

SR UNIVERSITY
AI ASSISTED CODING

BEGALA HASINI

2503A51L13

Lab 9.5 – Documentation Generation: Automatic Documentation and Code Comments

Lab Objectives:

- Inline Comments.
- Docstrings.
- Auto-Documentation Tools.
- AI assisted Summarization.

TASK DESCRIPTION #1 (Automatic Code Commenting)

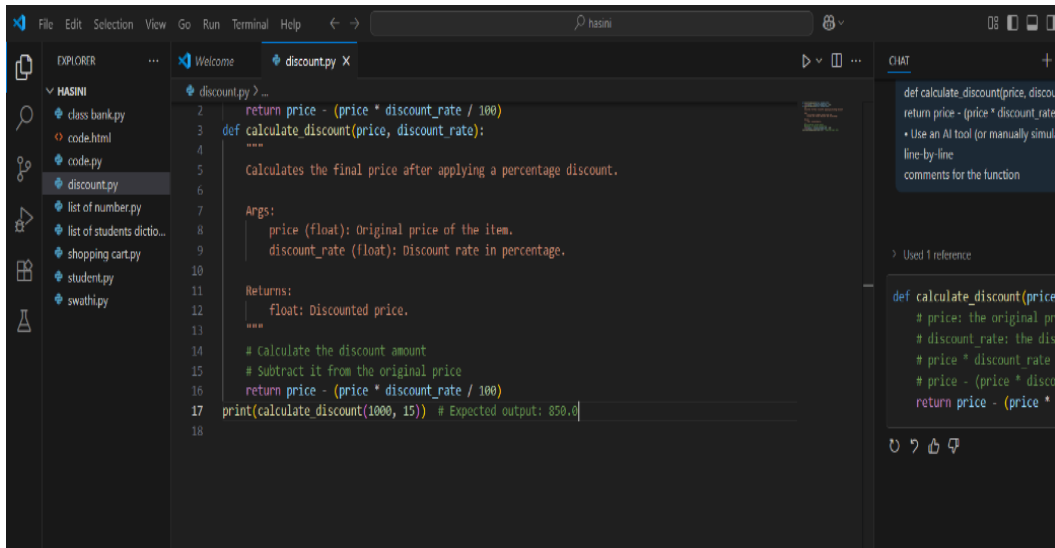
Scenario: You have been given a Python Function without comments .

```
def calculate_discount(price ,discount_rate):
```

```
    Return price –(price*discount_rate/100)
```

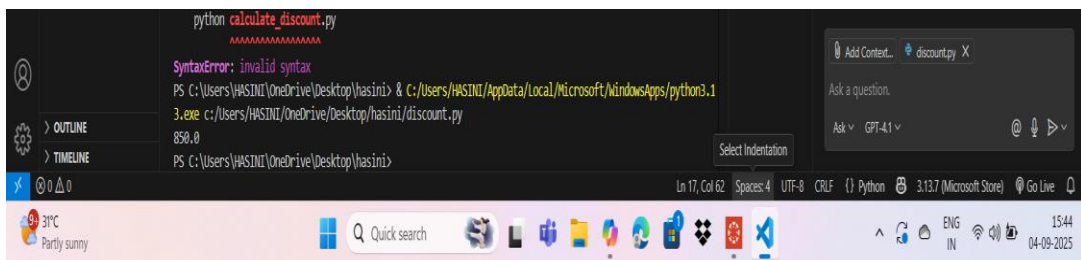
- Use an AI tool (or manually simulate it) to generate line-by-line comments for the function.
- Modify the function so that it includes a docstring in Google-style or NumPy-style format.
- Compare the auto-generated comments with your manually written version.

CODE:



```
1 return price - (price * discount_rate / 100)
2
3 def calculate_discount(price, discount_rate):
4     """
5     Calculates the final price after applying a percentage discount.
6
7     Args:
8         price (float): Original price of the item.
9         discount_rate (float): Discount rate in percentage.
10
11     Returns:
12         float: Discounted price.
13     """
14     # Calculate the discount amount
15     # Subtract it from the original price
16     return price - (price * discount_rate / 100)
17 print(calculate_discount(1000, 15)) # Expected output: 850.0
18
```

OUTPUT:



```
python calculate_discount.py
SyntaxError: invalid syntax
PS C:\Users\HASINI\OneDrive\Desktop\hasini> & C:/Users/HASINI/AppData/Local/Microsoft/WindowsApps/python3.13.exe c:/Users/HASINI/OneDrive/Desktop/hasini/discount.py
850.0
PS C:\Users\HASINI\OneDrive\Desktop\hasini>
```

