SCHOOLOFCOMPUTERSCIENCEANDARTIFICIALINTELLIGENCE		DEPARTMENTOFCOMPUTERSCIENCEENGINEERING	
ProgramName: B.Tech		**AssignmentType:Lab**	**AcademicYear:** 2025-2026
CourseCode	23CS002PC304	**CourseTitle**	AIAssistedCoding
Year/Sem	III/II	**Regulation**	R23
DateandDay of Assignment	**Week3 –Monday**	**Batch**	23CSBTB48
Name	B.SUSHMA	**HallTicket No**	2303A54040

AssignmentNumber:5.1

Q.No.	Question

Lab5:Ethical Foundations –Responsible AI Coding Practices Lab

Objectives:

- To explore the ethical risk sassociated with AI-generated code.
- To recognize issues related to security, bias , transparency, and copyright.
- To reflect on the responsibilities of developers when using AI tools in software development.
- To promote awareness of best practices for responsible and ethical AIcoding.

Lab Outcomes (LOs):

After completing this lab,students will be able to:

- Identify and avoid in secure coding patterns generated by AI tools.
- Detect and analyze potential bias or discriminatory logic in AI-generated outputs.
- Evaluate originality and licensing concerns in reused AI-generated code.
- Understand the importance of explainability and transparency in AI-assisted programming.
- Reflect on accountability and the human role in ethical AIcoding practices.

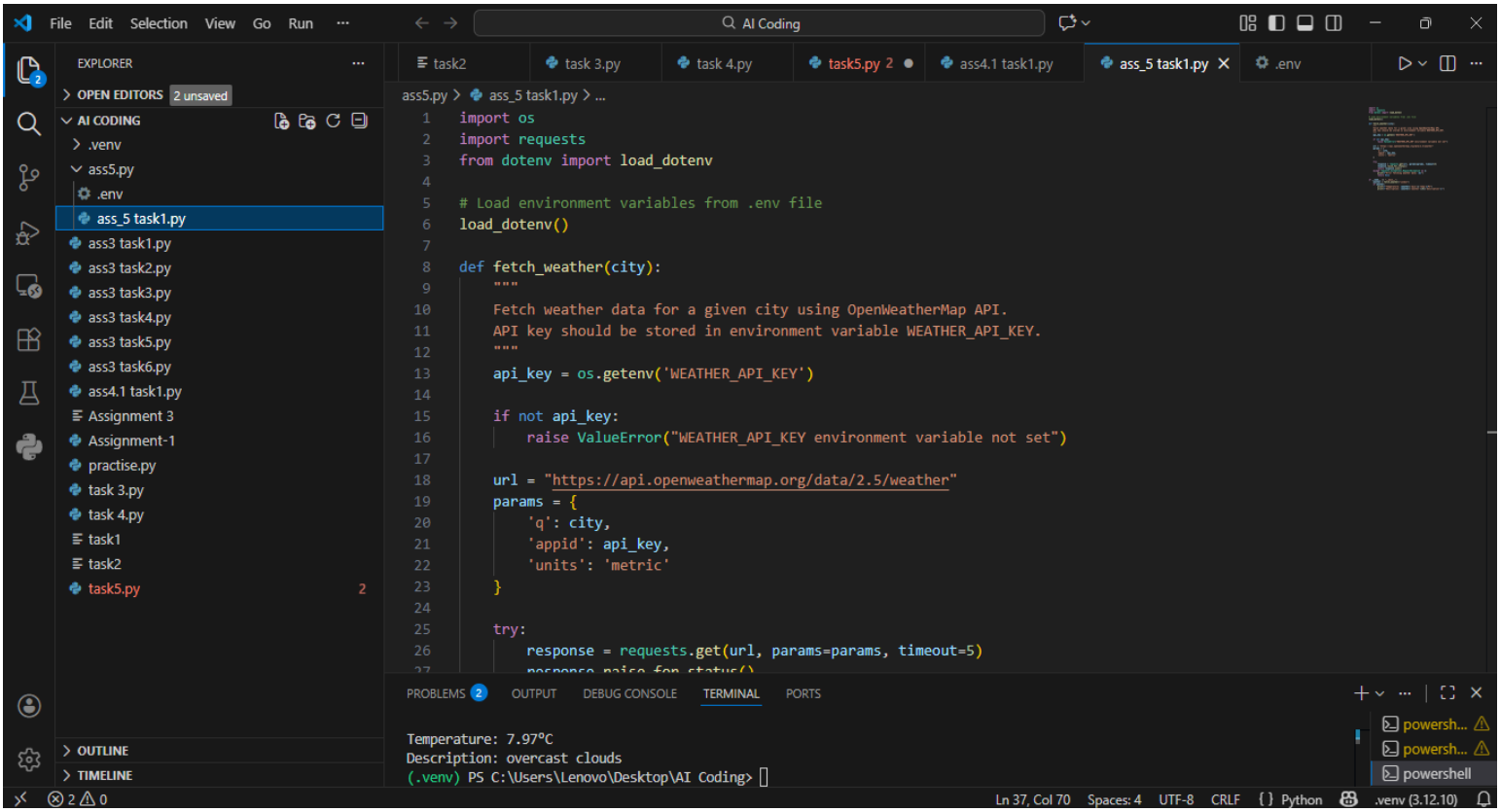
TaskDescription#1(PrivacyinAPIUsage)

