|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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:  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.Describe the comparison    **Observation : While bubble sort is easy to understand and implement, it is not practical for sorting anything but very small lists due to its poor time complexity. Python's built-in sorted() function (using Timsort) is highly optimized and is the preferred method for sorting in Python for most real-world applications due to its superior performance on larger datasets.**  **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: 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    **Observation : The generated function count\_chars takes a string as input and iterates through each character. It converts each character to lowercase to handle case-insensitivity. It checks if the character is a digit using isdigit(). If it's not a digit, it checks if it's an alphabet character using isalpha(). If it's an alphabet character, it further checks if it's a vowel or a consonant. The function maintains separate counters for vowels, consonants, and digits, incrementing them as it processes each character. Finally, it returns the total counts of vowels, consonants, and digits.**  **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  **Prompt:** to generate a Python program that performs file handling:  Create a text file  Write sample text  Read and display the content    **Observation:** **A file named sample.txt was successfully created.**  **The provided sample text was written into the sample.txt file.**  **The content of sample.txt was successfully read back and displayed in the output.**  **This confirms that the file handling operations (create, write, and read) were executed as expected**  **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.  **Prompt :** generate a Python program that implements a simple calculator using functions (add, subtract, multiply, divide). explain how the code works.      **Observation :**  **Functions for Operations:add(x, y), subtract(x, y), multiply(x, y), and divide(x, y) are defined to perform the basic arithmetic operations.**  **User Interface:The code prints a menu of available operations (Add, Subtract, Multiply, Divide) with corresponding numbers (1-4).**  **Input Loop:A while True loop is used to allow the user to perform multiple calculations until they choose to exit.**  **Continue or Exit:After each calculation, the program asks the user if they want to perform another calculation.If the user enters "no" (case-insensitive), the break statement is executed, exiting the while loop and ending the program.**  **Invalid Input:If the user enters an invalid choice (not 1, 2, 3, or 4), an "Invalid Input" message is printed.**  **This program provides a simple command-line interface for performing basic arithmetic operations.**  **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  Prompt : Create a Python program that checks if a given year is a leap year or not.    **Observation: Based on the input year 2023 and the output 2023 is not a leap year., the program correctly determined that the year 2023 is not a leap year according to the rules for calculating leap years.**  **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 |  |