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	Week11 - Monday	**Time(s)**	
Duration	2 Hours	**Applicable to Batches**	
AssignmentNumber:21.1(Present assignment number)/**24**(Total number of assignments)			

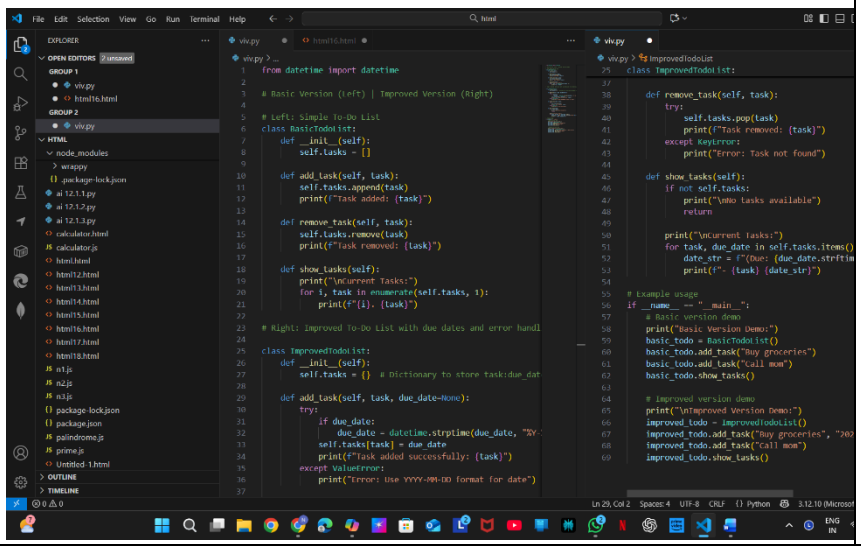
Q.No.	Question	*Expected Time to complete*
1	**Lab 21 – Collaborative AI Coding: Pair Programming and Version Control Integration** **Lab Objectives:** • Learn to use AI as a **pair programmer** for writing, reviewing, and optimizing code. • Understand how to integrate AI-generated code into **version**	Week11 - Monday

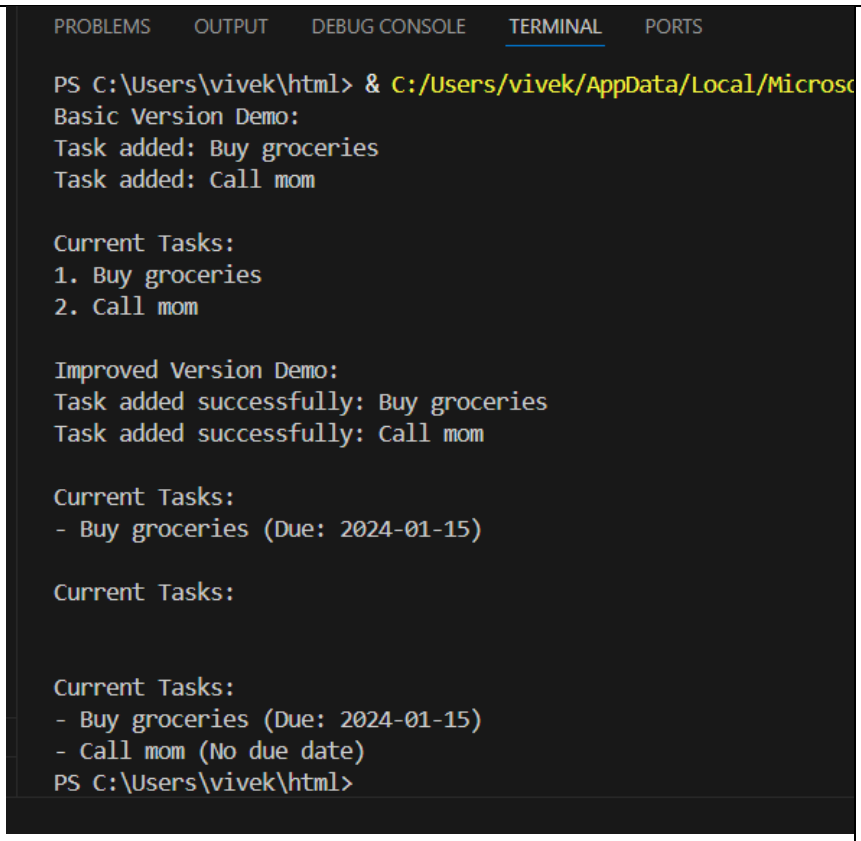
control systems (Git).

- Practice collaborative workflows such as **commit messages, branching, and merging**.

Task 1 – AI-Assisted Pair Programming

- Use AI to generate an initial program (e.g., a simple To-Do List Manager).
- Ask AI to suggest **enhancements** (such as error handling, additional features, or refactoring).
- Compare the original and improved versions to understand the role of AI in collaboration.

