

- Calculate the revenue for each flight using Booked Seats * Ticket Price.
- Compute the total revenue using Pandas operations.
- Return the total revenue value

Use Case No 3: Movie Ticket Booking System (MovieTicketBookingSystem.py)
Dataset

```
movies = {
   "Avengers: Endgame": {
      "total_seats": 10,
      "booked_seats": 7,
      "ticket_price": 250  # Price per ticket in ₹
   },
   "Inception": {
      "total_seats": 8,
      "booked_seats": 4,
      "ticket_price": 200
   },
   "The Dark Knight": {
      "total_seats": 12,
      "booked_seats": 11,
      "ticket_price": 300
   }
}
```

1.Write a Python function to calculate the total number of tickets sold for all movies.
Define: total_tickets_sold()
The function should:

- Sum up all the booked seats across different movies.
- Return the total count of sold tickets.

2.Write a Python function to find the movie that has generated the highest revenue.
Define: highest_revenue_movie()
The function should:

- Compute the total revenue for each movie using Booked Seats * Ticket Price.
- Identify and return the movie with the highest revenue.

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 whilelogging in back using the same credentials the timer would resume from the sametime it was stopped from the previous logout.
  To setup environment:
- You can run the application without importing any packages
  To launch application:
  python3 student.py
  python3 MovieTicketBookingSystem.py
  python3 FlightReservationSystem.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

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

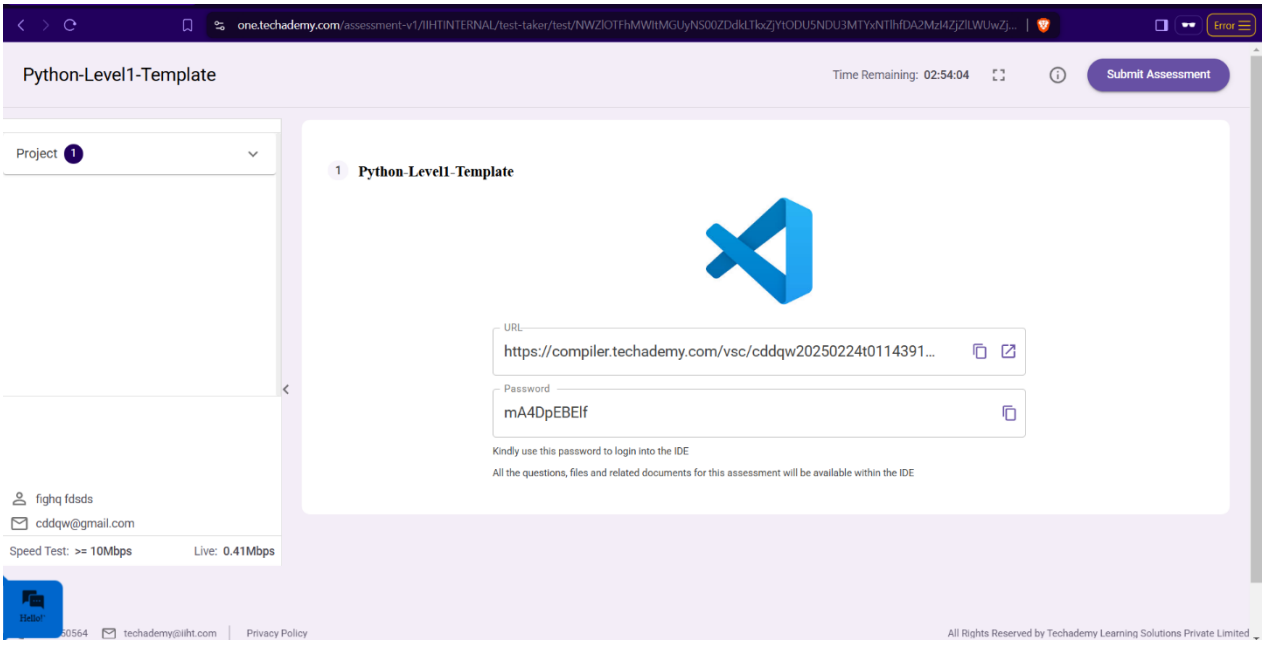
To run the application

python3 student.py
python3 MovieTicketBookingSystem.py
python3 FlightReservationSystem.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.**