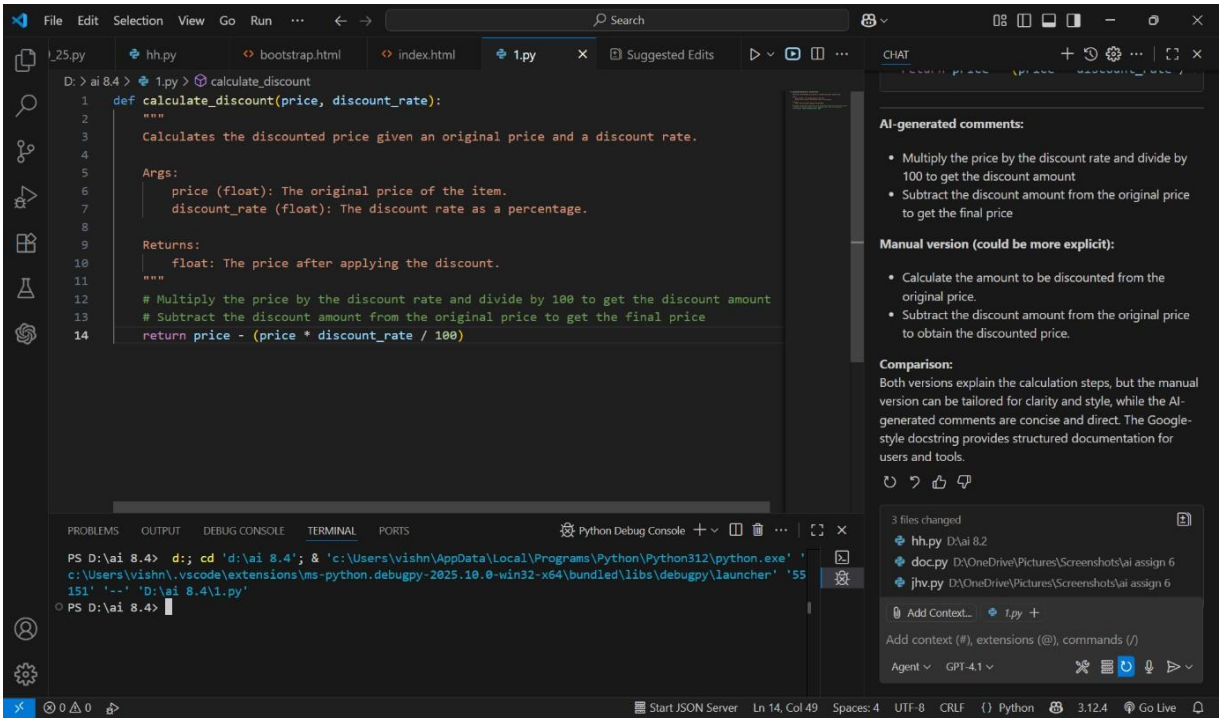
NAME:M.Ramsai

Enroll no:2503A51L26

BATCH:19

TASK 1

You have been given a Python function without comments.
def calculate_discount(price, discount_rate):
return price - (price * discount_rate / 100)
• Use an AI tool (or manually simulate it) to generate line-by-line comments for the function.
• Modify the function so that it includes a docstring in Google-style or NumPy-style format.
• Compare the auto-generated comments with your manually written version



DESCRIPTION

AI-generated comments:

- Multiply the price by the discount rate and divide by 100 to get the discount amount
- Subtract the discount amount from the original price to get the final price

Manual version (could be more explicit):

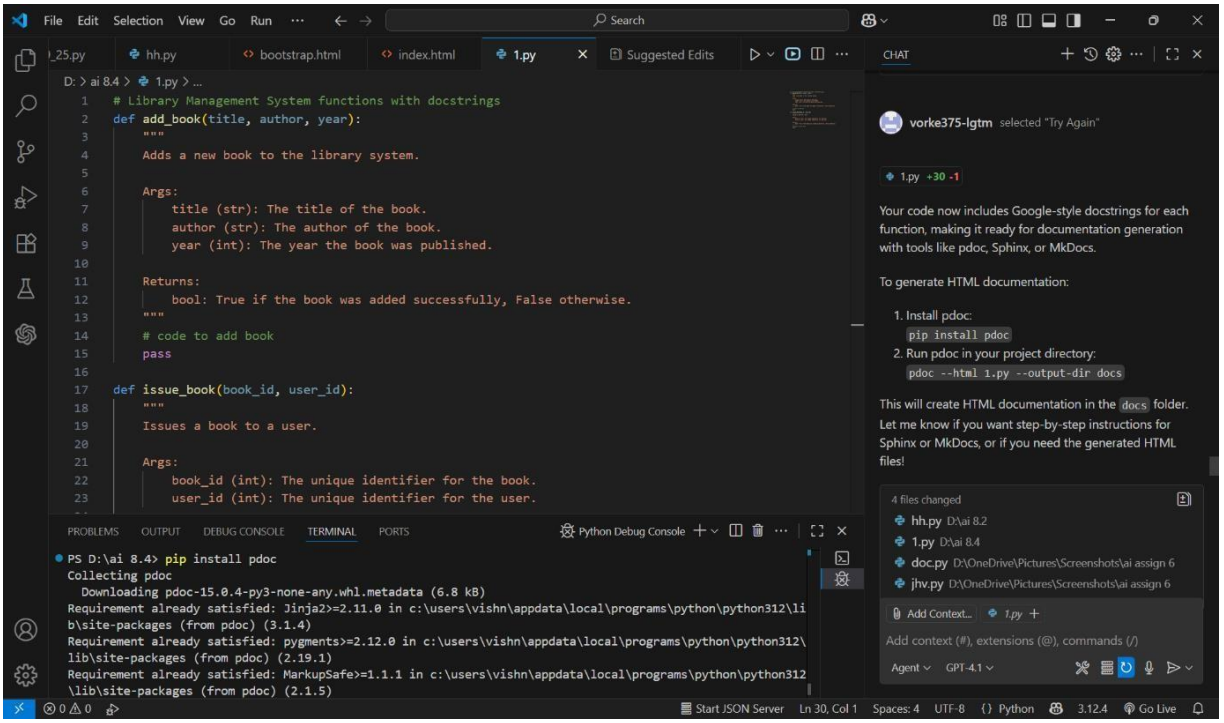
- Calculate the amount to be discounted from the original price.