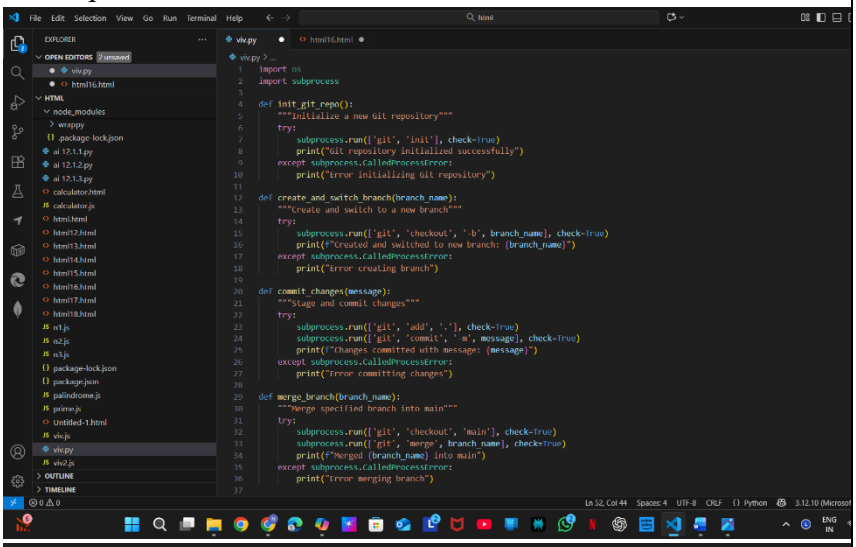




- Explanation: Defines two to-do implementations: BasicTodoList (uses a list) and ImprovedTodoList (uses a dict mapping task -> due_date).
- BasicTodoList: add_task/appends, remove_task/removes (no error handling), show_tasks prints numbered list.
- ImprovedTodoList: add_task accepts optional due_date string, parses YYYY-MM-DD to datetime and stores it; prints error on invalid date.
- Improved remove_task catches missing tasks and prints an error; show_tasks lists tasks with formatted due dates or "No due date".
- If run as **main**, the script demos both versions by adding tasks and printing their contents.

Task 2 – Version Control Integration

- Initialize a Git repository for your project.
- Perform the following steps:
    1. Create and switch to a new branch.
    2. Add AI-assisted modifications.
    3. Commit with a meaningful message.
    4. Merge changes back into the main branch.

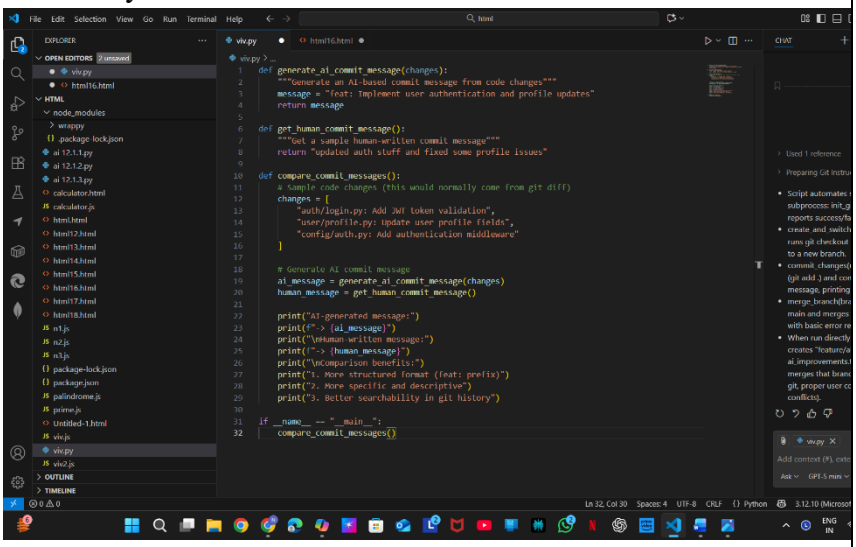
- Reflect on how version control supports collaborative development with AI assistance.
- 

- 

- Explanation: Script automates simple Git tasks via subprocess: init_git_repo() runs git init and reports success/failure.
- create_and_switch_branch(branch_name) runs git checkout -b to create and switch to a new branch.
- commit_changes(message) stages all files (git add .) and commits with the provided message, printing an error on failure.
- merge_branch(branch_name) checks out main and merges the given branch into it, with basic error reporting.
- When run directly it initializes a repo, creates "feature/ai-improvements", writes ai_improvements.txt, commits it, then merges that branch into main (requires git, proper user config, and may fail on conflicts).
-

Task 3 – AI for Commit Message Generation

- Use AI to generate **concise, meaningful commit messages** based on the changes made.
- Compare AI-generated messages with human-written ones.
- Discuss how AI can improve **documentation and traceability** in Git history.

