|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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** | | | | Dr. Rishabh Mittal | | | | |
| **Instructor(s) Name** | | | | |  | | --- | | Mr. S Naresh Kumar | | Ms. B. Swathi | | Dr. Sasanko Shekhar Gantayat | | Mr. Md Sallauddin | | Dr. Mathivanan | | Mr. Y Srikanth | | Ms. N Shilpa | | Dr. Rishabh Mittal (Coordinator) | | Dr. R. Prashant Kumar | | Mr. Ankushavali MD | | Mr. B Viswanath | | Ms. Sujitha Reddy | | Ms. A. Anitha | | Ms. M.Madhuri | | Ms. Katherashala Swetha | | Ms. Velpula sumalatha | | Mr. Bingi Raju | | | | | |
| **CourseCode** | | | 23CS002PC304 | **Course Title** | | AI Assisted Coding | | |
| **Year/Sem** | | | III/II | **Regulation** | | R23 | | |
| **Date and Day**  **of Assignment** | | | **Week1 – Tuesday** | **Time(s)** | | 23CSBTB01 To 23CSBTB52 | | |
| **Duration** | | | 2 Hours | **Applicable to**  **Batches** | | All batches | | |
| **Assignment Number:1.3**(Present assignment number)/**24**(Total number of assignments) | | | | | | | | |
|  | | | | | | | | |
|  | **Q.No.** | **Question** | | | | | | ***Expected Time***  ***to complete*** |  |
|  | 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 behavior and code quality across different platforms.   **Task 1: Cleaning Sensor Data**  **Scenario**:  You are cleaning IoT sensor data where negative values are invalid.  **Task:**  Use Gemini in Colab to generate a function that filters out all negative numbers from a list.  **Expected Output:**   * + Before/after list   + Screenshot of Colab execution   **PROMPT:** Write a Python program that asks the user to enter a list of sensor readings, filters out all negative values, and prints the list before and after cleaning. Take input as comma-separated numbers.  **CODE:**    **OUTPUT:**    **OBSERVATION:**  The program successfully removed all negative sensor readings, ensuring only valid data is used for analysis. User input was correctly taken, converted into a list, and cleaned using the filter function. This improves the accuracy and reliability of the IoT dataset.  **Task 2: String Character Analysis**  **Scenario:**  You are building a text-analysis feature.  **Task:**  Use Gemini to generate a Python function that counts vowels, consonants, and digits in a string.  **Expected Output:**   * + Working function   + Sample inputs and outputs   **PROMPT:** Write a Python function that takes a user-entered string and counts the number of vowels, consonants, and digits. Display the original string and show each count separately with sample input and output  **CODE:**  **OUTPUT:**    **OBSERVATION:**  The function accurately identifies vowels, consonants, and digits from the user-entered string, demonstrating correct character classification. The sample tests confirm that the program handles mixed inputs effectively, making it suitable for basic text-analysis applications.  **Task 3: Palindrome Check – Tool Comparison**  **Scenario**:  You must decide which AI tool is clearer for string logic.  **Task:**  Generate a palindrome-checking function using Gemini and Copilot, then compare the results.  **Expected Output:**   * + Side-by-side code comparison   + Observations on clarity and structure   **PROMPT:** Generate user-input based palindrome checking code in Python using Gemini style and Copilot style, then compare both with observations  **CODE AND OUTPUT:**  GEMINI:    COPILOT:    **OBSERVATION:**  **Gemini:** More descriptive, uses intermediate variables, easier for beginners to understand.  **Copilot:** Short, compact, efficient, but less explanatory.  Both produce correct logic, but Gemini is clearer while Copilot focuses on speed and brevity  **Task 4: Code Explanation Using AI**  **Scenario:**  You are reviewing unfamiliar code written by another developer.  **Task:**  Ask Gemini to explain a Python function (prime check OR palindrome check) line by line.  **Expected Output:**   * + Code snippet   + AI explanation   + Student comments on understanding   **PROMPT:** Take a Python function (prime check or palindrome check), generate a version that takes user input, explain it line by line as Gemini would, and provide observations on understanding  **CODE:**    **Line 1:** def is\_prime(num): This defines a function named *is\_prime* that accepts a number as input.  **Line 2:** if num <= 1: Numbers less than or equal to 1 are not prime, so we first check that condition.  **Line 3:** return False If the number is 1 or below, the function immediately returns *False*.  **Line 4**: for i in range(2, int(num\*\*0.5) + 1): The loop tests possible divisors from 2 to the square root of the number, which improves performance.  **Line 5:** if num % i == 0:  If the number is divisible by any *i*, it is not prime.  Line 6: return False  Since a divisor is found, the function returns *False*.  Line 7: return True If no divisors are found in the loop, the number is prime**.**  **OBSERVATION:**  The code efficiently checks if a number is prime by testing divisors only up to the square root, which improves performance. The logic is straightforward—return *False* when a divisor is found, otherwise return *True*. User input is handled cleanly, making the program interactive and easy to understand.  **Note: Report should be submitted as a word document for all tasks in a single document with prompts, comments & code explanation, and output and if required, screenshots.** | | | | | | Week1 - Monday |  |