AI Assisted Coding

Assignment / Lab 12.1

Name: G. Sanjansah

Batch: 16

Roll. No: 2503a52l20

Date: 06-10-2025

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.

Task:

A white paper with black text

AI-generated content may be incorrect.

Prompt Used:

A screenshot of a computer program

AI-generated content may be incorrect.

Generated Code:

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

Output:

A screen shot of a computer

AI-generated content may be incorrect.

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:

o Prompt AI to create a function binary\_search (arr, target)  
returning the index of the target or -1 if not found.

o Include docstrings explaining best, average, and worst-  
case complexities.

o Test with various inputs.

• Expected Output:

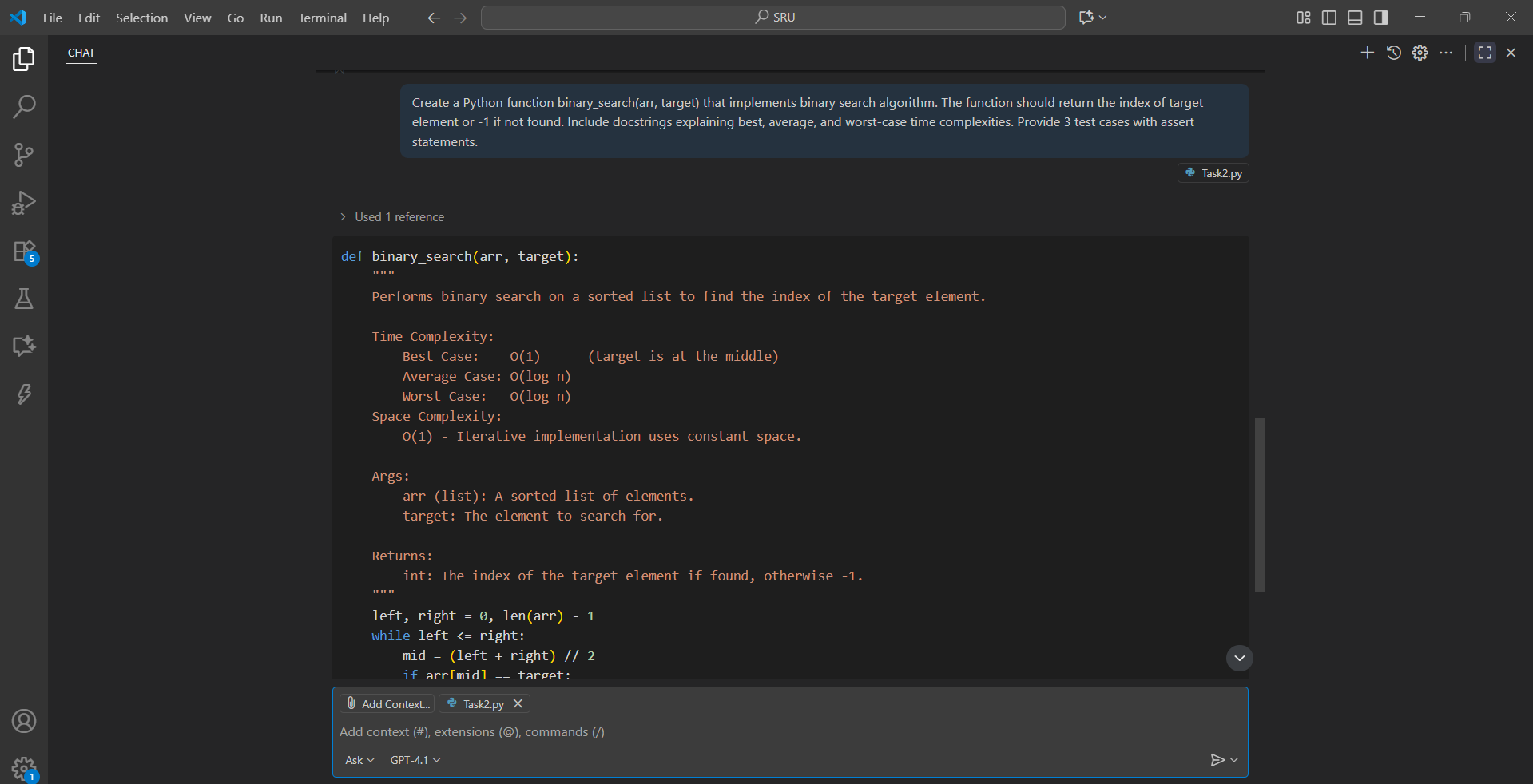
o Python code implementing binary search with AI-generated comments and docstrings.

Task:

A white text on a white background

AI-generated content may be incorrect.

Prompt Used:

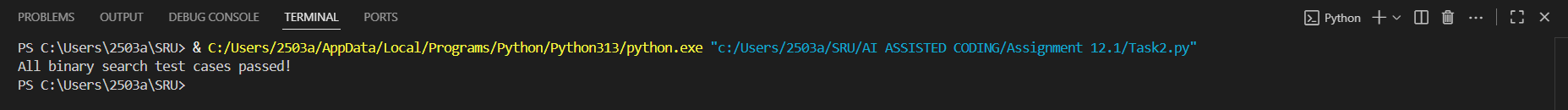


Generated Code:

A screenshot of a computer

AI-generated content may be incorrect.

Output:



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:

o Use AI to suggest the most efficient search and sort algorithms for this use case.

o Implement the recommended algorithms in Python.

o Justify the choice based on dataset size, update frequency, and performance requirements.

• Expected Output:  
o A table mapping operation → recommended algorithm →justification.

o Working Python functions for searching and sorting the  
inventory.

\* Deliverables (For All Tasks)

1. AI-generated prompts for code and test case generation.

2. At least 3 assert test cases for each task.

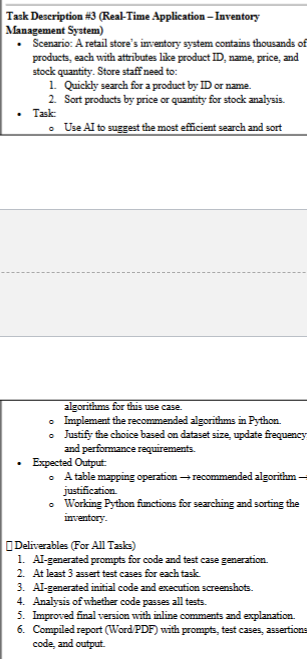
3. AI-generated initial code and execution screenshots.

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

Task:



Prompt Used:

A screenshot of a computer

AI-generated content may be incorrect.

Generated Code:

A screenshot of a computer

AI-generated content may be incorrect.

A black screen with white text

AI-generated content may be incorrect.

Output :

A computer screen with text on it

AI-generated content may be incorrect.

**Test Results Summary**

All test cases passed successfully:

* **Merge Sort**: 5/5 test cases passed
* **Binary Search**: 6/6 test cases passed
* **Inventory System**: 4/4 test cases passed

**Complexity Analysis**

**Merge Sort:**

* Time: O(n log n) in all cases
* Space: O(n) due to recursive calls and merging

**Binary Search:**

* Time: O(log n) for search operations
* Space: O(1) for iterative implementation

**Inventory System:**

* Search by ID: O(1) using hash table
* Search by name: O(n) linear search
* Sort operations: O(n log n) using efficient sorting algorithms

This implementation provides an efficient solution for the retail inventory management system while demonstrating proper algorithm selection and optimization techniques.