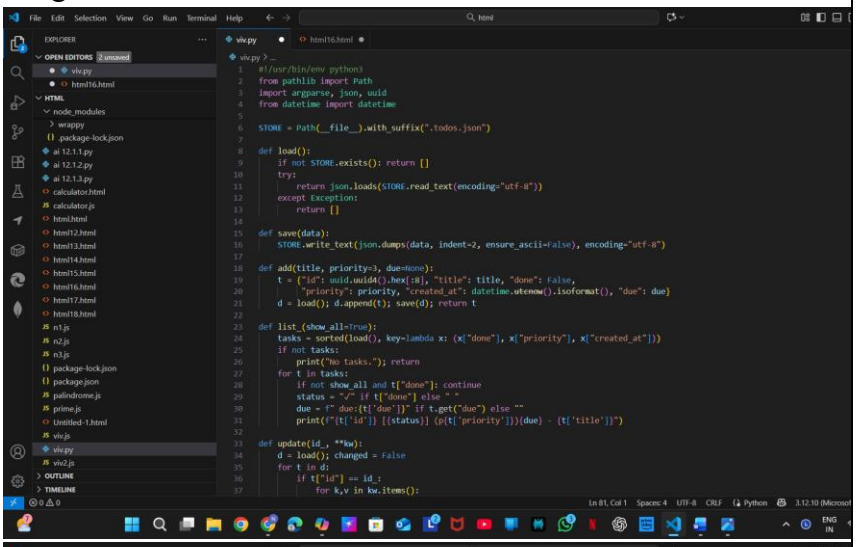


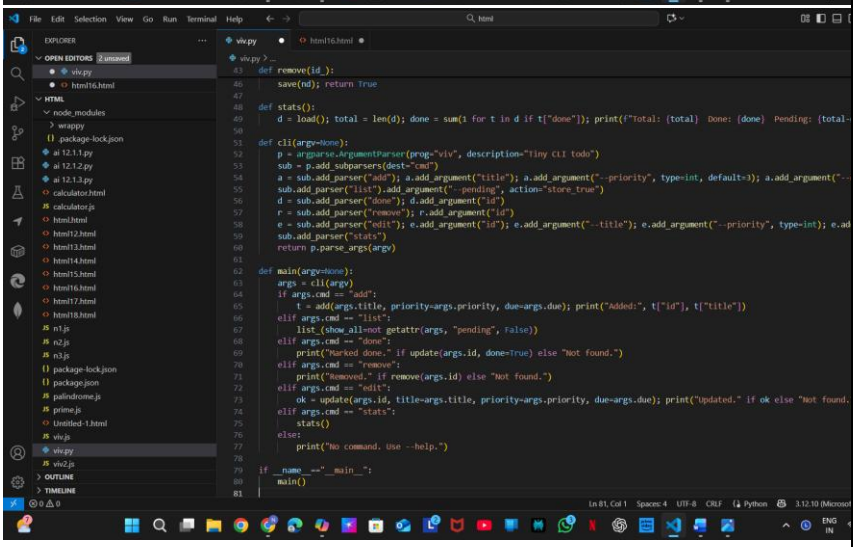
- Explanation: generate_ai_commit_message(changes) is a stub that returns a fixed, structured commit message (would normally derive text from the changes).
- get_human_commit_message() returns a sample human-written commit message.
- compare_commit_messages() prepares a mock list of changed files, calls both generators, and prints the AI/human messages plus three brief claimed benefits.
- The if **name** == "**main**" block runs compare_commit_messages() when the script is executed directly.
- Note: AI generation is mocked — to be useful replace the stub with real diff parsing or an ML/model call for context-aware messages.
-
-

Task 4 – Pair Programming Simulation

- Student A writes the initial version of a program.
- Student B, with AI assistance, suggests **improvements or adds features**.
- Both students collaborate using **Git branching and merging** to

integrate their work.

- 

- 

- Explanation: Tiny CLI todo app (c:\Users\vivek\html\viv.py) that stores tasks in a JSON file next to the script (same name with .todos.json).
- load()/save() handle JSON persistence; add() creates a task with an 8-char UUID, priority, created_at (UTC ISO), done flag, and optional due.
- list_() prints tasks sorted by done, priority, and creation time; supports hiding completed tasks with --pending.
- update(id, **kw) edits fields if provided, remove(id) deletes a task, and stats() prints totals/done/pending.
- CLI wired with argparse: commands are add, list, done, remove, edit, and stats; main() dispatches and prints simple status messages.
-

	✅ 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.	