|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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** | | | Week5 - Monday | **Time(s)** | |  | | | |
| **Duration** | | | 2 Hours | **Applicable to**  **Batches** | |  | | | |
| **AssignmentNumber: 9.1**(Present assignment number)/**24**(Total number of assignments) | | | | | | | | | |
|  | **Q.No.** | **Question** | | | | | | ***Expected Time***  ***to complete*** |  |
|  | 1 | **Lab 9 – Documentation Generation: Automatic Documentation and Code Comments**  **Lab Objectives**   * To use AI-assisted coding tools for generating Python documentation and code comments. * To apply zero-shot, few-shot, and context-based prompt engineering for documentation creation. * To practice generating and refining docstrings, inline comments, and module-level documentation. * To compare outputs from different prompting styles for quality analysis.   **Task Description #1** (Documentation – Google-Style Docstrings for Python Functions)   * Task: Use AI to add Google-style docstrings to all functions in a given Python script. * Instructions:   + Prompt AI to generate docstrings without providing any input-output examples.   + Ensure each docstring includes:     - Function description     - Parameters with type hints     - Return values with type hints     - Example usage   + Review the generated docstrings for accuracy and formatting.      * Expected Output #1:   + A Python script with all functions documented using correctly formatted Google-style docstrings.   **Task Description #2** (Documentation – Inline Comments for Complex Logic)   * Task: Use AI to add meaningful inline comments to a Python program explaining only complex logic parts. * Instructions:   + Provide a Python script without comments to the AI.   + Instruct AI to skip obvious syntax explanations and focus only on tricky or non-intuitive code sections.   + Verify that comments improve code readability and maintainability.      * Expected Output #2:   + Python code with concise, context-aware inline comments for complex logic blocks.   **Task Description #3** (Documentation – Module-Level Documentation)   * Task: Use AI to create a module-level docstring summarizing the purpose, dependencies, and main functions/classes of a Python file. * Instructions:   + Supply the entire Python file to AI.   + Instruct AI to write a single multi-line docstring at the top of the file.   + Ensure the docstring clearly describes functionality and usage without rewriting the entire code.        * Expected Output #3:   + A complete, clear, and concise module-level docstring at the beginning of the file.   **Task Description #4** (Documentation – Convert Comments to Structured Docstrings)   * Task: Use AI to transform existing inline comments into structured function docstrings following Google style. * Instructions:   + Provide AI with Python code containing inline comments.   + Ask AI to move relevant details from comments into function docstrings.   + Verify that the new docstrings keep the meaning intact while improving structure.      * Expected Output #4:   + Python code with comments replaced by clear, standardized docstrings.   **Task Description #5** (Documentation – Review and Correct Docstrings)   * Task: Use AI to identify and correct inaccuracies in existing docstrings. * Instructions:   + Provide Python code with outdated or incorrect docstrings.   + Instruct AI to rewrite each docstring to match the current code behavior.   + Ensure corrections follow Google-style formatting      * Expected Output #5:   + Python file with updated, accurate, and standardized docstrings.   **Task Description #6** (Documentation – Prompt Comparison Experiment)   * Task: Compare documentation output from a vague prompt and a detailed prompt for the same Python function. * Instructions:   + Create two prompts: one simple (“Add comments to this function”) and one detailed (“Add Google-style docstrings with parameters, return types, and examples”).   + Use AI to process the same Python function with both prompts.   + Analyze and record differences in quality, accuracy, and completeness.        * Expected Output #6:   + A comparison table showing the results from both prompts with observations. | | | | | | Week5 - Monday |  |