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:10.1(Present assignment number)/**24**(Total number of assignments)			

Q.No.	Question	*Expected Time to complete*
1	**Lab 10 – Code Review and Quality: Using AI to Improve Code Quality and Readability** **Lab Objectives** • Use AI for automated code review and quality enhancement. • Identify and fix syntax, logical, performance, and security issues in Python code. • Improve readability and maintainability through structured refactoring and comments.	Week5 - Monday

	• Apply prompt engineering for targeted improvements. • Evaluate AI-generated suggestions against PEP 8 standards and software engineering best practices	

	Task Description #1 – Syntax and Logic Errors
	Task: Use AI to identify and fix syntax and logic errors in a faulty Python script.
	Sample Input Code:
	# Calculate average score of a student
	def calc_average(marks):
	total = 0
	for m in marks:
	total += m
	average = total / len(marks)
	return avrage # Typo here
	
	marks = [85, 90, 78, 92]
	print("Average Score is ", calc_average(marks)
	Expected Output:
	• Corrected and runnable Python code with explanations of the fixes.
	CODE:	

```
"""
def calc_average(marks):
    total = 0
    for calc_average(marks):
    total = 0
    for m in marks:
        total +=m
    average=total/len(marks)
    return average
marks = [85,90,78,92]
print("Average score is",calc_average)
"""
def calc_average(marks: list[int]) -> float:
    """
    Calculate the average of a list of integer marks.

    Args:
        marks (list[int]): A list of integer marks.

    Returns:
        float: The average score.

    Example:
        >>> calc_average([85, 90, 78, 92])
        86.25
    """
    total = 0
    for m in marks:
        total += m
    average = total / len(marks)
    return average

marks = [85, 90, 78, 92]
print("Average score is", calc_average(marks))
```

OUTPUT:

```
PS C:\Users\DEEKSHA\OneDrive\Desktop\AIAC\Lab-10>
Average score is 86.25
PS C:\Users\DEEKSHA\OneDrive\Desktop\AIAC\Lab-10>
```

Task Description #2 – PEP 8 Compliance

Task: Use AI to refactor Python code to follow PEP 8 style guidelines.

Sample Input Code:

def area_of_rect(L,B):return L*B

print(area_of_rect(10,20))

Expected Output:

- Well-formatted PEP 8-compliant Python code.

CODE :

```python
#def area_of_rect(L,B):return L*B
#print(area_of_rect(10,20))
def area_of_rectangle(length: int, breadth: int) -> int:
    """
    Calculate the area of a rectangle using integer values.

    Args:
        length (int): The length of the rectangle (integer).
        breadth (int): The breadth (width) of the rectangle (integer).

    Returns:
        int: The area of the rectangle as an integer.

    Example:
        >>> area_of_rectangle(5, 3)
        15
    """
    # Multiply length by breadth to get the area (integers only)
    return length * breadth

# Hint: Use area_of_rectangle(length, breadth) with integer arguments to compute the area.
# Example usage:
print(area_of_rectangle(10, 20))  # Output: 200
```

OUTPUT:

```
200
PS C:\Users\DEEKSHA\OneDrive\Desktop\AIAC\Lab-10
```

Task Description #3 – Readability Enhancement

Task: Use AI to make code more readable without changing its logic**.**

Sample Input Code:

```
def c(x,y):
 return x*y/100
a=200
b=15
print(c(a,b))
```

Expected Output:

- Python code with descriptive variable names, inline comments, and clear formatting.

CODE:

```
"""
def c(x,y):
    return x*y/100
a=200
b=15
print(c(a,b))
"""
# Calculate the percentage increase from an old value to a new value

def calculate_percentage_increase(old_value: float, new_value: float) -> float:
    """
    Calculate the percentage increase from old_value to new_value.

    Args:
        old_value (float): The original value.
        new_value (float): The new value.

    Returns:
        float: The percentage increase from old_value to new_value.

    Example:
        >>> calculate_percentage_increase(200, 15)
        20.0
    """
    if old_value == 0:
        return float('inf')  # Avoid division by zero
    increase = new_value - old_value
    percentage_increase = (increase / old_value) * 100
    return percentage_increase

# Example usage:
old_price = 200
new_price = 15
print(f"Percentage increase from {old_price} to {new_price} is {calculate_percentage_increase(old_price, new_price
```

OUTPUT:

```
Percentage increase from 200 to 15 is -92.5%
PS C:\Users\DEEKSHA\OneDrive\Desktop\AIAC\Lab-10>
```

Task Description #4 – Refactoring for Maintainability

Task: Use AI to break repetitive or long code into reusable functions.