TASK DESCRIPTION #2(API Documentation Generator)

Scenario: A Team is building a **Libaray Management System** with multiple functions.

def add_book(title, author, year):

code to add book

pass

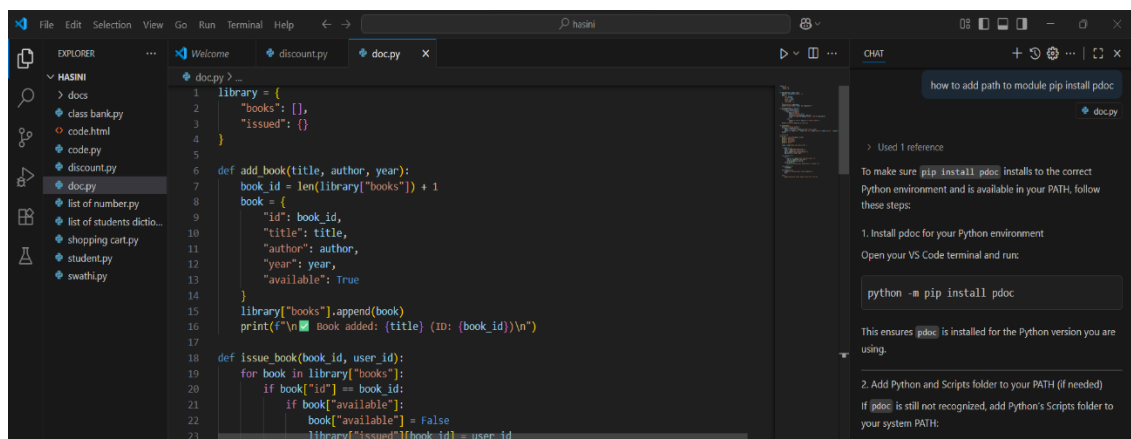
```
def issue_book(book_id, user_id)
```

```
# code to issue book
```

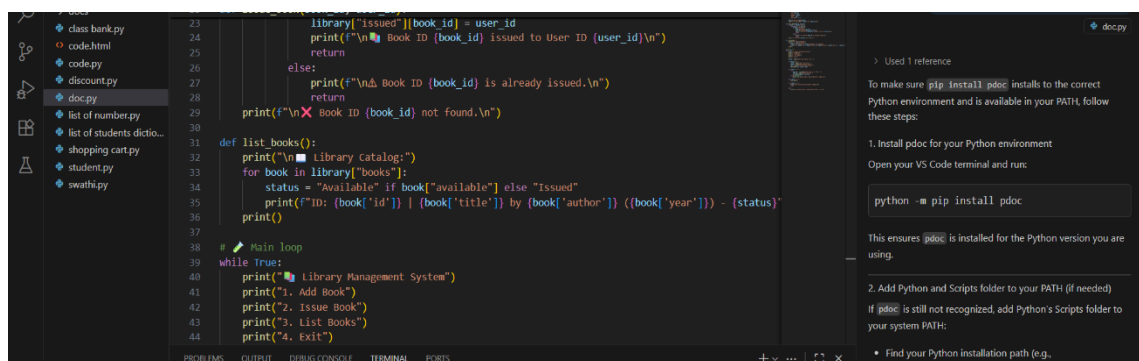
Pass

- 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.

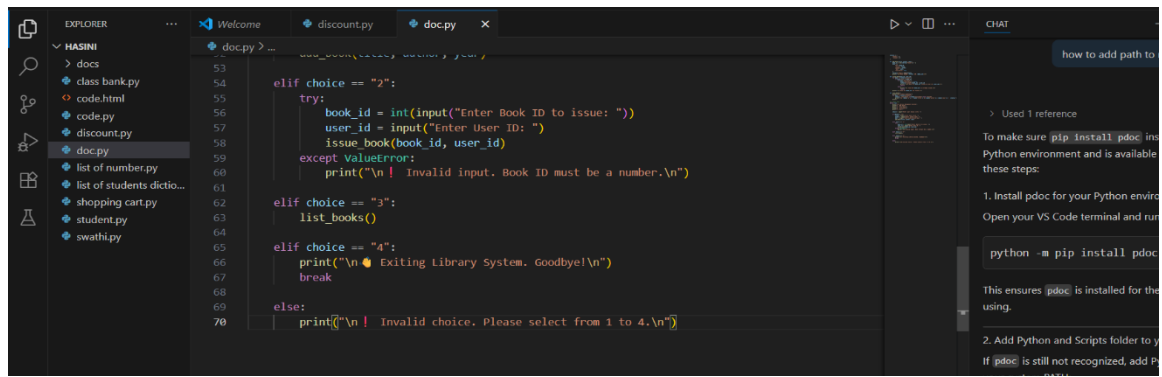
CODE :



