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:

```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

# Example usage
nums = [1, 2, 3, 4, 5, 6]
even_total, odd_total = sum_even_odd(nums)
print("Sum of even numbers:", even_total)
print("Sum of odd numbers:", odd_total)
```

Output:

```
PS C:\Users\Namitha\OneDrive\Desktop\AIAC\Lab-9> & 'c:\Users\Namitha\AppData\Local\Microsoft\WindowsApps\python3.11.exe' 'c:\Users\Namitha\.vscode\extensions\ms-python.debugpy-2025.10.0-win32-x64\bundled\libs\debugpy\launcher' '52752' '--' 'c:\Users\Namitha\OneDrive\Desktop\AIAC\Lab-9\Task-1.py'
Sum of even numbers: 12
Sum of odd numbers: 9
PS C:\Users\Namitha\OneDrive\Desktop\AIAC\Lab-9>
```

Docstring by me :

```
def sum_even_odd(numbers):
   even_sum = 0
   odd_sum = 0
“ “ “
```

This function gives the sum of odd numbers and even numbers in given list.

Args:a list of numbers(integrs)

Return: sum of odd numbers and even numbers is given.it is integer.

“ “ “

```
    for num in numbers:
        if num % 2 == 0:
            even_sum += num
        else:
            odd_sum += num

    return even_sum, odd_sum

# Example usage
nums = [1, 2, 3, 4, 5, 6]
even_total, odd_total = sum_even_odd(nums)
print("Sum of even numbers:", even_total)
print("Sum of odd numbers:", odd_total)
```

Docstring given by AI:

Task-1.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

# Example usage
nums = [1, 2, 3, 4, 5, 6]
even_total, odd_total = sum_even_odd(nums)
print("Sum of even numbers:", even_total)
print("Sum of odd numbers:", odd_total)
```

Comparision: I wrote the docstring in a simple shortcut way.i did not give more information about the result and also about datatypes but AI explained about the function ,Arguments and result very clearly

including their data types.

Task-2:

Code:

```python
class SRU_Student:
    def __init__(self, name, roll_no, hostel_status, fee_paid=0):
        self.name = name
        self.roll_no = roll_no
        self.hostel_status = hostel_status   # Yes/No or True/False
        self.fee_paid = fee_paid

    # Method to update fee
    def fee_update(self, amount):
        self.fee_paid += amount
        print(f"₹{amount} added. Total fee paid: ₹{self.fee_paid}")

    # Method to display student details
    def display(self):
        print(f"Name: {self.name}")
        print(f"Roll No: {self.roll_no}")
        print(f"Hostel Status: {self.hostel_status}")
        print(f"Fee Paid: ₹{self.fee_paid}")

# Example usage
if __name__ == "__main__":
    student1 = SRU_Student("Namitha", "SRU123", "Yes")
    student1.display()
    student1.fee_update(5000)
    student1.display()
```

Output:

```
osoft\WindowsApps\python3.11.exe' 'c:\Users\Namitha\.vscode\extensions\ms-pytho
706' '--' 'c:\Users\Namitha\OneDrive\Desktop\AIAC\Lab-9\Task-2.py'
Name: Namitha
Roll No: SRU123
Hostel Status: Yes
Fee Paid: ₹0
Fee Paid: ₹0
Fee Paid: ₹0
₹5000 added. Total fee paid: ₹5000
Name: Namitha
Roll No: SRU123
Hostel Status: Yes
Fee Paid: ₹5000
PS C:\Users\Namitha\OneDrive\Desktop\AIAC\Lab-9>
```

Inline comments by me:

```
class SRU_Student:
    # Constructor method to initialize student details
    def __init__(self, name, roll_no, hostel_status, fee_paid=0):
        self.name = name              # Student's name
        self.roll_no = roll_no        # Student's roll number
        self.hostel_status = hostel_status   # Whether student stays in hostel (Yes/No)
        self.fee_paid = fee_paid        # Initial fee paid (default = 0)

    # Method to update (add) fee amount
    def fee_update(self, amount):
        self.fee_paid += amount   # Add new amount to total fee
        print(f"₹{amount} added. Total fee paid: ₹{self.fee_paid}")

    # Method to display student details
    def display(self):
        print(f"Name: {self.name}")
        print(f"Roll No: {self.roll_no}")
        print(f"Hostel Status: {self.hostel_status}")
        print(f"Fee Paid: ₹{self.fee_paid}")

# Example usage
if __name__ == "__main__":
    # Create a student object
    student1 = SRU_Student("Namitha", "SRU123", "Yes")

    # Display initial details
    student1.display()
```

```
# Update fee by 5000
student1.fee_update(5000)

# Display updated details
student1.display()
```

Inline comments by AI:

Welcome | Task-1.py | Task-2.py

Task-2.py > SRU_Student

