System Requirements Specification Index

For

Python Basics and NumPy, Pandas

Restaurant ordering system 1.0

IIHT Pvt. Ltd.
fullstack@iiht.com

Use case on Python basics and NumPy and Pandas

Introduction

Bites Restaurant

Welcome to **Gourmet Hub** a newly opened digital restaurant! You have been hired as a software engineer to develop an efficient system for managing restaurant orders, analyzing sales, and calculating daily revenue statistics.

The restaurant serves various delicious meals, and your task is to create a **Restaurant Manager** system

You must create the following code with the condition make should do don't create the any new def function

Note :

- The test takers are asked strictly follow these names and menu items and values in the code .
- In the template code complete the ToDo list .

NO	CUSTOMER NAMES	MENU ITEMS	Values of the items
1	ALICE	Pizza	$12.99
2	BOB	Salad	$7.99
3	CHARLIE	Pasta	$10.99
4		Chicken Fried Rice	$13.99
5		Burger	$9.99

Project understanding

1. Menu Management

- The menu is stored as a dictionary with food items and their prices.
- The system supports **adding** items including tracking **non-vegetarian** dishes.

2. Order Processing

- Customers can place orders by selecting items from the menu.
- The system calculates the total cost and stores the orders in a list.
- Orders can be **viewed** at any time.

3. File I/O Operations

- Orders can be **saved** to a file (orders.txt) for persistence.
- Orders can be **loaded** from a file to restore data after a restart.

4. Sales Data Analysis (Using NumPy & Pandas)

- Generates an **order summary** using Pandas.
- Uses NumPy to calculate:
    - **Total revenue**
    - **Average order value**
    - **Minimum and maximum order values**

5. Customer & Item Insights

- Identifies the **most popular** menu item based on orders.
- Determines the **top-spending customer** based on total purchases.

Section 1: Python Basics - Data Types, Operators, Control Flow

Add the menu items in the list

```
{
   "Pizza": 12.99,
   "Burger": 9.99,
   "Pasta": 10.99,
   "Salad": 7.99,
   "Chicken Fried Rice": 13.99
}
```

Add the Dictionary to track non-veg items

Section 2: Functions, Modules, Packages

display_menu()

- Prints the restaurant's menu with item names and their prices.

take_order ()

- Takes a customer's name and a list of ordered items
- manager.take_order("Alice", ["Pizza", "Salad"])
- manager.take_order("Bob", ["Burger", "Pasta", "Chicken Fried Rice"])
- manager.take_order("Charlie", ["Pizza", "Chicken Fried Rice", "Burger"])

view_orders()

- Customer's name.
- List of ordered items.
- Total order cost.

- Use these customer name with the combination of the menu

- The name of the customers are Alice , bob , Charlie

available_items ()

- return available items

Section 3: Data Structures - Lists, Tuples, Dictionaries, Sets

- Add to menu ()

  To Check the items already present

- get_non_veg_items()

  get the non veg dish from the menu

Section 4: File I/O, Exception Handling

- save_orders_to_file ()

  Create a file order.txt and store the orders in the files

  Then display the orders from the files

  Section 5: Python for Data Science - NumPy, Pandas

  analyze_orders ()

- Calculate total_revenue ()
- **Calculate avg_order_value ()**
- **Calculate min_order_value ()**
- **Calculate max_order_value()**
- **Calculate get_most_popular_item ()**
- **Calculate get_top_spending_customer ()**

get_most_popular_item ()

- get the most popular item ordered in the menu .

- get_top_spending_customer

Question to solve

- The user need to count current order details
- Check total number of orders under taken
- Check if the orders are saved in orders.txt
- Which item is the popular item in the menu ?
- Which item is non veg item in the menu ?
- Show the total summary of the of the order
- Find out who is the top customer spending in the restaurant

Project structure

```
restaurant_management_system/
|— 🗁 src/                          # Source code
|   |— restaurant_manager.py       # Main restaurant manager script
|   |— # Demo script
|   |— orders.txt                  # data storage
|— 🗁 tests/                        # Test cases
|— 🗁 docs/                         # Documentation
|   |— Restaurant_document.docx
|— custom.ih
```

In the above project structure, you are given the template to complete the restaurant manager.py , template code .