Task:Use an AI tool to generate a Python program that connects to a weather API.

Prompt:
"Write a Python program using an AI tool to securely fetch weather data from an API without exposing sensitive API keys in the code."

Expected Output:

- Original AI code(check if keys are hard coded).
- Secure version using environment variables.



EXPLANATION : From this task, I understood that while using APIs in programs, **API keys should never be hardcoded** directly into the source code. If API keys are written openly in the program, anyone who accesses the file can misuse them, which can lead to security and privacy issues.This task helped me learn that a **secure approach is to store API keys in environment variables** and access them during runtime. This way, sensitive credentials remain hidden, and even if the code is shared or uploaded to a repository, the API key is protected. As a developer, it is my responsibility to ensure that confidential information is handled securely.

TaskDescription#2(Privacy & Security in File Handling)

Task:Use an AI tool to generate a Python script that stores user data(name,email,password)in a file.

Analyze: Check if the AI stores sensitive data in plain text or without encryption.

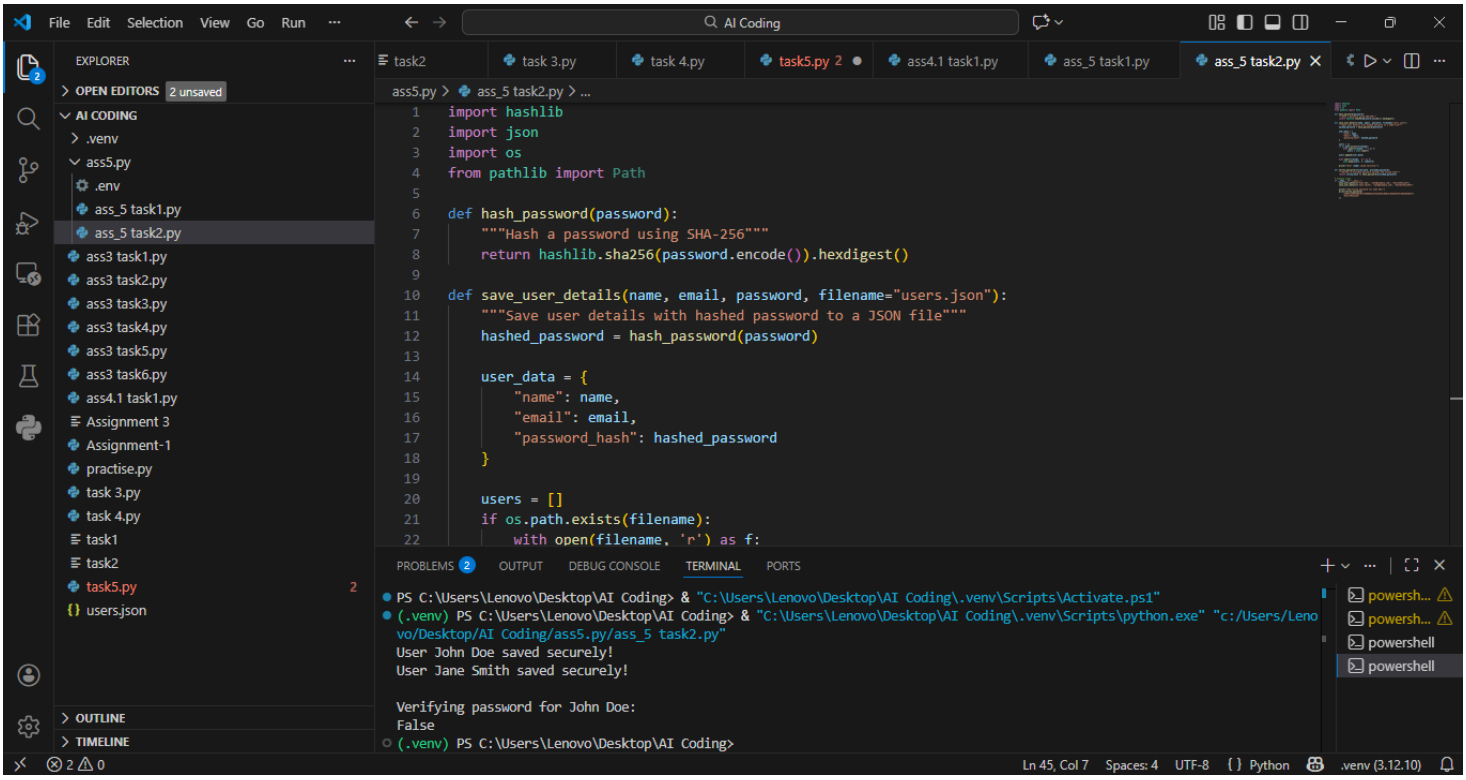
Expected Output:

- Identified privacy risks.
- Revised version with encrypted password storage (e.g.,hashing).

Prompt:

"Create a Python script using an AI tool to save user information (name, email, password) in a file. Evaluate the security of the password storage and update the script to store the password securely using hashing."

Code :



EXPLANATION : From this task, I learned that storing sensitive user information such as passwords in **plain text format is highly insecure**. If such files are accessed by unauthorized users, passwords can be easily read and misused, leading to serious security breaches.This task taught me the importance of using **password hashing techniques** instead of storing actual passwords. By converting passwords into hashed values with added salt, the original password cannot be retrieved even if the file is compromised. This ensures better privacy and protects users' data, highlighting the importance of secure coding practices when handling personal information.

Task Description #3 (Transparency in Algorithm Design)

Objective : Use AI to generate an Armstrong number checking function with comments and explanations.

Instructions:

1. Ask AI to explain the codeline – by - line.
2. Compare the explanation with code

functionality.