```python
class SRU_Student:
    def __init__(self, name, roll_no, hostel_status, fee_paid=0):
        # Initialize student details
        self.name = name
        self.roll_no = roll_no
        self.hostel_status = hostel_status   # Yes/No or True/False
        self.fee_paid = fee_paid

    # Method to update fee
    def fee_update(self, amount):
        self.fee_paid += amount  # Add the amount to fee_paid
        print(f"₹{amount} added. Total fee paid: ₹{self.fee_paid}")

    # Method to display student details
    def display(self):
        print(f"Name: {self.name}")  # Print student's name
        print(f"Roll No: {self.roll_no}")  # Print roll number
        print(f"Hostel Status: {self.hostel_status}")  # Print hostel status
        print(f"Fee Paid: ₹{self.fee_paid}")  # Print fee paid

# Example usage
if __name__ == "__main__":
    student1 = SRU_Student("Namitha", "SRU123", "Yes")  # Create a student object
    student1.display()  # Display initial details
    student1.fee_update(5000)  # Update fee by 5000
    student1.display()  # Display updated details
```

Comparision : Almost same comments are given by AI.In fact it gave in a simple easy language.i gave more comments than the AI.itb gave only main and required comments.however both inline comments are easy and simple to understand.

Task-3:

Code:

```python
import numpy as np

# Calculator functions using NumPy
def add(a, b):
    return np.add(a, b)

def subtract(a, b):
    return np.subtract(a, b)

def multiply(a, b):
    return np.multiply(a, b)

def divide(a, b):
    try:
        return np.divide(a, b)
    except ZeroDivisionError:
        return "Error: Division by zero not allowed"


# Example usage
if __name__ == "__main__":
    x, y = 15, 3

    print("Numbers:", x, "and", y)
    print("Addition:", add(x, y))
    print("Subtraction:", subtract(x, y))
    print("Multiplication:", multiply(x, y))
    print("Division:", divide(x, y))
```

Output:

```
bundled\libs\debugpy\launcher' '55101' '--' 'c:\Users\Na
Numbers: 15 and 3
Numbers: 15 and 3
Addition: 18
Addition: 18
Subtraction: 12
Multiplication: 45
Division: 5.0
PS C:\Users\Namitha\OneDrive\Desktop\AIAC\Lab-9>
```

DocString by me:

```
# Simple Calculator Script

def add(a, b):
“ “ “
This function used to add two given numbers.

Args(integers): a, b are any two numbers.
Returns(integer):Addition of two numbers.
“ “ “
   return  np.add(a , b)

def subtract(a, b):

“ “ “
This function used for subtraction purpose.

Args(integers): a, b are any two numbers.
Returns(integer):subtraction of two numbers.
“ “ “
   return np.subtract(a, b)


def multiply(a, b):
“ “ “
This function used for multiplication of two numbers.

Args(integers): a, b are any two numbers.

Returns(integer): multiplication of two numbers.
“ “ “

   Return np.multiply( a,b)

def divide(a, b):
   if b == 0:
      return "Error: Division by zero not allowed"
“ “ “
This function helps us to divide a number with other number.it also raises the error if the denominator is
0.

Args(integers): a, b are any two numbers.(b!=0)

Returns(float): division of two numbers.
“ “ “
return np.divide(a,b)
```

```
# Example usage
if __name__ == "__main__":
    x, y = 15, 3

    print("Numbers:", x, "and", y)
    print("Addition:", add(x, y))
    print("Subtraction:", subtract(x, y))
    print("Multiplication:", multiply(x, y))
    print("Division:", divide(x, y))
```

Docstring by AI:

```
ask-3.py > ...
import numpy as np
"""
Task-3.py
A simple calculator module using NumPy for basic arithmetic operations.
Functions:
    add(a, b): Returns the sum of a and b using np.add.
    subtract(a, b): Returns the difference of a and b using np.subtract.
    multiply(a, b): Returns the product of a and b using np.multiply.
    divide(a, b): Returns the quotient of a and b using np.divide. Handles division by zero.
Example:
    add(x, y)         # 18
    subtract(x, y)    # 12
    multiply(x, y)    # 45
    divide(x, y)      # 5.0
"""

# Calculator functions using NumPy
def add(a, b):
    return np.add(a, b)

def subtract(a, b):
    return np.subtract(a, b)

def multiply(a, b):
    return np.multiply(a, b)

def divide(a, b):
    try:
        return np.divide(a, b)
    except ZeroDivisionError:
        return "Error: Division by zero not allowed"
```

```python
# Example usage
if __name__ == "__main__":
    x, y = 15, 3

    print("Numbers:", x, "and", y)
    print("Addition:", add(x, y))
    print("Subtraction:", subtract(x, y))
    print("Multiplication:", multiply(x, y))
    print("Division:", divide(x, y))
```

Comparision:

I gave docstring for each function separately and clearly but AI gave only one docstring .it gave all information about all function clearly.