]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)	
rCourseCode	24CS002PC215	**CourseTitle**	AI Assisted Coding
Year/Sem	II/I	**Regulation**	R24
Date and Day of Assignment	Week5-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.

Manual docstring:

```python
def sum_even_odd(numbers):  # Function to calculate sum of even and odd numbers
    even_sum = 0  # Initialize sum of even numbers
    odd_sum = 0  # Initialize sum of odd numbers
    for num in numbers:  # Iterate through each number in the list
        if num % 2 == 0:  # Check if the number is even
            even_sum += num  # Add to even sum
        else:  # If the number is odd
            odd_sum += num  # Add to odd sum
    return even_sum, odd_sum # Return the sums as a tuple
```

Ai docstring :

```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
```

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.

Manual docstring:

```
class SRUStudent:
    """
    SRUStudent represents a student at SRU with attributes for personal and hostel details, status, and
    Attributes:
        name (str): The name of the student.
        roll_no (str): The roll number of the student.
        hostel (str): The hostel assigned to the student.
        status (str): The current status of the student (e.g., active, inactive).
        fee (int): The current fee amount for the student.
    Methods:
        __init__(name, roll_no, hostel, status):
            Initializes a new SRUStudent instance with the provided details.
        update_fee(amount):
            Adds the specified amount to the student's fee and prints the updated fee.
        display_details():
            Prints the student's details including name, roll number, hostel, status, and current fee.
    """
    def __init__(self, name, roll_no, hostel, status):
        self.name = name
        self.roll_no = roll_no
        self.hostel = hostel
        self.status = status
        self.fee = 0
```

OUTPUT DEBUG CONSOLE TERMINAL PORTS

Ai docstring :

```
        Prints the student's details including name, roll number, hoste
    """
    def __init__(self, name, roll_no, hostel, status):
        self.name = name
        self.roll_no = roll_no
        self.hostel = hostel
        self.status = status
        self.fee = 0

    def update_fee(self, amount):
        self.fee += amount
        print(f"Fee updated. Current fee: {self.fee}")

    def display_details(self):
        print(f"Name: {self.name}")
        print(f"Roll No: {self.roll_no}")
        print(f"Hostel: {self.hostel}")
        print(f"Status: {self.status}")
        print(f"Fee: {self.fee}")
```

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

```
def add(x, y):#defining functions for arthimetic operations
    return x + y
def subtract(x, y):
    return x - y
def multiply(x, y):
    return x * y
def divide(x, y):

    if y == 0:# handling division by zero
        print("Error: Division by zero is not allowed.")
        return
    else:
        return x / y
# Example usage
num1 = 10
num2 = 5
print("Addition:", add(num1, num2))#calling functions and printing results
print("Subtraction:", subtract(num1, num2))
print("Multiplication:", multiply(num1, num2))
print("Division:", divide(num1, num2))
print("Division by zero:", divide(num1, 0))
```

Ai docstring :

```
def add(x, y):#defining functions for arthimetic operations
    """
    Performs basic arithmetic operations.

    Functions:
        add(x, y): Returns the sum of x and y.
        subtract(x, y): Returns the difference of x and y.
        multiply(x, y): Returns the product of x and y.
        divide(x, y): Returns the quotient of x divided by y.
    """
    return x + y
def subtract(x, y):
    return x - y
def multiply(x, y):
    return x * y
"""
Divides two numbers and handles division by zero.
Parameters:
    x (float or int): The numerator.
    y (float or int): The denominator.
Returns:
    float: The result of dividing x by y if y is not zero.
    None: If y is zero, prints an error message and returns None.
"""
```

MS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

Users > kanishka > Downloads > aiac 3.py > add

```
    Divides two numbers and handles division by zero.
    Parameters:
        x (float or int): The numerator.
        y (float or int): The denominator.
    Returns:
        float: The result of dividing x by y if y is not zero.
        None: If y is zero, prints an error message and returns None.
    """
    if y == 0:# handling division by zero
            print("Error: Division by zero is not allowed.")
            return
        else:
            return x / y
    # Example usage
    num1 = 10
    num2 = 5
    print("Addition:", add(num1, num2))#calling functions and printing results
    print("Subtraction:", subtract(num1, num2))
    print("Multiplication:", multiply(num1, num2))
    print("Division:", divide(num1, num2))
    print("Division by zero:", divide(num1, 0))
```

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