Sample Input Code:

students = ["Alice", "Bob", "Charlie"]

print("Welcome", students[0])

print("Welcome", students[1])

print("Welcome", students[2])

Expected Output:

- Modular code with reusable functions.

CODE:

```
"""
students = ["Alice","Bob","Charlie"]
print("Welcome",students[0])
print("Welcome",students[1])
print("Welcome",students[2])
"""
def welcome_student(name: str) -> None:

    """
    Print a welcome message for a single student.

    Args:
        name (str): The name of the student.
    """
    print("Welcome", name)

def welcome_students(student_list: list[str]) -> None:
    """
    Print welcome messages for a list of students.

    Args:
        student_list (list[str]): List of student names.
    """
    for student in student_list:
        welcome_student(student)

# Example usage:
welcome_students("Alice")
```

OUTPUT:

```
pData/Local/Microsoft/WindowsApps/python3.11.exe c:/Us
Welcome A
Welcome l
Welcome i
Welcome c
Welcome e
```

Task Description #5 – Performance Optimization

Task: Use AI to make the code run faster.

Sample Input Code:

```
# Find squares of numbers
nums = [i for i in range(1,1000000)]
squares = []
for n in nums:
    squares.append(n**2)
print(len(squares))
```

Expected Output:

- Optimized code using list comprehensions or vectorized operations.

CODE:

```
"""
nums = [i for i in range(1,1000000)]
squares=[]
for n in nums:
    squares.append(n**2)
print(len(squares))
"""
import time
"""
This script measures the time taken to compute the squares of numbers from 1 to 999,999

Steps:
1. It creates a list of numbers from 1 to 999,999.
2. It computes the square of each number and stores them in a new list.
3. It prints the total number of squares computed.
4. It prints the time taken to perform the computation.

Example Output:
999999
Time taken to run the code: 0.1234 seconds
"""
start_time = time.time()
nums = [i for i in range(1,1000000)]
squares = []
for n in nums:
    squares.append(n**2)
print(len(squares))
end_time = time.time()
print(f"Time taken to run the code: {end_time - start_time:.4f} seconds")
```

OUTPUT:

```
PS C:\Users\DEEKSHA\OneDrive\Desktop\AIAC\Lab-10> &
999999
Time taken to run the code: 0.1009 seconds
```

Task Description #6 – Complexity Reduction

Task: Use AI to simplify overly complex logic.

Sample Input Code:

```
def grade(score):
    if score >= 90:
        return "A"
    else:
        if score >= 80:
            return "B"
        else:
            if score >= 70:
                return "C"
```

```
        else:
            if score >= 60:
                return "D"
            else:
                return "F"
```

Expected Output:

- Cleaner logic using elif or dictionary mapping.

CODE:

```
"""
def grade(score):
    if score>=90:
        return "A"
    else:
        if score>=80:
            return "B"
        else:
            if score >= 70:
                return "C"
            else:
                if score >= 60:
                    return "D"
                else:
                    return "F"
"""                         # Cleaner logic using elif and docstring
def grade(score: int) -> str:
    """
    Return the letter grade for a given score.

    Args:
        score (int): The numeric score (0-100).

    Returns:
        str: The letter grade ("A", "B", "C", "D", or "F").
    """
    if score >= 90:
        return "A"
    elif score >= 80:
        return "B"
    elif score >= 70:
        return "C"
    elif score >= 60:
        return "D"
    else:
```

```
        return "F"

# Get user input and print the grade
try:
    user_score = int(input("Enter the score (0-100): "))
    if 0 <= user_score <= 100:
        print(f"Grade: {grade(user_score)}")
    else:
        print("Please enter a score between 0 and 100.")
except ValueError:
    print("Invalid input. Please enter an integer value.")
```

OUTPUT:

```
PS C:\Users\DEEKSHA\OneDrive\Desktop\AIAC\Lab-1
Enter the score (0-100): 95
Grade: A
PS C:\Users\DEEKSHA\OneDrive\Desktop\AIAC\Lab-1
```