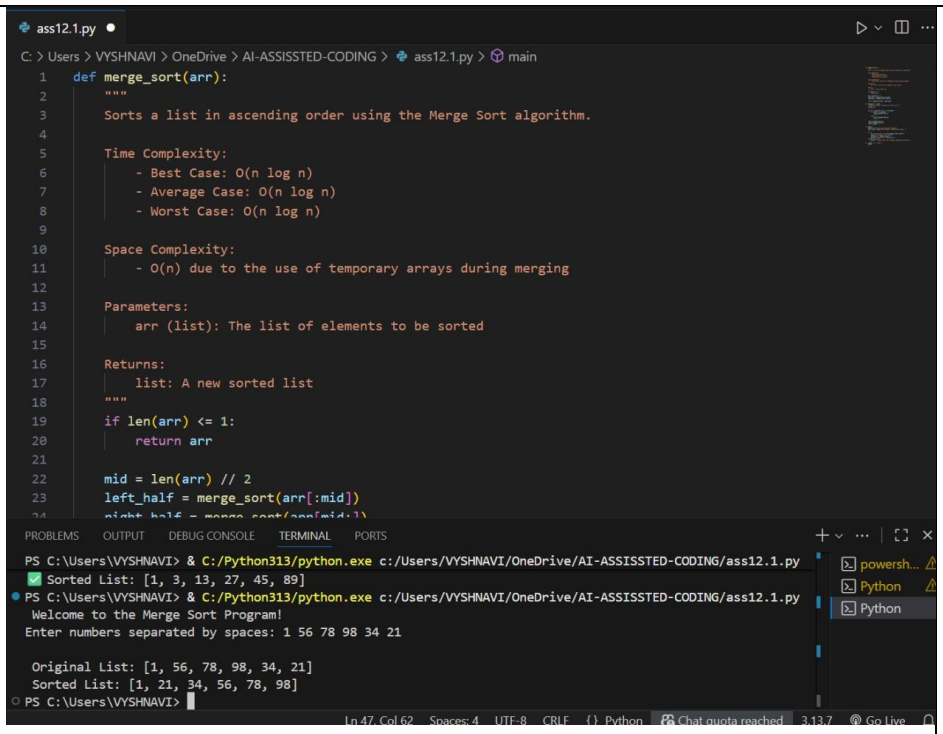
SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE		DEPARTMENT OF COMPUTER SCIENCE ENGINEERING	
Program Name: B. Tech		**Assignment Type: Lab**	**Academic Year:**2025-2026
Course Coordinator Name		Venkataramana Veeramsetty	
Instructor(s) Name		Dr. V. Venkataramana (Co-ordinator) Dr. T. Sampath Kumar 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)	
Course Code	24CS002PC215	**Course Title**	AI Assisted Coding
Year/Sem	II/I	**Regulation**	R24
Date and Day of Assignment	Week6 - Monday	**Time(s)**	
Duration	2 Hours	**Applicable to Batches**	
AssignmentNumber:12.1(Present assignment number)/**24**(Total number of assignments)			

Q.No.	Question	*Expected Time to complete*
1	**Lab 12: Algorithms with AI Assistance – Sorting, Searching, and Optimizing Algorithms** **Lab Objectives:** • Apply AI-assisted programming to implement and optimize sorting and searching algorithms. • Compare different algorithms in terms of efficiency and use	Week6 - Monday

	cases. • Understand how AI tools can suggest optimized code and complexity improvements.	
	Task Description #1 (Sorting – Merge Sort Implementation) • Task: Use AI to generate a Python program that implements the Merge Sort algorithm. • Instructions: o Prompt AI to create a function merge_sort(arr) that sorts a list in ascending order. o Ask AI to include time complexity and space complexity in the function docstring. o Verify the generated code with test cases. • Expected Output: o A functional Python script implementing Merge Sort with proper documentation. **PROMPT:** Write a Python program using Merge Sort to sort a list of numbers. Include a function merge_sort(arr) with a docstring that explains: – What the function does – Time complexity (best, average, worst) – Space complexity Add 3 test cases and print the sorted results. **SCREENSHOT:**	