Execution Steps to Follow:

1. All actions like build, compile, running application,running test cases will bethrough 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 & Exitoption 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:

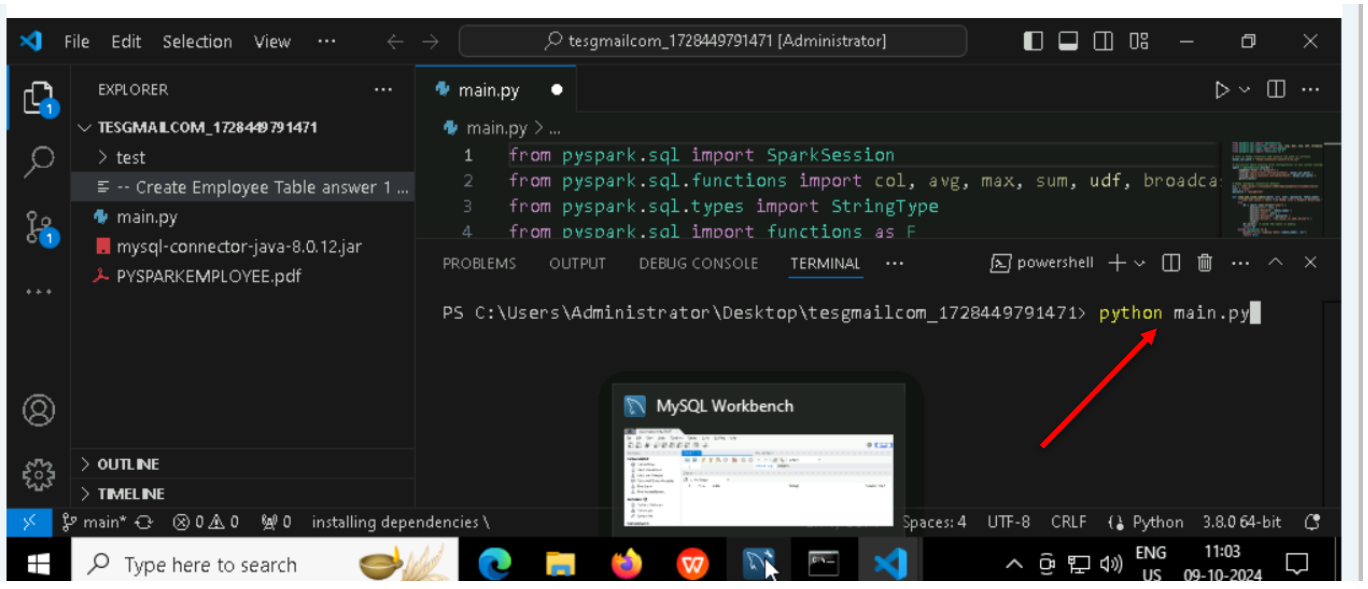
Python restaurant_manager.py

8. To run Test cases:

Python3 -m unittest

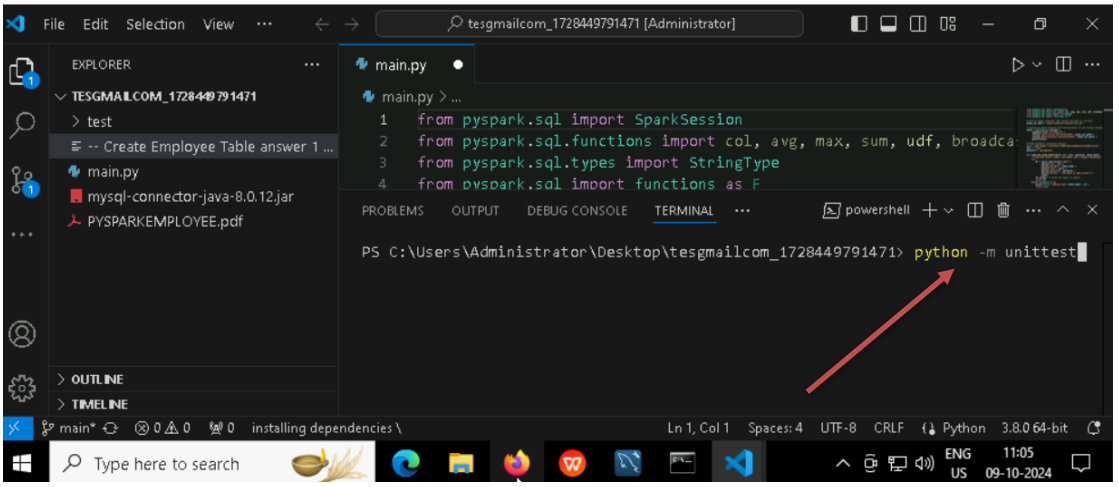
Before Final Submission also, you need to use CTRL+Shift+B-command compulsorily on code IDE. This will push or save the updated contents in the internalgit/repository for code

