SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE		DEPARTMENT OF COMPUTER SCIENCE ENGINEERING	
ProgramName:B. Tech		**Assignment Type: Lab**	**AcademicYear:**2025-2026
CourseCoordinatorName	Venkataramana Veeramsetty		
Instructor(s)Name	Dr. V. Venkataramana (Co-ordinator) Dr. T. Sampath Kumar Dr. Pramoda Patro Dr. Brij Kishor Tiwari Dr.J.Ravichander Dr. Mohammand Ali Shaik Dr. Anirodh Kumar Mr. S.Naresh Kumar Dr. RAJESH VELPULA Mr. Kundhan Kumar Ms. Ch.Rajitha Mr. M Prakash Mr. B.Raju Intern 1 (Dharma teja) Intern 2 (Sai Prasad) Intern 3 (Sowmya) NS_2 (Mounika)		
CourseCode	24CS002PC215	**CourseTitle**	AI Assisted Coding
Year/Sem	II/I	**Regulation**	R24
Date and Day of Assignment	Week4 - Wednesday	**Time(s)**	
Duration	2 Hours	**Applicableto Batches**	
AssignmentNumber:9.3(Present assignment number)/**24**(Total number of assignments)			

Q.No.	Question	*ExpectedTime to complete*
1	Lab 8: Documentation Generation: Automatic documentation and code comments **Lab Objectives:** • To understand the importance of documentation and code comments in software development. • To explore how AI-assisted coding tools can generate meaningful documentation and	Week4 - Wednesday

inline comments.

- To practice generating function-level and module-level docstrings automatically.
- To evaluate the quality, accuracy, and limitations of AI-generated documentation.
- To develop a small automated tool for documentation generation in Python..

Lab Outcomes (LOs):
After completing this lab, students will be able to:

- Apply AI-assisted coding tools to generate docstrings and inline comments for Python code.
- Critically analyze AI-generated documentation for correctness, completeness, and readability.
- Create structured documentation (function-level, module-level) following standard formats.

- Design and implement a mini documentation generator tool to automate code commenting and docstring creation.

Task Description#1 Basic Docstring Generation

- Write python function to return sum of even and odd numbers in the given list.
- Incorporate manual **docstring** in code with Google Style
- Use an AI-assisted tool (e.g., Copilot, Cursor AI) to generate a docstring describing the function.
- Compare the AI-generated docstring with your manually written one.

Expected Outcome#1: Students understand how AI can produce function-level documentation.

Task Description#2 Automatic Inline Comments

- Write python program for **sru_student** class with attributes like name, roll no., hostel_status and **fee_update** method and **display_details** method.
- Write comments manually for each line/code block
- Ask an AI tool to add inline comments explaining each line/step.
- Compare the AI-generated comments with your manually written one.

Expected Output#2: Students critically analyze AI-generated code comments.

Task Description#3

- Write a Python script with 3–4 functions (e.g., calculator: add, subtract, multiply, divide).
- Incorporate manual **docstring** in code with NumPy Style
- Use AI assistance to generate a module-level docstring + individual function docstrings.
- Compare the AI-generated docstring with your manually written one.

Expected Output#3: Students learn structured documentation for multi-function scripts

Push documentation whole workspace as .md file in GitHub Repository

Note: Report should be submitted a word document for all tasks in a single document with prompts, comments & code explanation, and output and if required, screenshots

TASK 1:
CODE BY USER:

```python
def sum_even_odd(numbers):
    even_sum = 0
    odd_sum = 0
    for num in numbers:
        if num % 2 == 0:
            even_sum += num
        else:
            odd_sum += num
    return even_sum, odd_sum
user_input = input("Enter numbers separated by spaces: ")
numbers = [int(x) for x in user_input.split()]

even, odd = sum_even_odd(numbers)
print("Sum of even numbers:", even)
print("Sum of odd numbers:", odd)
```

```
PS C:\Users\Anusha\OneDrive\Desktop\AI\ai>  & 'c:\Users\Anusha\AppData\Local\Program
\Users\Anusha\.vscode\extensions\ms-python.debugpy-2025.10.0-win32-x64\bundled\libs\
\Users\Anusha\OneDrive\Desktop\AI\ai\task1.py'
Enter numbers separated by spaces: 2 5 8 3 7
Sum of even numbers: 10
Sum of odd numbers: 15
PS C:\Users\Anusha\OneDrive\Desktop\AI\ai> ^C
PS C:\Users\Anusha\OneDrive\Desktop\AI\ai>
PS C:\Users\Anusha\OneDrive\Desktop\AI\ai>  c:; cd 'c:\Users\Anusha\OneDrive\Desktop
a\Local\Programs\Python\Python313\python.exe' 'c:\Users\Anusha\.vscode\extensions\ms
4\bundled\libs\debugpy\launcher' '60068' '--' 'c:\Users\Anusha\OneDrive\Desktop\AI\a
```