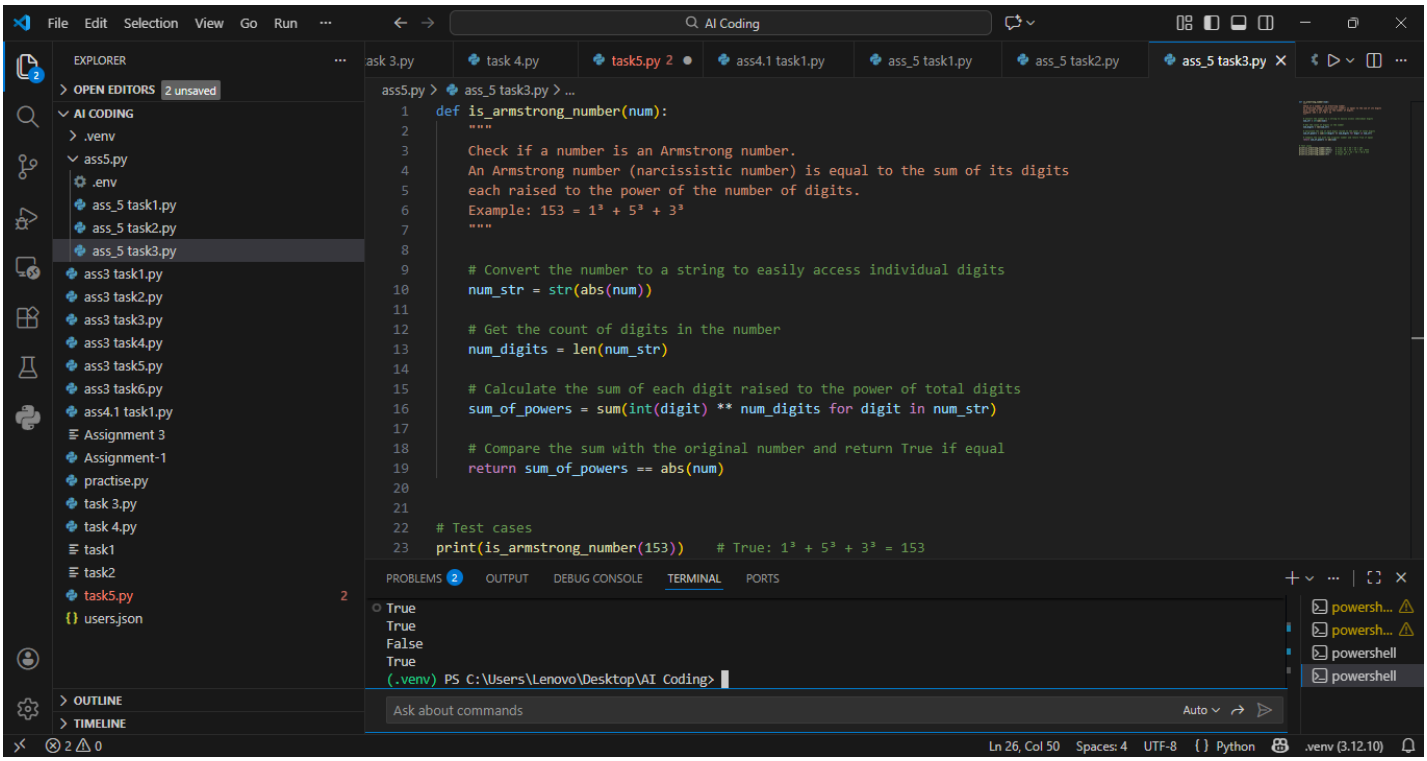
3. Expected Output:

- Transparent , commented code.
- Correct , easy-to-understand explanation.

Prompt:

"Ask an AI tool to generate a Python function that checks if a number is an Armstrong number. Include detailed comments explaining each line of the code and verify if the explanation matches the functionality."

Code :



EXPLANATION : From this task, I understood the importance of **transparency and explainability** in AI-generated code. Simply generating working code is not enough; the logic behind each step must be clear and understandable to humans.By generating an Armstrong number program with line-by-line comments, I learned how each part of the code contributes to the final result. This task helped me verify that the explanation correctly matches the actual functionality of the code. Clear explanations make the program easier to debug, maintain, and trust, especially when AI tools are used in development.

Task Description #4 (Transparency in Algorithm Comparison)