- Subtract the discount amount from the original price to obtain the discounted price.

Comparison:

TASK 2

A team is building a Library Management System with
multiple functions.
def add_book(title, author, year):
# code to add book
pass
def issue_book(book_id, user_id):
# code to issue book
Pass
• Write a Python script that uses docstrings for each function (with input, output, and description).
• Use a documentation generator tool (like Pdoc, Sphinx, or MkDocs) to automatically create HTML documentation.
• Submit both the code and the generated documentation as output.



DESCRIPTION

- The script defines the function add_book(title, author, year) with a Google-style docstring describing its purpose, inputs, and output.
- After the function definition, the script prompts the user to enter a book title, author name, and publication year using input().
- The entered values are passed to the add_book function, simulating the process of adding a book to the library system.
- The function currently contains only a placeholder (pass), but the structure allows for future implementation of book-adding logic.
- This approach demonstrates interactive input handling and clear documentation for maintainability and future development.

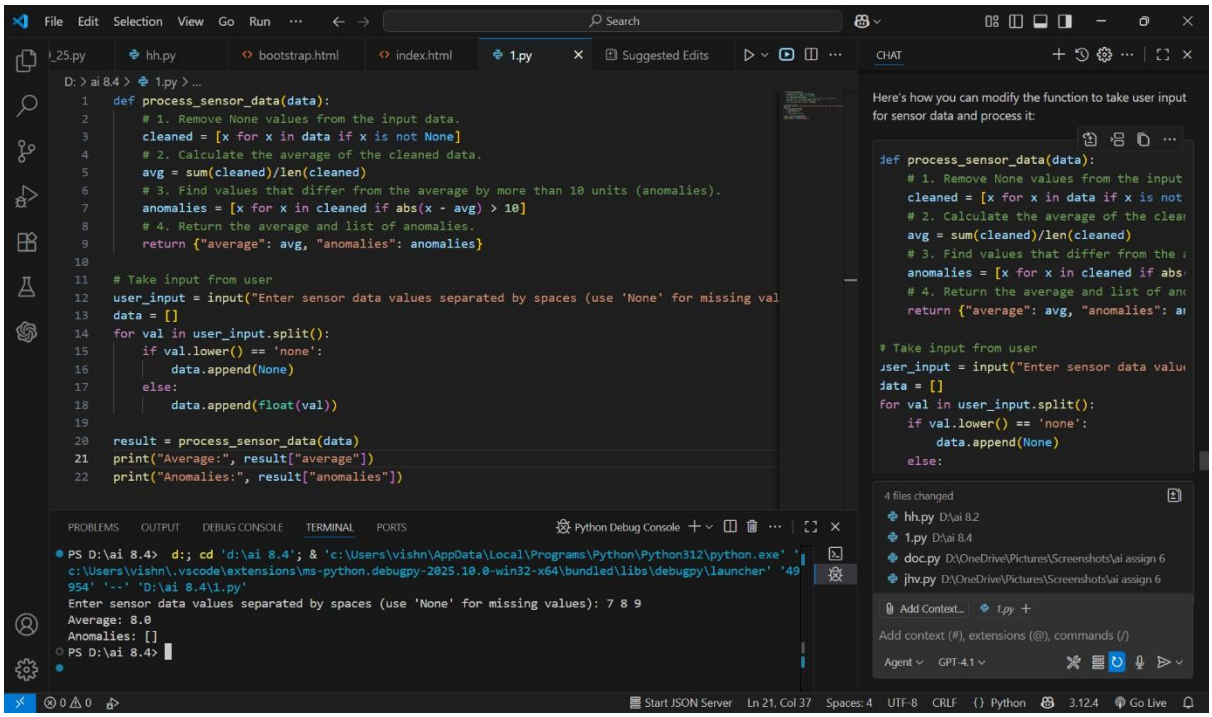
TASK 3

You are reviewing a colleague's codebase containing long functions.