MANUAL DOCSTRING CODE :

```python
def sum_even_odd(numbers):
    """
    This function adds up all the even numbers and all the odd numbers in the list.
    You give it a list of numbers, and it tells you the total of the evens and the total of the odds.
    Returns both sums as a pair.
    """
    even_sum = 0
    odd_sum = 0
    for num in numbers:
        if num % 2 == 0:
            even_sum += num
        else:
            odd_sum += num
    return even_sum, odd_sum
user_input = input("Enter numbers separated by spaces: ")
numbers = [int(x) for x in user_input.split()]

even, odd = sum_even_odd(numbers)
print("Sum of even numbers:", even)
print("Sum of odd numbers:", odd)
```

DOCSTRING IN CODE WITH GOOGLE STYLE :

Welcome | task1.py

task1.py > sum_even_odd

```python
def sum_even_odd(numbers):
    """
    Calculates the sum of even and odd numbers in a given list.

    Args:
        numbers (list of int): A list of integers to be processed.

    Returns:
        tuple: A tuple containing two integers:
            - The sum of even numbers.
            - The sum of odd numbers.
    """
    even_sum = 0
    odd_sum = 0
    for num in numbers:
        if num % 2 == 0:
            even_sum += num
        else:
            odd_sum += num
    return even_sum, odd_sum
user_input = input("Enter numbers separated by spaces: ")
numbers = [int(x) for x in user_input.split()]

even, odd = sum_even_odd(numbers)
print("Sum of even numbers:", even)
print("Sum of odd numbers:", odd)
```

COMPARISION : The user docstring is informal,simple and easy to understand to a friend where as ai docstring is formal, structutred and formatting (like ards and returns).

TASK 2:
CODE BY USER:

Welcome | task1.py | task2.py

task2.py > ...

```python
class sru_student:
    def __init__(self, name, roll_no, hostel_status):
        self.name = name
        self.roll_no = roll_no
        self.hostel_status = hostel_status
        self.fee_paid = False

    def fee_update(self, status):
        self.fee_paid = status

    def display_details(self):
        print("Name:", self.name)
        print("Roll No:", self.roll_no)
        print("Hostel Status:", self.hostel_status)
        print("Fee Paid:", "Yes" if self.fee_paid else "No")

name = input("Enter student name: ")
roll_no = input("Enter roll number: ")
hostel_status = input("Hostel status (Yes/No): ")

student = sru_student(name, roll_no, hostel_status)

fee_status = input("Has the fee been paid? (Yes/No): ")
student.fee_update(fee_status.lower() == "yes")

student.display_details()
```

OUTPUT:

```
 'c:\Users\Anusha\OneDrive\Desktop\AI\ai\task2.py'
Enter student name: bhavana
Enter roll number: 2403a52249
Hostel status (Yes/No): yes
Has the fee been paid? (Yes/No): yes
Name: bhavana
Roll No: 2403a52249
Hostel Status: yes
Fee Paid: Yes
PS C:\Users\Anusha\OneDrive\Desktop\AI\ai>
```

MANUAL COMMENTS:

Welcome | task1.py | task2.py

task2.py > sru_student

```python
class sru_student:
    # This sets up the student with their name, roll number, hostel status, and fee status
    def __init__(self, name, roll_no, hostel_status):
        self.name = name
        self.roll_no = roll_no
        self.hostel_status = hostel_status
        self.fee_paid = False

    # This method updates whether the fee is paid or not
    def fee_update(self, status):
        self.fee_paid = status

    # This method prints out all the details of the student
    def display_details(self):
        print("Name:", self.name)
        print("Roll No:", self.roll_no)
        print("Hostel Status:", self.hostel_status)
        print("Fee Paid:", "Yes" if self.fee_paid else "No")

# Ask the user for student details
name = input("Enter student name: ")
roll_no = input("Enter roll number: ")
hostel_status = input("Hostel status (Yes/No): ")

# Make a student object with the details
student = sru_student(name, roll_no, hostel_status)

# Ask if the fee is paid and update it
fee_status = input("Has the fee been paid? (Yes/No): ")
student.fee_update(fee_status.lower() == "yes")

# Show all the details
student.display_details()
name = input("Enter student name: ")
roll_no = input("Enter roll number: ")
hostel_status = input("Hostel status (Yes/No): ")

student = sru_student(name, roll_no, hostel_status)

fee_status = input("Has the fee been paid? (Yes/No): ")
student.fee_update(fee_status.lower() == "yes")

student.display_details()
```

INLINE COMMENTS EXPLAINED BY THE AI :