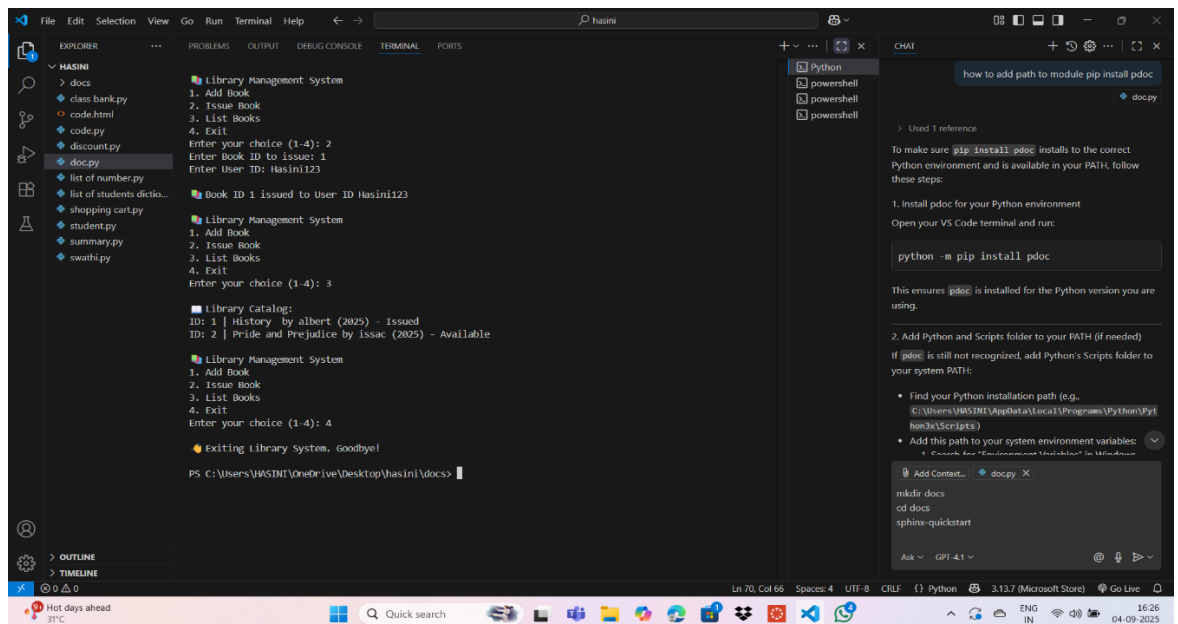
```
1 library = {
2     "books": [],
3     "issued": {}
4 }
5
6 def add_book(title, author, year):
7     book_id = len(library["books"]) + 1
8     book = {
9         "id": book_id,
10        "title": title,
11        "author": author,
12        "year": year,
13        "available": True
14    }
15    library["books"].append(book)
16    print(f"\n Book added: (title) (ID: {book_id})\n")
17
18 def issue_book(book_id, user_id):
19     for book in library["books"]:
20         if book["id"] == book_id:
21             if book["available"]:
```



```
23         library["issued"][book_id] = user_id
24         print(f"\n Book ID {book_id} issued to User ID {user_id}\n")
25         return
26     else:
27         print(f"\n Book ID {book_id} is already issued.\n")
28         return
29     print(f"\n Book ID {book_id} not found.\n")
30
31 def list_books():
32     print("\n Library Catalog:")
33     for book in library["books"]:
34         status = "Available" if book["available"] else "Issued"
35         print(f"ID: {book['id']} | {book['title']} by {book['author']} | ({book['year']}) - {status}")
36     print()
37
38 # Main loop
39 while True:
40     print("\n Library Management System")
41     print("1. Add Book")
42     print("2. Issue Book")
43     print("3. List Books")
44     print("4. Exit")
```



OUTPUT :



TASK DESCRIPTION #3 (AI_Assisted Code Summarization)

SCENARIO: You are reviewing a colleagues codebase containing long functions.

```
def process_sensor_data(data):
```

```
Cleansed = [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 :