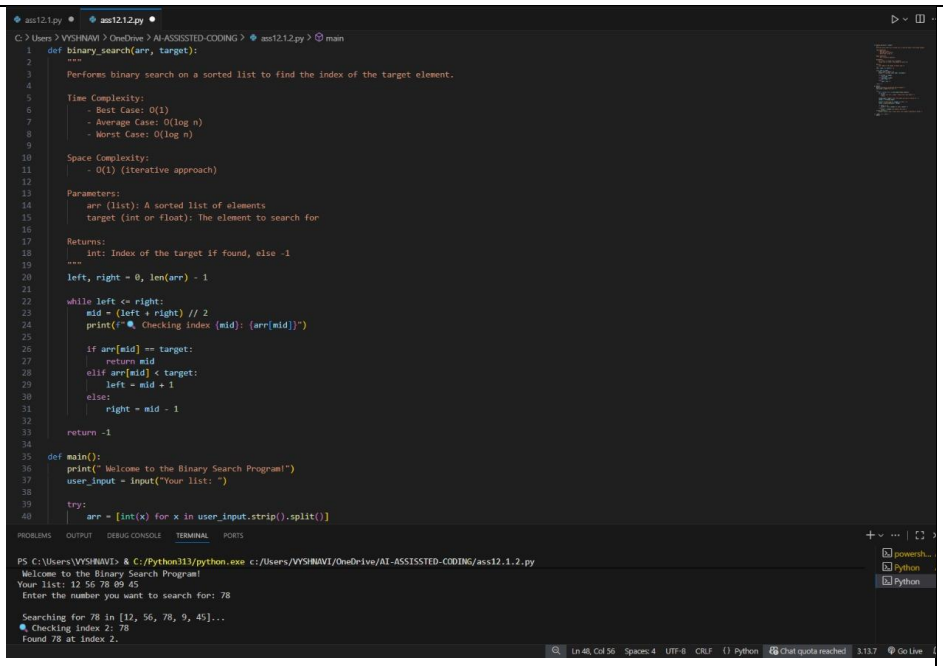
Task Description #2 (Searching – Binary Search with AI Optimization)

- Task: Use AI to create a binary search function that finds a target element in a sorted list.
- Instructions:
  - Prompt AI to create a function binary_search(arr, target) returning the index of the target or -1 if not found.
  - Include docstrings explaining best, average, and worst-case complexities.
  - Test with various inputs.
- Expected Output:
  - Python code implementing binary search with AI-generated comments and docstrings.

PROMPT:

"Write a Python program that asks the user for a sorted list of numbers and a number to search for. Use binary search to find the number and print its index or say it's not found. Include comments and a short explanation of how the code works."

SCREENSHOT:



```
def binary_search(arr, target):
    """
    Performs binary search on a sorted list to find the index of the target element.

    Time Complexity:
        - Best Case: O(1)
        - Average Case: O(log n)
        - Worst Case: O(log n)

    Space Complexity:
        - O(1) (iterative approach)

    Parameters:
        arr (list): A sorted list of elements
        target (int or float): The element to search for

    Returns:
        int: Index of the target if found, else -1
    """
    left, right = 0, len(arr) - 1

    while left <= right:
        mid = (left + right) // 2
        print(f"🔍 Checking index {mid}: {arr[mid]}")

        if arr[mid] == target:
            return mid
        elif arr[mid] < target:
            left = mid + 1
        else:
            right = mid - 1

    return -1

def main():
    print("🔍 Welcome to the Binary Search Program!")
    user_input = input("Your list: ")

    try:
        arr = [int(x) for x in user_input.strip().split()]
```

```
PS C:\Users\VYSHNAVI> & C:/Python313/python.exe c:/Users/VYSHNAVI/OneDrive/AI-ASSISSTED-CODING/ass12.1.2.py
 Welcome to the Binary Search Program!
Your list: 12 56 78 09 45
 Enter the number you want to search for: 78

 Searching for 78 in [12, 56, 78, 9, 45]...
🔍 Checking index 2: 78
Found 78 at index 2.
```

