**AI ASSISTED CODING**

**ASSIGNMENT – 9.5**

Name : Perala Akshaya

HT NO.: 2403A51264

BATCH:11

**Task-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

**Prompt**: Given this Python function:

def calculate\_discount (price, discount\_rate):

return price - (price \* discount\_rate / 100)

1. Add **line-by-line comments** explaining each line.
2. Rewrite the function with a **Google-style docstring** (include description, parameters, return, example).
3. Compare AI-generated comments with your manually written comments.

**Code:**

A screenshot of a computer program

AI-generated content may be incorrect.

A screen shot of a computer

AI-generated content may be incorrect.

**Output:**

A screen shot of a computer

AI-generated content may be incorrect.

**Task-2: (API Documentation Generator)**

Scenario: 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.

**Prompt:** Given these functions:

def add\_book (title, author, year):

pass

def issue\_book (book\_id, user\_id):

pass

1. Add **docstrings** with description, inputs, outputs, and example.
2. Generate **HTML documentation** using pdoc, Sphinx, or MkDocs.
3. Submit the Python script and generated docs.

**Code:**

A screen shot of a computer program

AI-generated content may be incorrect.

A computer screen shot of a program code

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.

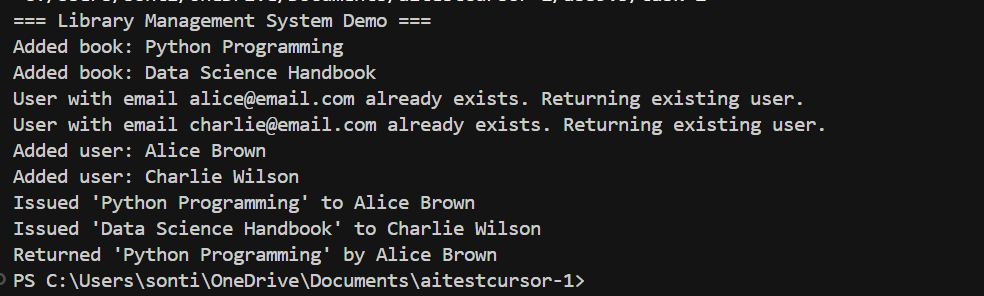
A screen shot of a computer program

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.

Output:



**Task-3: (AI-Assisted Code Summarization)**

Scenario: 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.

**Prompt:** Given this function:

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}

Write:

1. A 2–3-line summary comment.

2. Step-by-step flow-style comments.

3. A short paragraph on real-world use cases.

**Code:**

A screenshot of a computer program

AI-generated content may be incorrect.

A computer screen with text and numbers

AI-generated content may be incorrect.

**Output:**

A screenshot of a computer screen

AI-generated content may be incorrect.

**Task-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?

**Prompt**: You are an AI Python documentation assistant.

1. Add Google-style docstrings to all functions in this Python script.

2. Add inline comments explaining the logic.

3. Generate a simple plain-English usage guide.

Python Script:

<PASTE YOUR PYTHON CODE HERE>

Output:

- Python code with docstrings and comments.

- Plain-English usage guide.

**Code:**

**Readme code:**

A screenshot of a chatbot application

AI-generated content may be incorrect.

**Chatbot code:**

A screenshot of a computer program

AI-generated content may be incorrect.

**Output:**

A screenshot of a computer screen

AI-generated content may be incorrect.