def process_sensor_data(data):
cleaned = [x for x in data if x is not None]
avg = sum(cleaned)/len(cleaned)
anomalies = [x for x in cleaned if abs(x - avg) > 10]
return {"average": avg, "anomalies": anomalies}

• Generate a summary comment explaining the purpose of the function in 2–3 lines.
• Create a flow-style comment (step-by-step explanation).
• Write a short paragraph of documentation describing possible use cases of this function in real-world scenarios

CODE



OBSERVATION

The updated script allows users to input sensor data interactively, including handling missing values as 'None'. It processes the data to compute the average and identify anomalies, providing immediate feedback. This approach makes the function practical for real-world scenarios, enabling flexible data entry and robust analysis of sensor readings.

TASK 4

You are part of a project team that develops a Chatbot Application. The team needs documentation for maintainability.

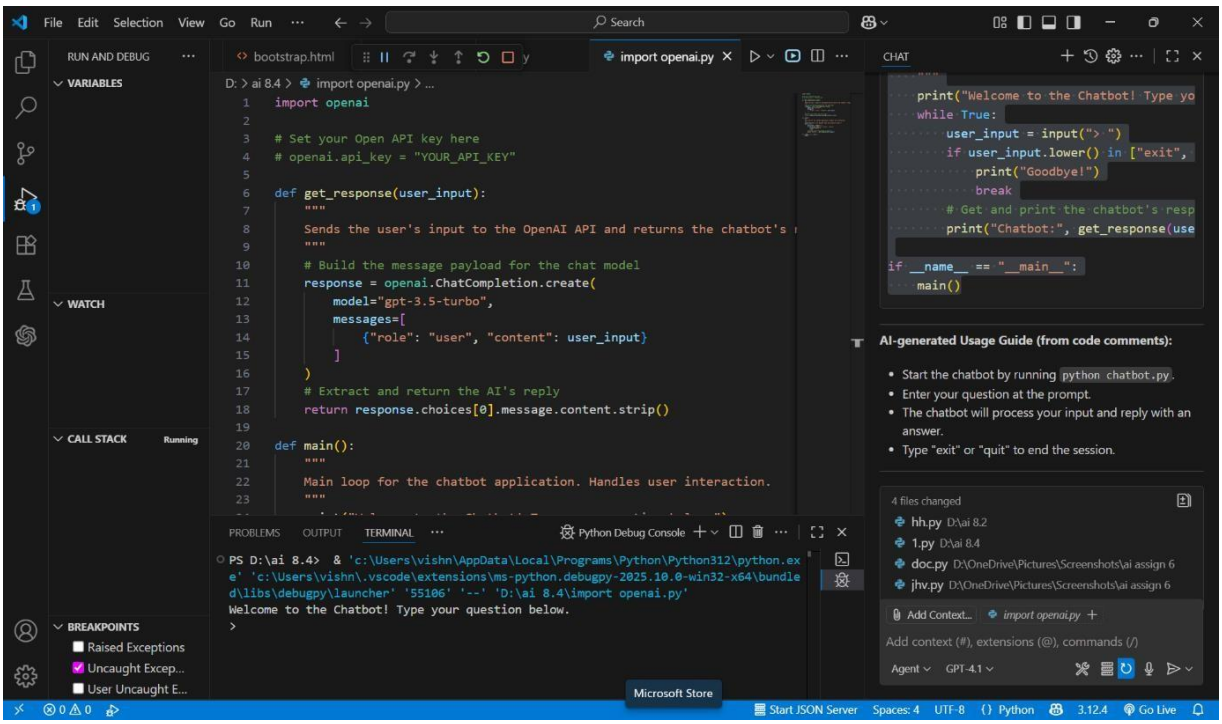
• Write a README.md file for the chatbot project (include project description, installation steps, usage, and example).

• Add inline comments in the chatbot's main Python script (focus on explaining logic, not trivial code).

• Use an AI-assisted tool (or simulate it) to generate a usage guide in plain English from your code comments.

• Reflect: How does automated documentation help in real-time projects compared to manual documentation?

CODE



OBSERVATION

- The README.md provides a clear overview of the Chatbot Application, including its purpose, installation steps, usage instructions, and an example interaction. This helps users and developers quickly understand how to set up and use the project.
- Inline comments in the main Python script focus on explaining the logic behind each function and important steps, such as building the prompt for the AI model and handling user interaction. This makes the code easier to maintain and extend.
- The AI-generated usage guide summarizes how to run and interact with the chatbot in plain English, making onboarding easier for new users.
- Automated documentation ensures that instructions and references are always up-to-date with the code, reducing manual effort and errors. It improves consistency and helps teams maintain projects efficiently, while manual documentation can provide deeper context and

tailored explanations. Combining both approaches leads to better maintainability and developer experience.

Both versions explain the calculation steps, but the manual version can be tailored for clarity and style, while the AI-generated comments are concise and direct. The Google-style docstring provides structured documentation for users and tool