Screen shot to run the program



To run the application

- **Python restaurant_manager.py**



To run the testcase

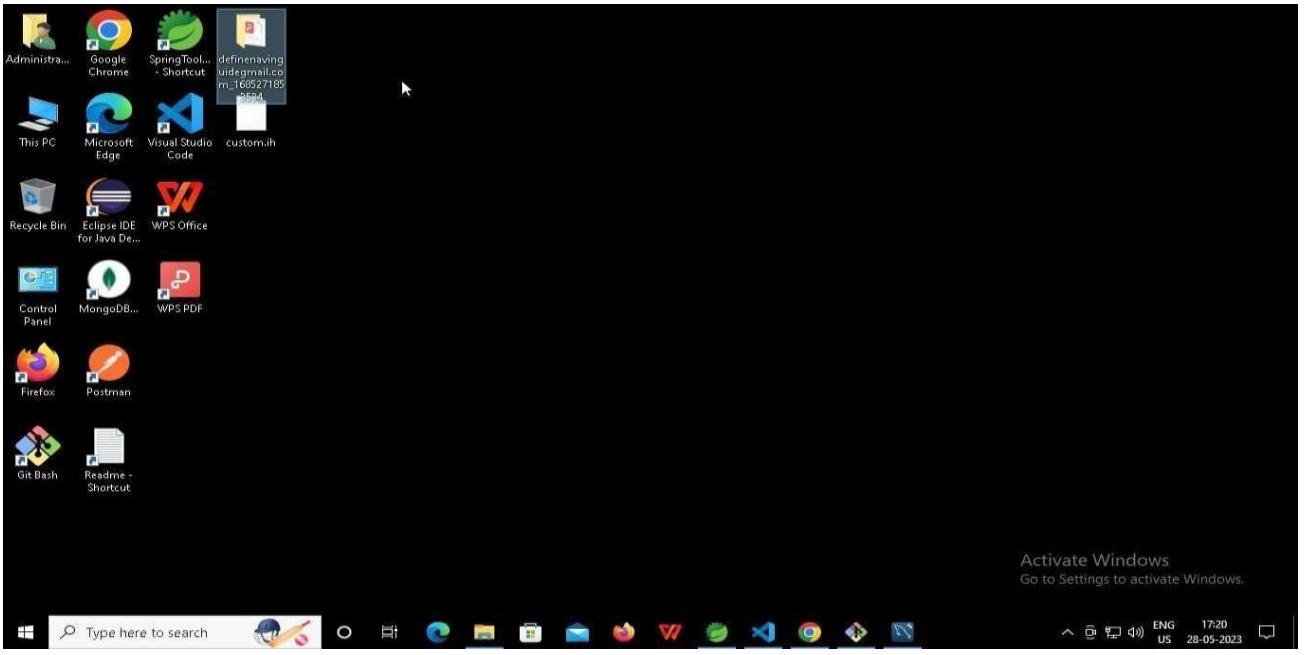
- **Python3 -m unittest**

Screenshot to push the application to github

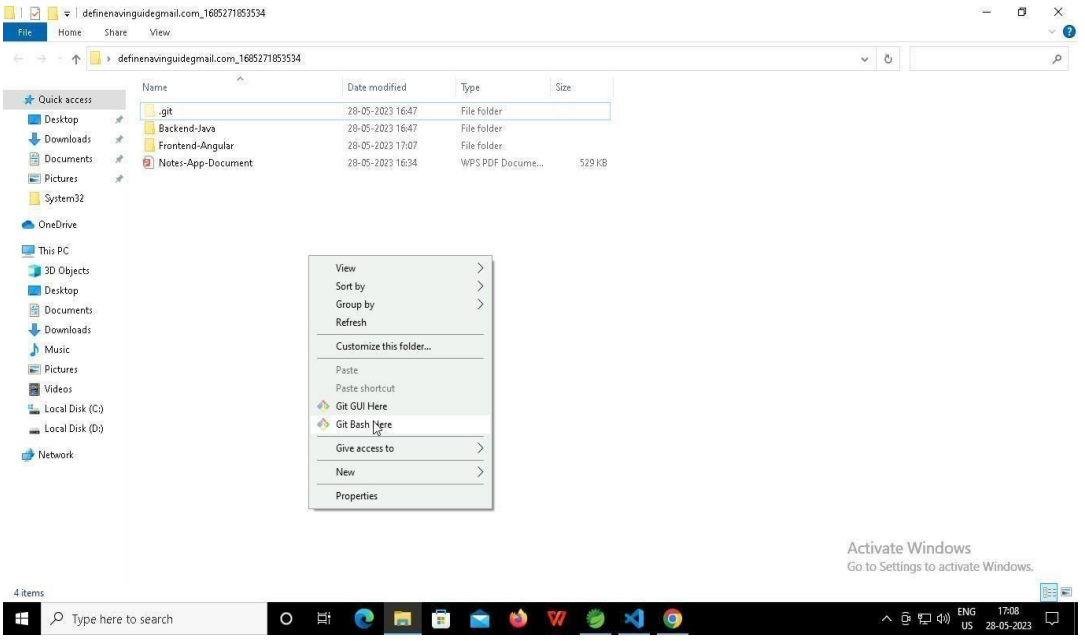
-----x-----

You can run test cases as many numbers of times and at any stage of Development, to check