|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name: Tharani. K**  **HT NO: 2303A52067**  **Batch: 32** | | | | |
|  | **Q.No.** | **Question** |  |  |
|  | 1 | Lab 2: Exploring Additional AI Coding Tools beyond Copilot – Gemini (Colab) and Cursor AI  **Lab Objectives:**   * To explore and evaluate the functionality of Google Gemini for AI-assisted coding within Google Colab. * To understand and use Cursor AI for code generation, explanation, and refactoring. * To compare outputs and usability between Gemini, GitHub Copilot, and Cursor AI. * To perform code optimization and documentation using AI tools.   **Lab Outcomes (LOs):**  After completing this lab, students will be able to:   * Generate Python code using Google Gemini in Google Colab. * Analyze the effectiveness of code explanations and suggestions by Gemini. * Set up and use Cursor AI for AI-powered coding assistance. * Evaluate and refactor code using Cursor AI features. * Compare AI tool behaviour and code quality across different platforms.   **Task 1: Word Frequency from Text File**   * **Scenario:** You are analyzing log files for keyword frequency. * **Task:** Use Gemini to generate Python code that reads a text file and counts word frequency, then explains the code. * **Prompt:** Create a Python program that creates a text file, writes sample text into it, then reads the file and displays its contents. Add clear comments in the code.   **Code:**    **Output:**    **Explanation:**  This program reads a text file and goes through its contents word by word. As it reads, it keeps track of each word and remembers how many times it appears. If the same word shows up again, the program simply increases its count. Once the entire file is processed, it prints out each word along with how often it was used. This makes it easy to understand which words appear most frequently in the text.  **Task 2: File Operations Using Cursor AI**   * **Scenario:** You are automating basic file operations. * **Task:** Use Cursor AI to generate a program that:   + Creates a text file   + Writes sample text   + Reads and displays the content     **Code:**  **Output:**    **Explanation:**  This program shows how basic file handling works in Python using a cursor. It starts by creating a text file and writing some sample text into it. After that, the file is opened again in read mode to display its contents on the screen. Through this process, the program demonstrates how easily Python can create, write to, and read from files. These file operations are commonly used in tasks like automation and simple data storage.  **Top of Form**  **Bottom of Form**  **Task 3: CSV Data Analysis**   * **Scenario:** You are processing structured data from a CSV file. * **Task:** Use Gemini in Colab to read a CSV file and calculate mean, min, and max. * **Prompt:**     **Code:**     * **Expected Output:**   **Explanation:**  This program reads data from a CSV file using Python. It focuses on a column of numerical values and calculates useful statistics such as the average (mean), the smallest value, and the largest value. By doing this, the program shows how CSV files can be analyzed efficiently. Such analysis is commonly used in data processing and analytics applications.  **Task 4: Sorting Lists – Manual vs Built-in**   * **Scenario:** You are reviewing algorithm choices for efficiency. * **Task:** Use **Gemini** to generate:   + Bubble sort   + Python’s built-in sort()   + Compare both implementations.   **Prompt:**    **Code:**  **Manual Bubble Sort Code:**    **Output:**    **Sorting using sort() function:**    **Output:**    **Comparison:**  The sort() function is highly efficient and best suited for real-world programs because it is fast, reliable, and easy to use. Manual Bubble Sort, on the other hand, is mainly used for educational purposes to understand how sorting works internally, but it is not practical for large datasets due to its poor performance. |  |  |