|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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** | | | Week4 - Wednesday | **Time(s)** | |  | | | |
| **Duration** | | | 2 Hours | **Applicableto**  **Batches** | |  | | | |
| **AssignmentNumber:9.3**(Present assignment number)/**24**(Total number of assignments) | | | | | | | | | |
|  | | | | | | | | | |
|  | | | | | | | | | |
|  | **Q.No.** | **Question** | | | | | | ***ExpectedTime***  ***to complete*** |  |
|  | 1 | Lab 8: Documentation Generation: Automatic documentation and code comments  **Lab Objectives:**   * To understand the importance of documentation and code comments in software development. * To explore how AI-assisted coding tools can generate meaningful documentation and inline comments. * To practice generating function-level and module-level docstrings automatically. * To evaluate the quality, accuracy, and limitations of AI-generated documentation. * To develop a small automated tool for documentation generation in Python..     **Lab Outcomes (LOs):**  After completing this lab, students will be able to:   * Apply AI-assisted coding tools to generate docstrings and inline comments for Python code. * Critically analyze AI-generated documentation for correctness, completeness, and readability. * Create structured documentation (function-level, module-level) following standard formats. * Design and implement a mini documentation generator tool to automate code commenting and docstring creation.   **Task Description#1 Basic Docstring Generation**   * Write python function to return sum of even and odd numbers in the given list. * Incorporate manual **docstring** in code with Google Style * Use an AI-assisted tool (e.g., Copilot, Cursor AI) to generate a docstring describing the function. * Compare the AI-generated docstring with your manually written one.   **PROMPT: Write python function to return sum of even and odd numbers in the given list.**  **CODE:**  **MY MANUAL DOCSTRING:**  **def sum\_even\_odd(numbers):**  **"""**  **Return the sum of even and odd numbers in a given list.**  **This function takes a list of integers and calculates two sums:**  **one for all even numbers and one for all odd numbers in the list.**  **Args:**  **numbers (list of int): The list of integers to process.**  **Returns:**  **tuple: A tuple containing:**  **- The sum of even numbers (int)**  **- The sum of odd numbers (int)**  CODE WITH DOCSTRING IN VS:    **Expected Outcome#1:** Students understand how AI can produce function-level documentation.  **OUTPUT:**    **Comparsion** **AI-generated docstring with MY manual:**   * **manual docstring: Clearly states the purpose ("Return the sum of even and odd numbers in a given list") and explains what the function does in a separate sentence.Uses a structured format, including a description, argument details, and an example of usage.** * **ai docstring: Only briefly states "Calculates the sum of even and odd numbers in a list" without further explanation.Only lists arguments and return values, with no example or extended description.**   **In summary, the first docstring is more descriptive, structured, and user-friendly than the second.**  **Task Description#2 Automatic Inline Comments**   * Write python program for **sru\_student** class with attributes like name, roll no., hostel\_status and **fee\_update** method and **display\_details** method. * Write comments manually for each line/code block. * Ask an AI tool to add inline comments explaining each line/step. * Compare the AI-generated comments with your manually written one.   **Prompt:**  **Write python program for sru\_student class with attributes like name, roll no., hostel\_status and fee\_update method and display\_details method.