Welcome | task1.py | task2.py

task2.py > sru_student > __init__

```python
class sru_student:
    """
    A class representing a student at SRU with attributes for name, roll number, hostel status, and fee payment status.
    Attributes:
        name (str): The name of the student.
        roll_no (str or int): The roll number of the student.
        hostel_status (str): The hostel accommodation status of the student.
        fee_paid (bool): Indicates whether the student's fee has been paid.
    Methods:
        __init__(name, roll_no, hostel_status):
            Initializes a new SRU student with the given name, roll number, and hostel status. Fee status is set to unpaid by default.
        fee_update(status):
            Updates the fee payment status of the student.
        display_details():
            Prints all details of the student including name, roll number, hostel status, and fee payment status.
    """
    # This sets up the student with their name, roll number, hostel status, and fee status
    def __init__(self, name, roll_no, hostel_status):
        self.name = name
        self.roll_no = roll_no
        self.hostel_status = hostel_status
        self.fee_paid = False

    # This method updates whether the fee is paid or not
    def fee_update(self, status):
        self.fee_paid = status

    # This method prints out all the details of the student
    def display_details(self):
        print("Name:", self.name)
        print("Roll No:", self.roll_no)
        print("Hostel Status:", self.hostel_status)
        print("Fee Paid:", "Yes" if self.fee_paid else "No")

# Ask the user for student details
name = input("Enter student name: ")
roll_no = input("Enter roll number: ")
hostel_status = input("Hostel status (Yes/No): ")

# Make a student object with the details
student = sru_student(name, roll_no, hostel_status)

# Ask if the fee is paid and update it
fee_status = input("Has the fee been paid? (Yes/No): ")
student.fee_update(fee_status.lower() == "yes")

# Show all the details
student.display_details()
name = input("Enter student name: ")
roll_no = input("Enter roll number: ")
hostel_status = input("Hostel status (Yes/No): ")

student = sru_student(name, roll_no, hostel_status)

fee_status = input("Has the fee been paid? (Yes/No): ")
student.fee_update(fee_status.lower() == "yes")

student.display_details()
```

COMPARISION : The user docstring is informal,simple and easy to understand to a friend where as ai docstring is formal, structutred and suitable for technical documentation.

TASK 3:

CODE BY USER :

Welcome | task1.py | task2.py | task3.py

task3.py > ...

```python
def add(a, b):
    return a + b

def subtract(a, b):
    return a - b

def multiply(a, b):
    return a * b

def divide(a, b):
    if b == 0:
        return "Cannot divide by zero"
    return a / b
x = float(input("Enter first number: "))
y = float(input("Enter second number: "))

op = input("Choose operation (+, -, *, /): ")

if op == "+":
    print("Result:", add(x, y))
elif op == "-":
    print("Result:", subtract(x, y))
elif op == "*":
    print("Result:", multiply(x, y))
elif op == "/":
    print("Result:", divide(x, y))
else:
    print("Invalid operation")
```

```
PS C:\Users\Anusha\OneDrive\Desktop\AI\ai>
PS C:\Users\Anusha\OneDrive\Desktop\AI\ai>  c:; cd
ms\Python\Python313\python.exe' 'c:\Users\Anusha\.
her' '54038' '--' 'c:\Users\Anusha\OneDrive\Deskto
Enter first number: 10
Enter second number: 5
Choose operation (+, -, *, /): +
Result: 15.0
PS C:\Users\Anusha\OneDrive\Desktop\AI\ai>
```

MANUAL DOCSTRING WITH CODE IN NUMPY STYLE:

task3.py > ...

```python
"""
Simple calculator for two numbers.
"""

def add(a, b):
    """Add two numbers."""
    return a + b

def subtract(a, b):
    """Subtract b from a."""
    return a - b

def multiply(a, b):
    """Multiply two numbers."""
    return a * b

def divide(a, b):
    """Divide a by b. Returns error if b is zero."""
    if b == 0:
        return "Cannot divide by zero"
    return a / b

x = float(input("Enter first number: "))
y = float(input("Enter second number: "))
op = input("Choose operation (+, -, *, /): ")

if op == "+":
    print("Result:", add(x, y))
elif op == "-":
    print("Result:", subtract(x, y))
elif op == "*":
    print("Result:", multiply(x, y))
elif op == "/":
    print("Result:", divide(x, y))
else:
    print("Invalid operation")
```

GENERATED BY A MODULE-LEVEL DOCSTRING+INDIVIDUAL FUNCTION :

Welcome | task1.py | task2.py | task3.py

task3.py > ...

```python
"""
Calculator for two numbers.
"""

def add(a, b):
    """Add two numbers."""
    return a + b

def subtract(a, b):
    """Subtract b from a."""
    return a - b

def multiply(a, b):
    """Multiply two numbers."""
    return a * b

def divide(a, b):
    """Divide a by b. Error if b is zero."""
    if b == 0:
        return "Cannot divide by zero"
    return a / b

x = float(input("Enter first number: "))
y = float(input("Enter second number: "))
op = input("Choose operation (+, -, *, /): ")

if op == "+":
    print("Result:", add(x, y))
elif op == "-":
    print("Result:", subtract(x, y))
elif op == "*":
    print("Result:", multiply(x, y))
elif op == "/":
    print("Result:", divide(x, y))
else:
    print("Invalid operation")
```

- **COMPARISION : The user docstring is very short ,informal,and direct where as ai docstring is longer,formal and follows documentation standards like numpy,return values.**
