|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE** | | | | | **DEPARTMENT OF COMPUTER SCIENCE ENGINEERING** | | | | |
| **ProgramName:**B. Tech | | | | **Assignment Type: Lab** | | | **AcademicYear:**2025-2026 | | |
| **CourseCoordinatorName** | | | | 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) | | | | | | |
| **CourseCode** | | | 24CS002PC215 | **CourseTitle** | | AI Assisted Coding | | | |
| **Year/Sem** | | | II/I | **Regulation** | | R24 | | | |
| **Date and Day**  **of Assignment** | | | Week1 - Thursday | **Time(s)** | |  | | | |
| **Duration** | | | 2 Hours | **Applicableto**  **Batches** | | 24CSBTB01 To 24CSBTB39 | | | |
| **AssignmentNumber:2.4**(Present assignment number)/**24**(Total number of assignments) | | | | | | | | | |
|  | | | | | | | | | |
|  | **Q.No.** | **Question** | | | | | | ***ExpectedTime***  ***to complete*** |  |
|  | 1 | Lab 2: Exploring Additional AI Coding Tools – 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 behavior and code quality across different platforms.   **Task Description #1**  **•** Open Google Colab and use Google Gemini to generate Python code that performs sorting of a list using both the bubble sort algorithm and Python’s built-in sort() function. Compare the two implementations.  **Expected Output #1**  **•** Two sorting implementations: Bubble sort (manual logic) and Built-in sort()  **Prompt:**  **•** Give python code that performs sorting of a list using both the bubble sort algorithm and Python’s built-in sort() function using dynamic input**.**    **Observation:**   * **Takes user input as a space-separated string of numbers.** * **Converts input into a list of integers using map(int, str.split()).** * **Uses two nested loops to repeatedly swap adjacent elements if they are in the wrong order.** * **Continues until the entire list is sorted.** * **Sorts a copy of the original list using my\_list.copy() to avoid modifying the original input.** * **Prints both the original and the sorted list.**   **Task Description #2**  **•** In Colab, use Google Gemini to generate a Python function that takes a string and returns:  The number of vowels, The number of consonants, The number of digits in the string  **Expected Output #2-**  **•** Complete function that Iterates through characters of a string and Counts vowels, consonants, and digits  **Prompt:**  • Give python code that takes a string and returns the number of vowels, The number of consonants, The number of digits in the string using dynamic input.    **Observation:**   * Check if a number is prime (although this code hasn't been executed in the current state, the cell with this code is present). * For counting characters in a string successfully took dynamic input and produced the correct counts for the example provided in the execution output.   **Task Description #3**  **•** Install and set up Cursor AI. Use it to generate a Python program that performs file handling:  Create a text file  Write sample text  Read and display the content  **Expected Output #3**  **•** Functional code that creates a .txt file, writes content to it, and reads it back.  • Screenshot of Cursor AI interface showing: Prompt used,  Generated code, Output of file operations.  **Promt:**  • Give a python code that performs file handling create a text file using dynamic    **Observation:**  **•**This Python code snippet prompts the user to enter a file name and content. It then attempts to create a new text file with the specified name and writes the provided content into it. Upon successful creation and writing, it prints a confirmation message indicating the file name and successful operation. If there is an error during the file creation or writing process (such as permission issues), it catches the IOError exception and prints an error message.  **Task Description #4**  • Ask Google Gemini to generate a Python program that implements a simple calculator using functions (add, subtract, multiply, divide). Then, ask Gemini to explain how the code works.  **Expected Output #4**  **•** Complete calculator code with user input and operation selection.  • Line-by-line explanation or markdown-style explanation provided by Gemini.  • Screenshot of both the code and explanation in Colab.  **Promt:**  • Give python code that implements a simple calculator using functions (add, subtract, multiply, divide).    **Observation:**  • This Python code implements a simple calculator that performs basic arithmetic operations: addition, subtraction, multiplication, and division. It defines separate functions for each operation. The program then enters a loop, prompting the user to select an operation and input two numbers. It includes error handling to ensure the user enters valid numbers and prevents division by zero. Finally, it calls the appropriate function based on the user's choice and prints the result.  **Task Description #5** • Use Cursor AI to create a Python program that checks if a given year is a leap year or not. Try different prompt styles and see how Cursor modifies its code suggestions.  **Expected Output #5** • A functional program to check leap year with sample input/output • At least two versions of the code (from different prompts) • A short comparison of which version is better and why  **Promt:**  🡪Give python code that checks if a given year is a leap year or not.    **Observation:**  **🡪** The code in this cell provides a clear and functional implementation for determining if a given year is a leap year. It includes a well-defined function is\_leap\_year that encapsulates the standard leap year logic, checking for divisibility by 4, 100, and 400. Furthermore, the code effectively utilizes dynamic input, prompting the user to enter a year to check. It also incorporates basic error handling using a try-except block to catch ValueError in case the user enters non-integer input, providing a user-friendly message. The output from the last execution, which correctly identified 2024 as a leap year based on the user's input, confirms the code's functionality for valid integer inputs.  **Note: Report should be submitted a word document for all tasks in a single document with prompts, comments & code explanation, and output and if required, screenshots**  **Evaluation Criteria:**   | **Criteria** | **Max Marks** | | --- | --- | | Two sorting implementations: Bubble sort (manual logic) and Built-in sort() (Task#1) | 0.5 | | Counts vowels, consonants, and digits(Task#2) | 0.5 | | Functional code that creates a .txt file, writes content to it, and reads it back- Use cursor (Task#3) | 0.5 | | Complete calculator code with user input and operation selection. (Task#4) | 0.5 | | A functional program to check leap year with sample input/output-use Cursor (Task#5) | 0.5 | | **Total** | **2.5 Marks** | | | | | | | Week1 - Thursday |  |