**  **Code in vs:**    **MANUAL COMMENTS:**  **# Student class for SRU**  **class sru\_student:**  **def \_\_init\_\_(self, name, roll\_no, hostel\_status):**  **self.name = name # Student name**  **self.roll\_no = roll\_no # Roll number**  **self.hostel\_status = hostel\_status # Hostel Yes/No**  **self.fee\_paid = False # Fee not paid by default**  **def fee\_update(self, status):**  **self.fee\_paid = status # Update fee status**  **def display\_details(self):**  **print(f"Name: {self.name}") # Show name**  **print(f"Roll No.: {self.roll\_no}") # Show roll no**  **print(f"Hostel Status: {self.hostel\_status}") # Show hostel info**  **print(f"Fee Paid: {'Yes' if self.fee\_paid else 'No'}") # Show fee status**  **# Create student object**  **student1 = sru\_student("Rahul", "SRU123", "Yes")**  **# Mark fee as paid**  **student1.fee\_update(True)**  **# Show all details**  **student1.display\_details()**  **AI GENERATED COMMENTS:**  class sru\_student: # Define a class for SRU students  def \_\_init\_\_(self, name, roll\_no, hostel\_status): # Constructor to initialize student details  self.name = name # Store student's name  self.roll\_no = roll\_no # Store student's roll number  self.hostel\_status = hostel\_status # Store hostel status (Yes/No)  self.fee\_paid = False # Initialize fee status as not paid  def fee\_update(self, status): # Method to update fee payment status  self.fee\_paid = status # Set fee\_paid to given status (True/False)  def display\_details(self): # Method to display student details  print(f"Name: {self.name}") # Print student's name  print(f"Roll No.: {self.roll\_no}") # Print student's roll number  print(f"Hostel Status: {self.hostel\_status}") # Print hostel status  print(f"Fee Paid: {'Yes' if self.fee\_paid else 'No'}") # Print fee payment status  # Example usage  student1 = sru\_student("Rahul", "SRU123", "Yes") # Create a student object with details  student1.fee\_update(True) # Update fee status to paid  student1.display\_details() # Display all details of  **Expected Output#2:** Students critically analyze AI-generated code comments.  **COMPARSION** :  On the other hand, AI-generated comments are more formal and detailed, using full sentences and technical language. They’re better suited for documentation or when explaining code to someone new. In short, manual comments are quick and easy to follow, while AI comments are more structured and informative.  **OUTPUT:**  **Task Description#3**   * Write a Python script with 3–4 functions (e.g., calculator: add, subtract, multiply, divide). * Incorporate manual **docstring** in code with NumPy Style * Use AI assistance to generate a module-level docstring + individual function docstrings. * Compare the AI-generated docstring with your manually written one. * **PROMPT: Write a Python script with 3–4 functions (e.g., calculator: add, subtract, multiply, divide).**   **AI CODE WITH DOCSTRING:**      **MANUAL DOCSTRING:**  **def add(a, b):**  **"""**  **Add two numbers.**  **Parameters**  **----------**  **a, b : int or float**  **Numbers to add.**  **Returns**  **-------**  **int or float**  **Sum of a and b.**  **"""**  **return a + b**  **def subtract(a, b):**  **"""**  **Subtract b from a.**  **Parameters**  **----------**  **a, b : int or float**  **Numbers to subtract.**  **Returns**  **-------**  **int or float**  **Difference of a and b.**  **"""**  **return a - b**  **def multiply(a, b):**  **"""**  **Multiply two numbers.**  **Parameters**  **----------**  **a, b : int or float**  **Numbers to multiply.**  **Returns**  **-------**  **int or float**  **Product of a and b.**  **"""**  **return a \* b**  **def divide(a, b):**  **"""**  **Divide a by b.**  **Parameters**  **----------**  **a, b : int or float**  **Numerator and denominator.**  **Returns**  **-------**  **float**  **Result of division.**  **Raises**  **------**  **ValueError**  **If b is zero.**  **"""**  **if b == 0:**  **raise ValueError("Cannot divide by zero.")**  **return a / b**  **# Example**  **print("Add:", add(10, 5))**  **print("Subtract:", subtract(10, 5))**  **print("Multiply:", multiply(10, 5))**  **print("Divide:", divide(10, 5))**  **OUTPUT:**    **COMPARSION:**   1. **MANUAL CODE -First code uses NumPy-style docstrings – clean, structured, with sections like Parameters, Returns, and Raises.** 2. **AI DOC STRING:Second code uses a mixed/Google-style docstring only in multiply, while other functions have no docstrings.** 3. **First one is consistent and more professional for documentation.** 4. **Second one is simpler, but incomplete and lacks uniformity.** 5. **Overall, the first style is better for clarity and learning.**   **Expected Output#3:** Students learn structured documentation for multi-function scripts  **Push documentation whole workspace as .md file in GitHub Repository**  **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** | | | | | | Week4 - Wednesday |  |