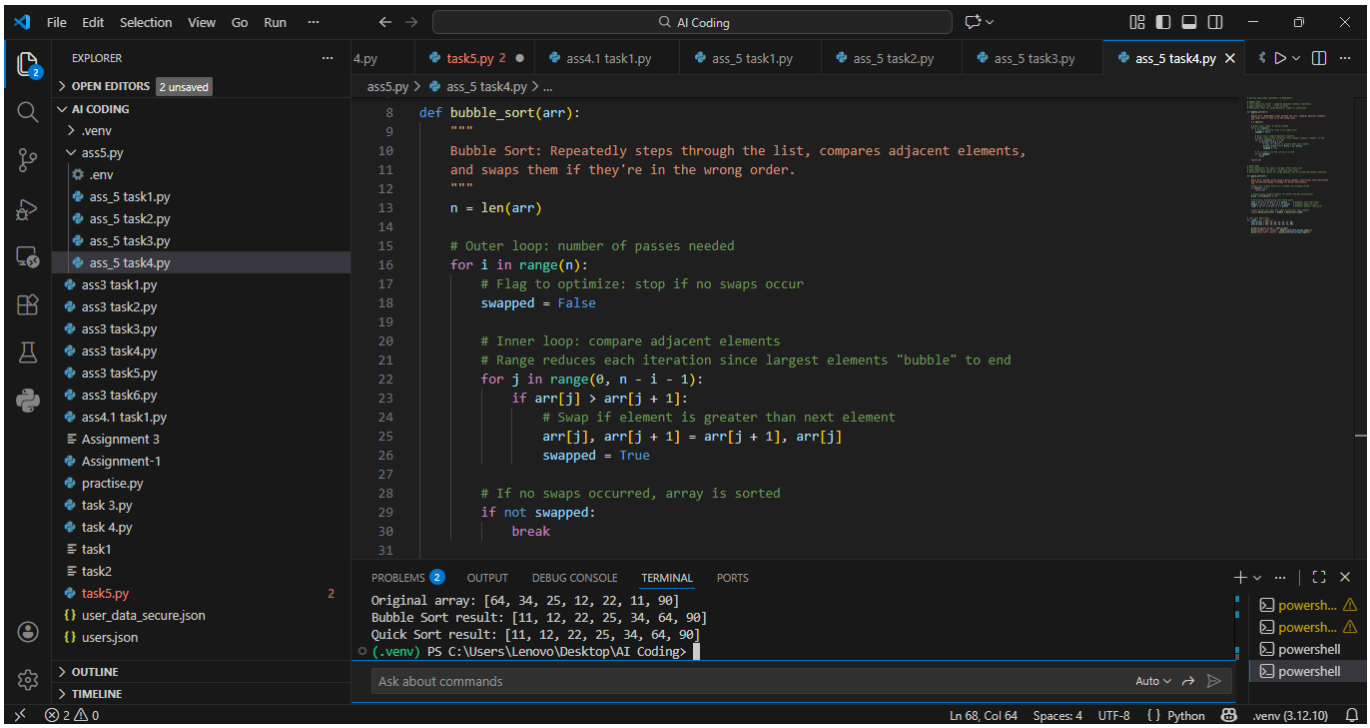
Task:Use AI to implement two sorting algorithms(e.g.,Quick Sort and Bubble Sort).

Prompt: "Use an AI tool to implement two sorting algorithms, QuickSort and BubbleSort, in Python. Add comments to explain each step and highlight the differences in their logic and efficiency."

Expected Output:

- Code for both algorithms.
- Transparent , comparative explanation of their logic and efficiency.

- Codes :



EXPLANATION : From this task, I learned that different algorithms can solve the same problem but with **different efficiency and performance**. Bubble Sort is simple and easy to understand, but it is inefficient for large datasets because it repeatedly compares adjacent elements.On the other hand, Quick Sort uses a **divide-and-conquer approach**, making it much faster and more efficient for large inputs. By comparing both algorithms with explanations, I understood how transparency in algorithm design helps developers choose the right solution based on performance requirements and use cases.

Task Description #5 (Transparency in AIR ecommendations)

