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	**Week5 – Monday**	**Time(s)**	23CSBTB01 To 23CSBTB52
Duration	2 Hours	**Batch**	**Batch-46**
Name	**N.Jashwanth Reddy**	**HallTicket No.**	**2303A53017**
Assignment Number: 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**	Week5 - Monday

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

Lab Outcomes

1. Students will be able to use AI tools to review code.
2. Students will be able to improve code quality and readability.
3. Students will be able to identify and fix common coding issues.

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.

Prompt:

Identify and fix syntax and logical errors in the following Python code and provide corrected runnable code.

Code:

```
def 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(marks))
```

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.

Prompt:

Refactor the following Python code to follow PEP 8 style guidelines and improve readability.

Code:

```
# Function to calculate area of a rectangle
def area_of_rectangle(length, breadth):
   return length * breadth
print("Area of Rectangle:", area_of_rectangle(10, 20))
```

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.

Prompt:

Improve the readability of the following Python code by using meaningful names and proper formatting without changing its logic.

Code:

```
def calculate_percentage(amount, percentage):
   return amount * percentage / 100
principal_amount = 200
interest_rate = 15

result = calculate_percentage(principal_amount, interest_rate)

print("Calculated Percentage Value:", result)
```

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.	

	Prompt:
	Refactor the following repetitive Python code into reusable functions to improve maintainability.
	Code:
	def welcome_students(student_list):
	for student in student_list:
	print("Welcome", student)
	
	students = ["Alice", "Bob", "Charlie"]
	welcome_students(students)
	
	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.	

Prompt:

Optimize the following Python code to improve performance and memory efficiency.

Code:

```
nums = range(1, 1_000_000)
squares = [n ** 2 for n in nums]

print(len(squares))
```

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.

Prompt:

Simplify the following nested conditional logic to make the code cleaner

	and more readable.
	
	Code:
	def grade(score):
	  if score >= 90:
	  return "A"
	  elif score >= 80:
	  return "B"
	  elif score >= 70:
	  return "C"
	  elif score >= 60:
	  return "D"
	  else:
	  return "F"	