howmany test cases are passed/failed and accordingly refactor your code.

1. **Make sure before final submission you commit all changes to git**. For that open theproject folder available on desktop

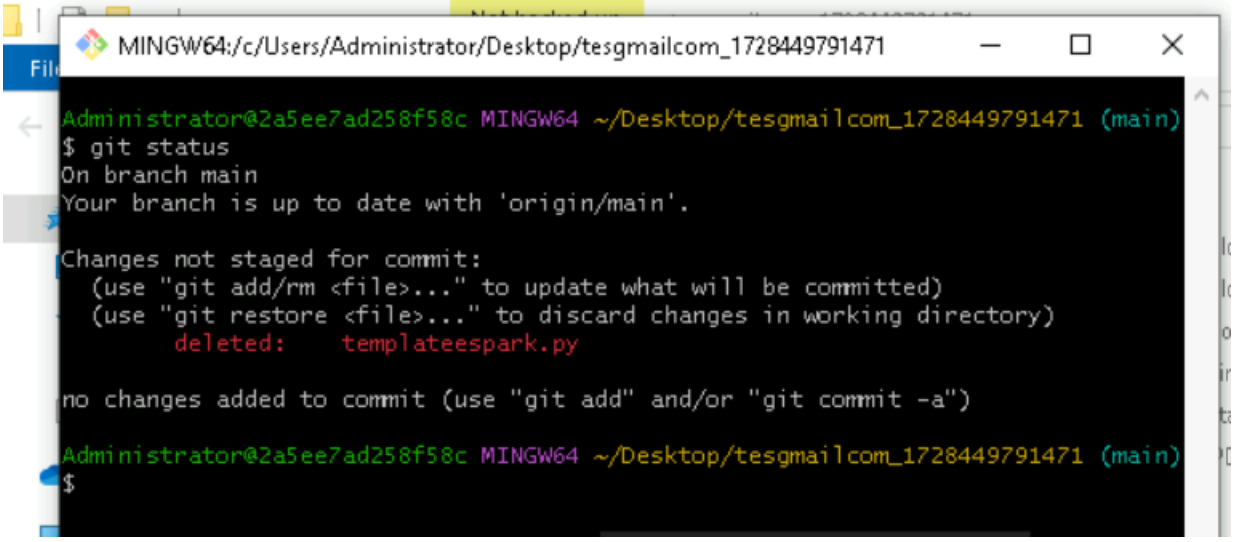


a. **Right click in folder and open Git Bash**



b. In Git bash terminal, run following commands

c. git status



d. git add .