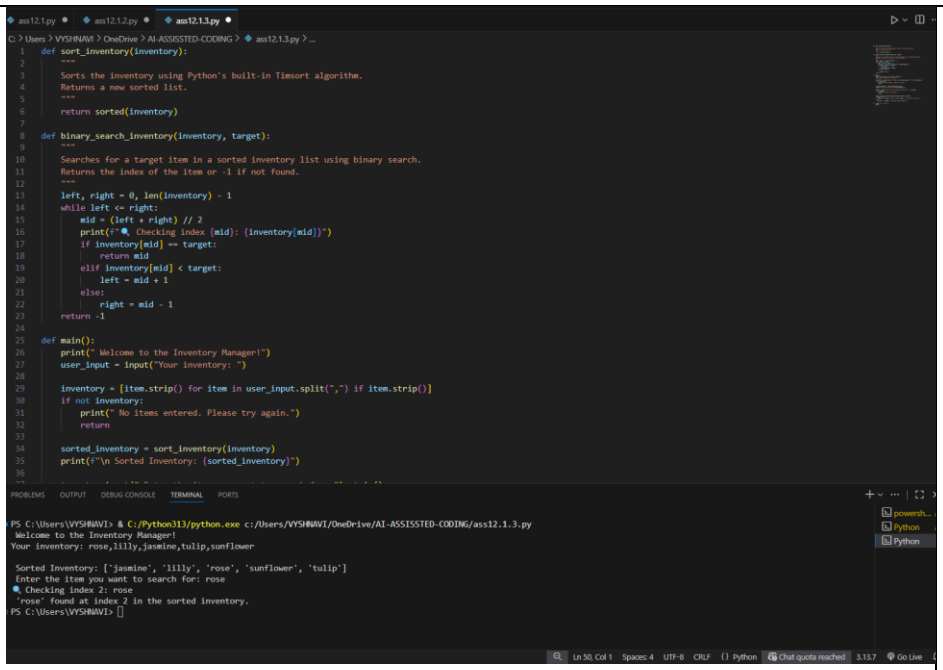
Task Description #3 (Real-Time Application – Inventory Management System)

- Scenario: A retail store's inventory system contains thousands of products, each with attributes like product ID, name, price, and stock quantity. Store staff need to:
    1. Quickly search for a product by ID or name.
    2. Sort products by price or quantity for stock analysis.
- Task:
    - Use AI to suggest the most efficient search and sort algorithms for this use case.
    - Implement the recommended algorithms in Python.
    - Justify the choice based on dataset size, update frequency, and performance requirements.
- Expected Output:
    - A table mapping operation → recommended algorithm → justification.
    - Working Python functions for searching and sorting the inventory.

PROMPT:

"Create a Python program that lets the user enter a list of inventory items, sorts them, and searches for a specific item using binary search. The program should be easy to use, show clear messages, and print the index of the item if found."

SCREENSHOT:



```
ass12.1.py ● ass12.1.2.py ● ass12.1.3.py ●
C: > Users > VYSHNAVI > OneDrive > AI-ASSISSTED-CODING > ass12.1.3.py > ...
def sort_inventory(inventory):
    """
    Sorts the inventory using Python's built-in Timsort algorithm.
    Returns a new sorted list.
    """
    return sorted(inventory)

def binary_search_inventory(inventory, target):
    """
    Searches for a target item in a sorted inventory list using binary search.
    Returns the index of the item or -1 if not found.
    """
    left, right = 0, len(inventory) - 1
    while left <= right:
        mid = (left + right) // 2
        print(f"🔍 Checking index {mid}: {inventory[mid]}")
        if inventory[mid] == target:
            return mid
        elif inventory[mid] < target:
            left = mid + 1
        else:
            right = mid - 1
    return -1

def main():
    print("📦 Welcome to the Inventory Manager!")
    user_input = input("Your inventory: ")

    inventory = [item.strip() for item in user_input.split(",") if item.strip()]
    if not inventory:
        print("⚠️ No items entered. Please try again.")
        return

    sorted_inventory = sort_inventory(inventory)
    print(f"\n Sorted Inventory: {sorted_inventory}")

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
PS C:\Users\VYSHNAVI> & C:/Python313/python.exe c:/Users/VYSHNAVI/OneDrive/AI-ASSISSTED-CODING/ass12.1.3.py
📦 Welcome to the Inventory Manager!
Your inventory: rose,lilly,jasmine,tulip,sunflower

 Sorted Inventory: ['jasmine', 'lilly', 'rose', 'sunflower', 'tulip']
 Enter the item you want to search for: rose
🔍 Checking index 2: rose
 'rose' found at index 2 in the sorted inventory.
PS C:\Users\VYSHNAVI>
```

✅ Deliverables (For All Tasks)

1. AI-generated prompts for code and test case generation.
2. At least 3 assert test cases for each task.
3. AI-generated initial code and execution screenshots.
4. Analysis of whether code passes all tests.
5. Improved final version with inline comments and explanation.
6. Compiled report (Word/PDF) with prompts, test cases, assertions, code, and output.