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.

prompts :

1.Write python function to return sum of even and odd numbers in the given list user defined with out doc string
2.add doc string erasing my docstring

code::

```
def calculate_even_odd_sums(input_list):
    """
     calculates the sum of even and the odd numbers
    input:list of numbers
    output:sum of even numbers,sum of odd numbers

    """
      even_sum = 0
      odd_sum = 0
      for number in input_list:
          try:
              num = int(number)
              if num % 2 == 0:
                  even_sum += num
              else:
                  odd_sum += num
          except ValueError:
              print(f"Warning: Skipping non-integer input: {number}")
              continue
      return even_sum, odd_sum
user_input_str = input("Enter a list of numbers separated by spaces: ")
user_list_str = user_input_str.split()
even_total, odd_total = calculate_even_odd_sums(user_list_str)
print(f"Sum of even numbers: {even_total}")
print(f"Sum of odd numbers: {odd_total}")
```

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

Args:
    numbers: A list of integers.

Returns:
    A tuple containing the sum of even numbers and the sum of odd numbers (even_sum, odd_sum).
  """
  even_sum = 0
  odd_sum = 0
  for number in numbers:
    if number % 2 == 0:
      even_sum += number
    else:
      odd_sum += number
  return even_sum, odd_sum

# Example usage:
my_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
even_total, odd_total = sum_even_odd(my_list)
print(f"Sum of even numbers: {even_total}")
print(f"Sum of odd numbers: {odd_total}")
```

output:

```
Enter a list of numbers separated by spaces: 1 2 3 4 5 6 7 8 9 0
Sum of even numbers: 20
Sum of odd numbers: 25
```

observations and code expalnation

1. Manual docstring is short and simple, while AI docstring is detailed and structured.
2. Manual explains input/output in plain words, AI uses standard Google Style format.
3. Manual is easy to write quickly, AI looks professional and formal.
4. Manual may lack data types and clarity, AI clearly specifies arguments and return values.
5. Manual is suitable for small assignments, AI is better for real-world coding practices.

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.

prompts :

1.Write python program for sru_student class with attributes like name, roll no., hostel_status and fee_update method and display_details method with out comments or doc string user defined without comments and with user input

2.now remove my manually added comments and add your comments

code:

```
class sru_student:
    # Constructor to initialize a new student with basic details
    def __init__(self, name, roll_no, hostel_status):
        # Store the student's name
        self.name = name
        # Store the student's roll number
        self.roll_no = roll_no
        # Store hostel status (Yes/No)
        self.hostel_status = hostel_status
        # Keep track of the total fee paid by the student (initially 0)
        self.fee_paid = 0
    # Method to update the fee details when student makes a payment
    def fee_update(self, amount):
        # Add the new payment amount to the total fee paid
        self.fee_paid += amount
    # Method to display student information in a readable format
    def display_details(self):
        print("Student Details:")
        print("Name:", self.name)
        print("Roll No:", self.roll_no)
        print("Hostel Status:", self.hostel_status)
        print("Fee Paid:", self.fee_paid)
# Take student details as input from the user
user_name = input("Enter student name: ")
user_roll_no = input("Enter student roll number: ")
user_hostel_status = input("Enter hostel status (Yes/No): ")
# Create a student object with the given details
student1 = sru_student(user_name, user_roll_no, user_hostel_status)
# Display the student details using the method
student1.display_details()
```

```python
class sru_student:
    # Class definition for representing a student entity with core attributes and behavioral methods.
    def __init__(self, name, roll_no, hostel_status):
        # Attribute initialization: Assigning provided values to object properties.
        self.name = name  # String attribute for student's identifier (name).
        self.roll_no = roll_no  # String attribute for student's unique registration number.
        self.hostel_status = hostel_status  # String attribute indicating residency status ('Yes'/'No').
        self.fee_paid = 0  # Numeric attribute initialized to zero, representing cumulative fee payments.
    # Method for updating the financial state (fee paid) of the student object.
    def fee_update(self, amount):
        # State modification: Incrementing the 'fee_paid' attribute by the input 'amount'.
        self.fee_paid += amount
    # Method for presenting the current state of the student object's attributes.
    def display_details(self):
        # Output generation: Printing attribute values to standard output in a structured format.
        print("Student Details:")
        print("Name:", self.name)
        print("Roll No:", self.roll_no)
        print("Hostel Status:", self.hostel_status)
        print("Fee Paid:", self.fee_paid)
# Data acquisition: Prompting the user for input to instantiate a student object.
user_name = input("Enter student name: ")
user_roll_no = input("Enter student roll number: ")
user_hostel_status = input("Enter hostel status (Yes/No): ")
# Object instantiation: Creating a new instance of the sru_student class with user-provided data.
student1 = sru_student(user_name, user_roll_no, user_hostel_status)
# Method invocation: Calling the display_details method on the created student object to visualize its sta
student1.display_details()
```

output:

```
Enter student name: vishinu
Enter student roll number: 1120
Enter hostel status (Yes/No): no
Student Details:
Name: vishinu
Roll No: 1120
Hostel Status: no
Fee Paid: 0
```

observations and code expalnation

- Manual comments are conversational, while AI comments are formal and technical.
- Manual focuses on explaining the purpose, AI focuses on describing each step.
- Manual feels natural for beginners, AI feels structured but robotic.
- Manual is better for learning/exams, AI is better for professional projects.
- Manual may miss standardization, AI may miss the “why” behind the code.

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

prompts :

1.Write a Python script with 3–4 functions (e.g., calculator: add, subtract, multiply, with out docstring

2.now remove my doc string and add module level doc string

manual numpy style doc string

```
"""
A simple calculator program that performs basic arithmetic operations
like addition, subtraction, and multiplication.
"""

def add(x, y):
    """
    Add two numbers.

    Parameters
    ----------
    x : int or float
        First number
    y : int or float
        Second number

    Returns
    -------
    int or float
        Sum of x and y
    """
    return x + y


def subtract(x, y):
    """
    Subtract two numbers.

    Parameters
```

```
def subtract(x, y):
    """
    Subtract two numbers.

    Parameters
    ----------
    x : int or float
        First number
    y : int or float
        Second number

    Returns
    -------
    int or float
        Difference (x - y)
    """
    return x - y


def multiply(x, y):
    """
    Multiply two numbers.

    Parameters
    ----------
    x : int or float
        First number
    y : int or float
        Second number
```

```
while True:
    print("Select operation:")
    print("1.Add")
    print("2.Subtract")
    print("3.Multiply")
    print("4.Exit")

    choice = input("Enter choice(1/2/3/4): ")

    if choice in ('1', '2', '3'):
        try:
            num1 = float(input("Enter first number: "))
            num2 = float(input("Enter second number: "))
        except ValueError:
            print("Invalid input. Please enter numbers.")
            continue

        if choice == '1':
            print(num1, "+", num2, "=", add(num1, num2))

        elif choice == '2':
            print(num1, "-", num2, "=", subtract(num1, num2))

        elif choice == '3':
            print(num1, "*", num2, "=", multiply(num1, num2))

    elif choice == '4':
        break
    else:
        print("Invalid Input")
```

AI generated module level doc string

```
"""
This module provides a simple command-line calculator
that supports addition, subtraction, and multiplication.
Users can select an operation and provide two numbers
to perform the calculation. The program continues to run
until the user chooses to exit.
"""

def add(x, y):
    """Return the sum of two numbers."""
    return x + y

def subtract(x, y):
    """Return the difference of two numbers."""
    return x - y

def multiply(x, y):
    """Return the product of two numbers."""
    return x * y


# ---------------- Main Program ----------------

while True:
    print("Select operation:")
    print("1.Add")
    print("2.Subtract")
    print("3.Multiply")
    print("4.Exit")
```

```python
while True:
    print("Select operation:")
    print("1.Add")
    print("2.Subtract")
    print("3.Multiply")
    print("4.Exit")

    choice = input("Enter choice(1/2/3/4): ")

    if choice in ('1', '2', '3'):
        try:
            num1 = float(input("Enter first number: "))
            num2 = float(input("Enter second number: "))
        except ValueError:
            print("Invalid input. Please enter numbers.")
            continue

        if choice == '1':
            print(num1, "+", num2, "=", add(num1, num2))

        elif choice == '2':
            print(num1, "-", num2, "=", subtract(num1, num2))

        elif choice == '3':
            print(num1, "*", num2, "=", multiply(num1, num2))

    elif choice == '4':
        break
    else:
        print("Invalid Input")
```

output :

```
Select operation:
1.Add
2.Subtract
3.Multiply
4.Exit
Enter choice(1/2/3/4): 1
Enter first number: 1
Enter second number: 2
1.0 + 2.0 = 3.0
Select operation:
1.Add
2.Subtract
3.Multiply
4.Exit
Enter choice(1/2/3/4): 4
```

observations and code expalnation :

1. Manual docstrings are detailed and follow NumPy style, while AI docstrings are short and simple.
2. Manual clearly defines parameters, return types, and structure, AI just gives one-line descriptions.
3. Manual looks professional for academic and research projects, AI looks quick and minimal for coding.
4. Manual takes more effort to write but improves readability, AI is faster but less informative.
5. Manual focuses on clarity and standardization, AI focuses on brevity and ease of generation.