```

5 # Flow-style steps:
6 # 1. Remove None values from the input data.
7 # 2. Calculate the average of the remaining readings.
8 # 3. Find readings that differ from the average by more than 10 units (anomalies).
9 # 4. Return both the average and the list of anomalies in a dictionary.
10
11 """
12 This function is useful for analyzing sensor data streams, such as temperature, humidity, or pressure,
13 where occasional faulty or missing values may occur. It helps in cleaning the data, summarizing trends,
14 and flagging outliers for further investigation or alerting. Applications include IoT device monitoring,
15 environmental data analysis, and automated quality control systems.
16 """
17
18 def process_sensor_data(data):
19     # Step 1: Filter out None values from the input list
20     cleaned = [x for x in data if x is not None]
21
22     # Step 2: Compute the average of the cleaned data
23     avg = sum(cleaned) / len(cleaned)
24
25     # Step 3: Identify anomalies that differ from the average by more than 10 units
26     anomalies = [x for x in cleaned if abs(x - avg) > 10]
27
28     # Step 4: Return the average and list of anomalies as a dictionary
29     return {"average": avg,}
30
31     # Step 4: Return the average and list of anomalies as a dictionary
32     return {"average": avg, "anomalies": anomalies}
33
34     # ...existing code...
35
36 # Example usage:
37 if __name__ == "__main__":
38     data = [20, 22, None, 35, 21, 19, 50, None, 23]
39     result = process_sensor_data(data)
40     print(result)

```

OUTPUT:

```

PS C:\Users\HASINI\OneDrive\Desktop\hasini\docs> & C:\Users\HASINI\AppData\Local\Microsoft\WindowsApps\python3.13.exe c:\Users\HASINI\OneDrive\Desktop\hasini\summary.py
{"average": 27.142857142857142, "anomalies": [20, 22, 35, 21, 19, 50, None, 23]}

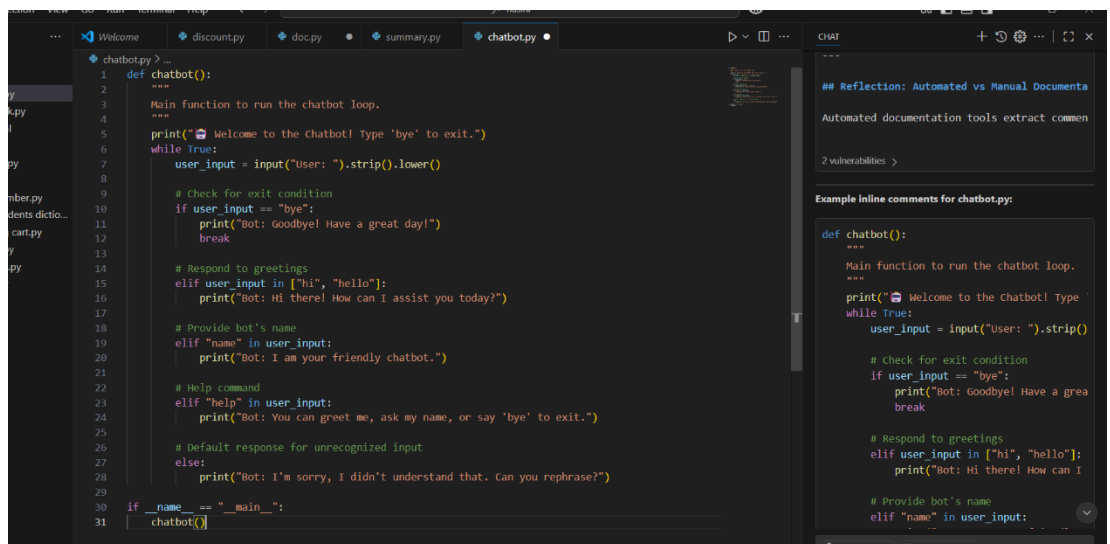
```

TASK DESCRIPTION #4 (Real-Time Project Documentation)

Scenario: 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



