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 9: 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., Gemini, Copilot, Cursor AI) to generate a docstring describing the function.
- Compare the AI-generated docstring with your manually written one.

- **Prompt : Write a Python function to return the sum of even and odd numbers in a list and generate a Google-style docstring for it.**

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

    Args:
        numbers: A list of numbers.

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

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

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

```
PS C:\Users\T-SHIRISHA\OneDrive\Documents\AIAC> & C:\Users\T-
y
Sum of even numbers: 30
Sum of odd numbers: 25
PS C:\Users\T-SHIRISHA\OneDrive\Documents\AIAC>
```

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.

- **Prompt : Write a Python program for a sru_student class with attributes and methods, and generate inline comments explaining each line of code.**

```
count.py > ...
class sru_student:
    """
    Represents a student at SRU.
    """
    def __init__(self, name, roll_no, hostel_status):
        """
        Initializes a new sru_student object.
        Args:
            name (str): The name of the student.
            roll_no (str): The roll number of the student.
            hostel_status (bool): True if the student is a hosteller, False otherwise.
        """
        self.name = name # Assign the provided name to the student object's name attribute
        self.roll_no = roll_no # Assign the provided roll number to the student object's roll_no attribute
        self.hostel_status = hostel_status # Assign the provided hostel status to the student object's hostel_status attribute
        self.fees_paid = 0 # Initialize the fees_paid attribute to 0 for a new student

    def fee_update(self, amount):
        """
        Updates the fee paid by the student.
        Args:
            amount (float): The amount of fee paid.
        """
        if amount > 0: # Check if the fee amount is positive
            self.fees_paid += amount # Add the paid amount to the total fees paid
            print(f"Fee updated for {self.name}. Total fees paid: {self.fees_paid}") # Print a confirmation message
        else:
            print("Invalid amount. Fee update failed.") # Print an error message if the amount is not positive


    def display_details(self):
        """
        Displays the details of the student.
        """
        print("Student Details:") # Print a header for the student details
        print(f"Name: {self.name}") # Print the student's name
        print(f"Roll No: {self.roll_no}") # Print the student's roll number
```

```
        print(f"Roll No: {self.roll_no}") # Print the student's roll number
        # Print the hostel status, displaying "Hosteller" or "Day Scholar" based on the boolean value
        print(f"Hostel Status: {'Hosteller' if self.hostel_status else 'Day Scholar'}")
        print(f"Fees Paid: {self.fees_paid}") # Print the total fees paid by the student

# Example usage:
student1 = sru_student("Alice", "SRU123", True) # Create an instance of the sru_student class for Alice
student1.display_details() # Display the details for student1
student1.fee_update(5000) # Update the fees paid for student1
student1.display_details() # Display the updated details for student1

student2 = sru_student("Bob", "SRU456", False) # Create an instance of the sru_student class for Bob
student2.display_details() # Display the details for student2
student2.fee_update(3000) # Update the fees paid for student2
student2.fee_update(2000) # Update the fees paid for student2 again
student2.display_details() # Display the updated details for student2
```

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

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

Name: Alice
Roll No: SRU123
Hostel Status: Hosteller
Fees Paid: 0
Fee updated for Alice. Total fees paid: 5000
Student Details:
Name: Alice
Roll No: SRU123
Hostel Status: Hosteller
Fees Paid: 5000
Student Details:
Name: Bob
Roll No: SRU456
Hostel Status: Day Scholar
Fees Paid: 0
Fee updated for Bob. Total fees paid: 3000
Fee updated for Bob. Total fees paid: 5000
Student Details:
Name: Bob
Roll No: SRU456
Hostel Status: Day Scholar
Fees Paid: 5000
PS C:\Users\T-SHIRISHA\OneDrive\Documents\AIAC>
```

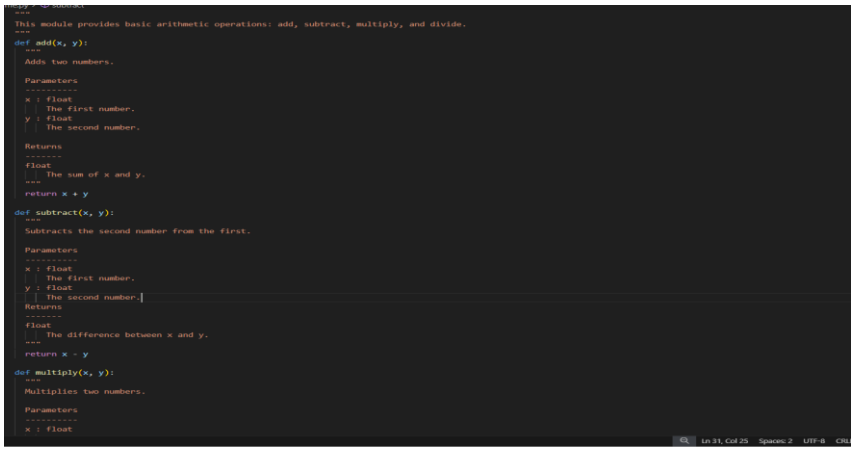
Task Description#3

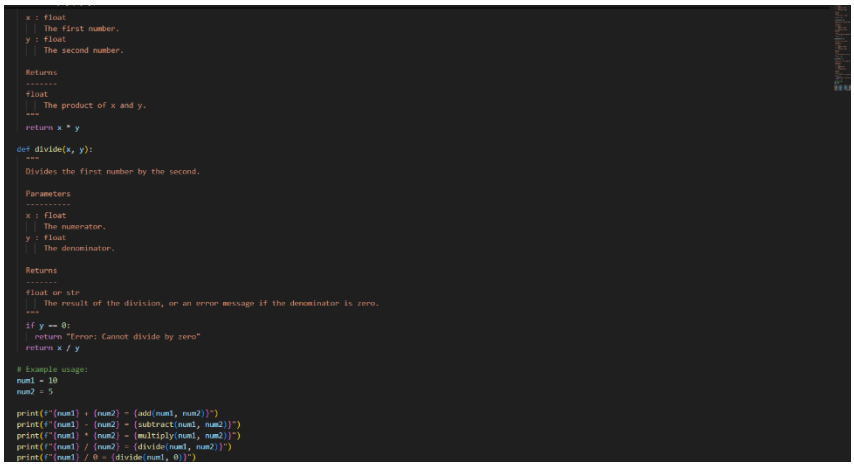
- Write a Python script with 3–4 functions (e.g., calculator: add, subtract, multiply, divide).
- Incorporate manual **docstring** in code with NumPy Style
- Use AI assistance to generate a module-level docstring + individual function

docstrings.

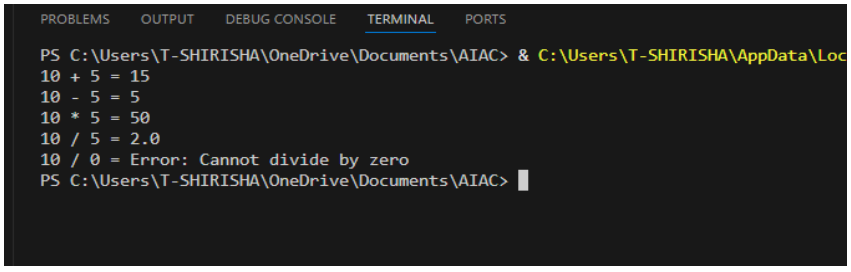
- Compare the AI-generated docstring with your manually written one.

- **Prompt : Write a Python calculator script with add, subtract, multiply, and divide functions, and create NumPy-style docstrings for the module and each function.**





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