Name: M. Bharath

Roll No:2303A52060

Lab 10 – Code Review and Quality: Using AI to Improve Code

Quality and Readability

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:

```
# Calculate average score of a student
```

Code:

```
def calc_average(marks):
    total = 0 # indentation error, should be indented
    for m in marks:
        total += m # indentation error, should be indented
    average = total / len(marks)
    return average # return outsside the funtion

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

output:

```
s\dell\AppData\Local\Programs\Python\Python314\python.exe c:\Users\dell\.vscode\extensions\ms-python.debu
.0-win32-x64\bundled\libs\debugpy\launcher 62759 -- "C:\Users\dell\OneDrive\Ai Assitant\.vscode\AI_ASS_La
py" "
Average Score is  86.25
```

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:

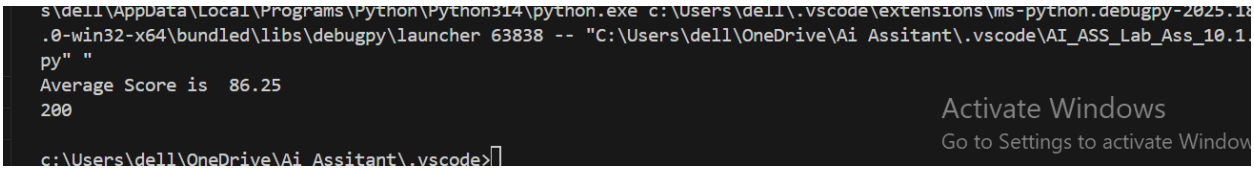
```
#Task2 – PEP 8 Compliance
```

CODE:

```
def area_of_rect(length, breadth):
    """Return the area of a rectangle."""
    return length * breadth
```

```
print(area_of_rect(10, 20))
```

Output:



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:

```
# Task 3 Readability Enhancement
```

```
#Calculate the given percentage of a value.
```

```
# Calculate result
```

Code:

```
def calculate_percentage(value, percentage):
#Calculate the given percentage of a value.
    return (value * percentage) / 100
# Given values
principal_amount = 200
interest_rate = 15
# Calculate result
result = calculate_percentage(principal_amount, interest_rate)
print(result)
```

Output:

```
c:\Users\dell\OneDrive\Ai Assitant\.vscode>
c:\Users\dell\OneDrive\Ai Assitant\.vscode> c: && cd "c:\Users\dell\OneDrive\Ai Assitant\.vscode" && cmd /C "c:\
s\dell\AppData\Local\Programs\Python\Python314\python.exe c:\Users\dell\.vscode\extensions\ms-python.debugpy-202
.0-win32-x64\bundled\libs\debugpy\launcher 53469 -- "C:\Users\dell\OneDrive\Ai Assitant\.vscode\AI_ASS_Lab_Ass_1
py" "
30.0
```

Activate Windows

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:

```
#Task 4 Refactoring for Maintainability
```

```
#Print a welcome message for each student.
```

Code:

```
def welcome_students(student_list):
#Print a welcome message for each student.
    for student in student_list:
        print("Welcome", student)
students = ["Alice", "Bob", "Charlie"]
welcome_students(students)
```

output:

```
c:\Users\dell\OneDrive\Ai Assitant\.vscode>
c:\Users\dell\OneDrive\Ai Assitant\.vscode> c: && cd "c:\Use
s\dell\AppData\Local\Programs\Python\Python314\python.exe c:
.0-win32-x64\bundled\libs\debugpy\launcher 61010 -- "C:\User
py" "
Welcome Alice
Welcome Bob
Welcome Charlie
```

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:

```
#Task6 – Complexity Reduction

#Task: Use AI to simplify overly complex logic.

# simplified version using elif or dictionary mapping
```

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"
# or using dictionary mapping
def grade(score):
    grade_mapping = {
        (90, 100): "A",
        (80, 89): "B",
        (70, 79): "C",
```

```
        (60, 69): "D",
        (0, 59): "F"
    }
    for range_tuple, grade in grade_mapping.items():
        if range_tuple[0] <= score <= range_tuple[1]:
            return grade
    return "Invalid Score"
#print the score
print(grade(95))  # Output: A
print(grade(85))  # Output: B
print(grade(75))  # Output: C
print(grade(65))  # Output: D
print(grade(55))  # Output: F
```

Output:

```
/C "c:\Users\dell\AppData\Local\Programs\Python\
python.debugpy-2025.18.0-win32-x64\bundled\libs\
tant\.vscode\AI_ASS_Lab_Ass_10.1.py" "
A
B
C
D
F
```