```
1 def chatbot():
2     """
3     Main function to run the chatbot loop.
4     """
5     print("👋 Welcome to the Chatbot! Type 'bye' to exit.")
6     while True:
7         user_input = input("User: ").strip().lower()
8
9         # Check for exit condition
10        if user_input == "bye":
11            print("Bot: Goodbye! Have a great day!")
12            break
13
14        # Respond to greetings
15        elif user_input in ["hi", "hello"]:
16            print("Bot: Hi there! How can I assist you today?")
17
18        # Provide bot's name
19        elif "name" in user_input:
20            print("Bot: I am your friendly chatbot.")
21
22        # Help command
23        elif "help" in user_input:
24            print("Bot: You can greet me, ask my name, or say 'bye' to exit.")
25
26        # Default response for unrecognized input
27        else:
28            print("Bot: I'm sorry, I didn't understand that. Can you rephrase?")
29
30 if __name__ == "__main__":
31     chatbot()
```

CHAT

Reflection: Automated vs Manual Documenta

Automated documentation tools extract commen

2 vulnerabilities >

Example inline comments for chatbot.py:

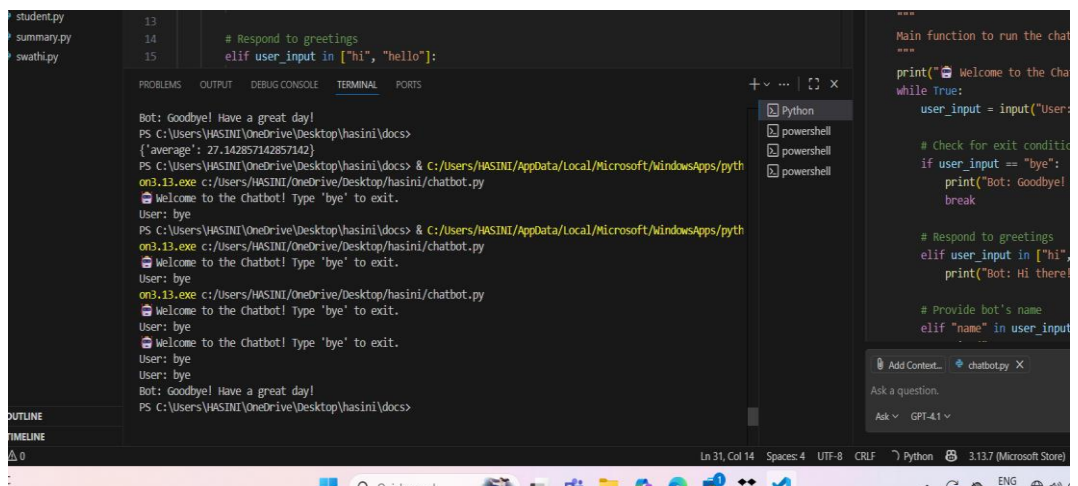
```
def chatbot():
    """
    Main function to run the chatbot loop.
    """
    print("👋 Welcome to the Chatbot! Type
    while True:
        user_input = input("User: ").strip()

        # Check for exit condition
        if user_input == "bye":
            print("Bot: Goodbye! Have a grea
            break

        # Respond to greetings
        elif user_input in ["hi", "hello"]:
            print("Bot: Hi there! How can I

        # Provide bot's name
        elif "name" in user_input:
```

OUTPUT



```
13
14 # Respond to greetings
15 elif user_input in ["hi", "hello"]:
```

Bot: Goodbye! Have a great day!

PS C:\Users\HASINI\OneDrive\Desktop\hasini\docs>

{'average': 27.142857142857142}

PS C:\Users\HASINI\OneDrive\Desktop\hasini\docs> & C:\Users\HASINI\AppData\Local\Microsoft\WindowsApps\python3.13.exe c:\Users\HASINI\OneDrive\Desktop\hasini\chatbot.py

👋 Welcome to the Chatbot! Type 'bye' to exit.

User: bye

PS C:\Users\HASINI\OneDrive\Desktop\hasini\docs> & C:\Users\HASINI\AppData\Local\Microsoft\WindowsApps\python3.13.exe c:\Users\HASINI\OneDrive\Desktop\hasini\chatbot.py

👋 Welcome to the Chatbot! Type 'bye' to exit.

User: bye

👋 Welcome to the Chatbot! Type 'bye' to exit.

User: bye

👋 Welcome to the Chatbot! Type 'bye' to exit.

User: bye

👋 Welcome to the Chatbot! Type 'bye' to exit.

User: bye

Bot: Goodbye! Have a great day!

PS C:\Users\HASINI\OneDrive\Desktop\hasini\docs>

Main function to run the chat

```
print("👋 Welcome to the Cha
while True:
    user_input = input("User:

    # Check for exit conditio
    if user_input == "bye":
        print("Bot: Goodbye!
        break

    # Respond to greetings
    elif user_input in ["hi",
        print("Bot: Hi there!
```

OBSERVATIONS :

By completing this Assignment ,I am able to:

- Compare the auto-generated comments with manually written version using AI Tools.
- Generates the API Documentation Generator using the python scripts .
- Generated the AI_Assisted Code Summarization.
- Implements the Real-Time Project Documentation.