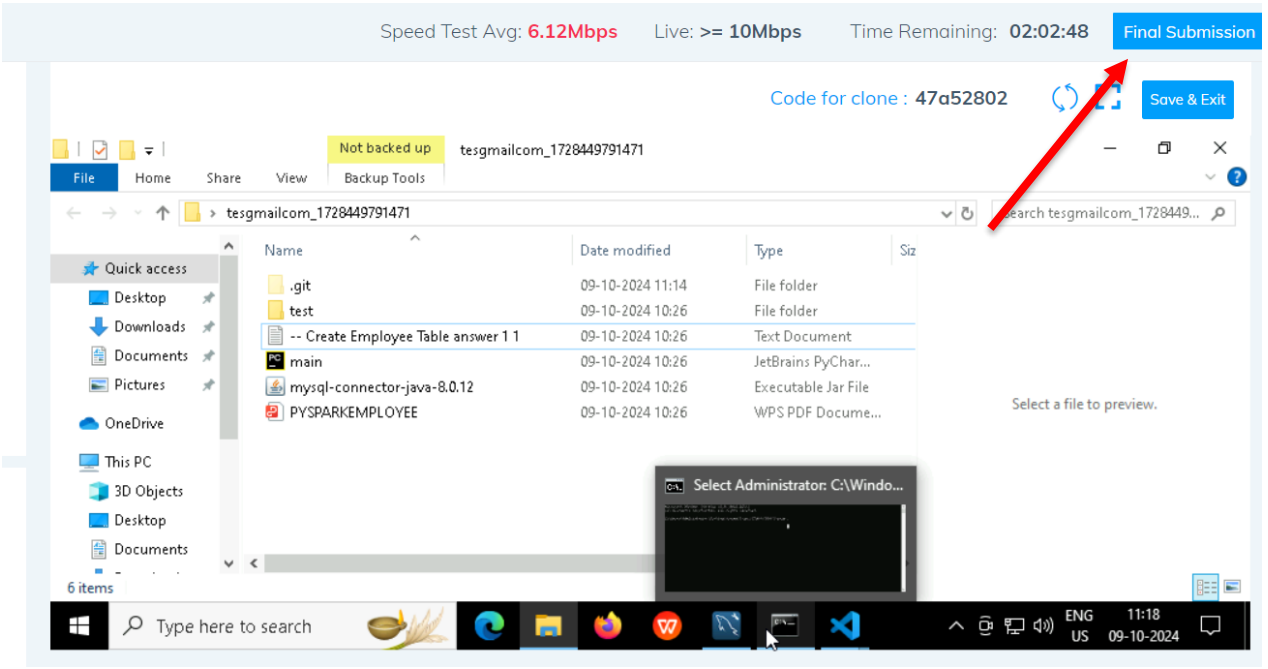
e. git commit -m “First commit”
(You can provide any message every time you commit)

```
Administrator@2a5ee7ad258f58c MINGW64 ~/Desktop/tesgmailcom_1728449791471 (main)
$ git commit -m "first commit"
[main f97ce24] first commit
 1 file changed, 91 deletions(-)
 delete mode 100644 templateespark.py
```

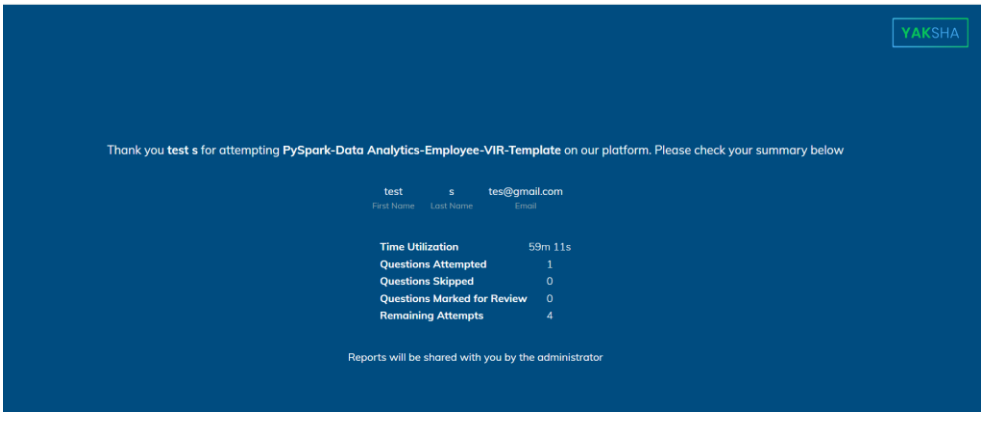
f. git push

```
Administrator@2a5ee7ad258f58c MINGW64 ~/Desktop/tesgmailcom_1728449791471 (main)
$ git push
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 212 bytes | 212.00 KiB/s, done.
Total 2 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/IIHTDevelopers/tesgmailcom_1728449791471.git
   a1c1905..f97ce24  main -> main
```

After you have pushed your code Finally click on the final submission button



You should see a screen like this you will have to wait for the results . after getting this page you can leave the system



-----x-----