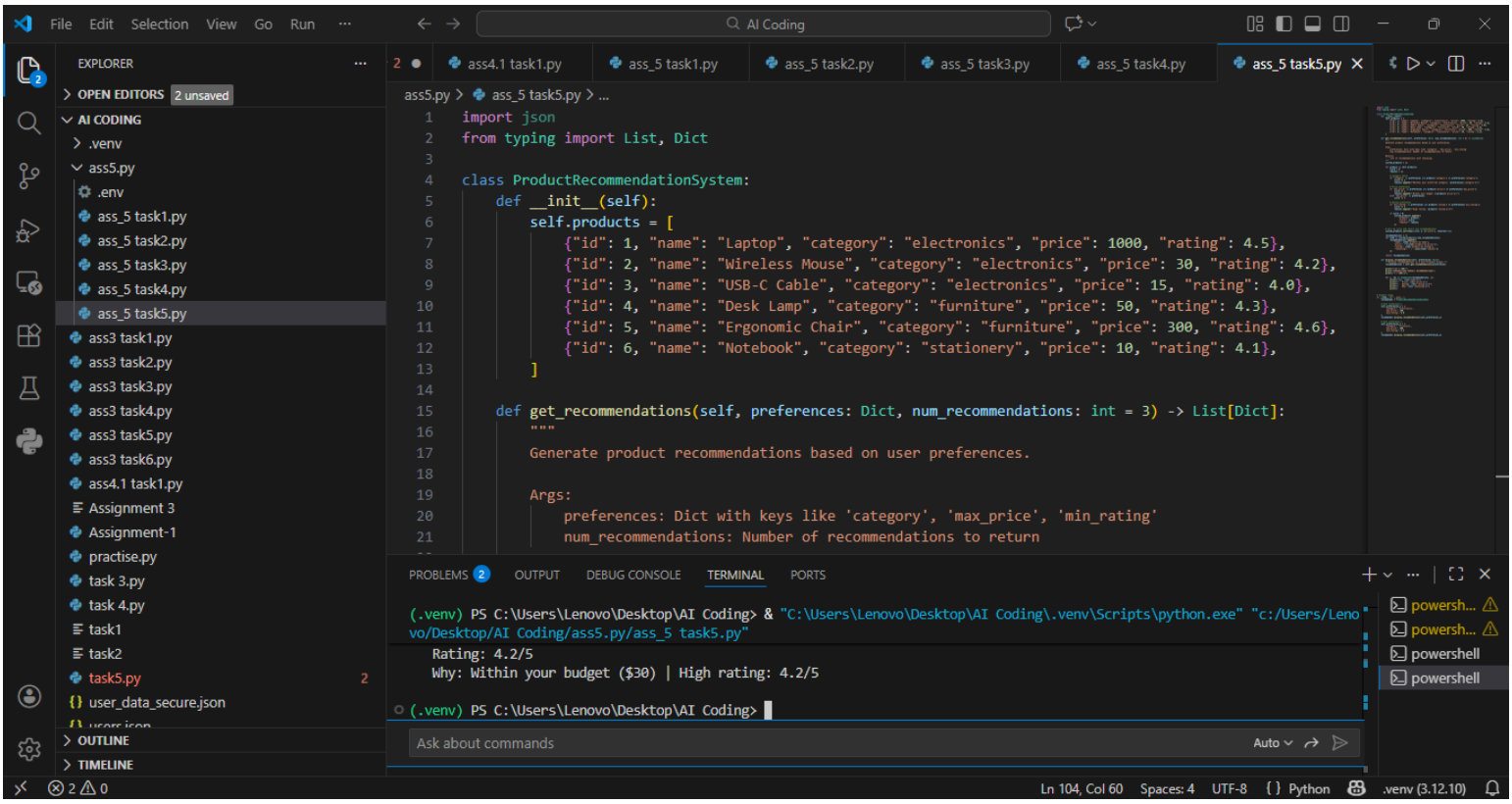
Task:Use AI to create a product recommendation system.

Prompt: "Develop a product recommendation system using an AI tool that provides suggestions based on user preferences. Ensure the system explains the reasoning behind each recommendation."

Expected Output:

- Code with explainable recommendations.
- Evaluation of whether explanations are understandable.

Code :



EXPLANATION : From this task, I understood that AI recommendation systems should not only suggest items but also **explain why a particular recommendation was made**. Without explanations, users may not trust or understand AI decisions.This task helped me realize the importance of **explainable AI**, where recommendations are based on user preferences, past behavior, or similar users' choices. Providing reasons improves transparency, builds user trust, and allows users to better understand how AI systems work, making AI-assisted applications more reliable and ethical.