

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		1. Dr. Mohammed Ali Shaik 2. Dr. T Sampath Kumar 3. Mr. S Naresh Kumar 4. Dr. V. Rajesh 5. Dr. Brij Kishore 6. Dr Pramoda Patro 7. Dr. Venkataramana 8. Dr. Ravi Chander 9. Dr. Jagjeeth Singh	
CourseCode	24CS002PC215	CourseTitle	AI Assisted Coding
Year/Sem	II/I	Regulation	R24
Date and Day of Assignment	Week2-Tuesday	Time(s)	
Duration	2 Hours	Applicableto Batches	24CSBTB01 To 24CSBTB39
AssignmentNumber:3.2(Present assignment number)/24(Total number of assignments)			

Question	
----------	--

Lab 3: Prompt Engineering – Improving Prompts and Context Management

Lab Objectives:

- To understand how prompt structure and wording influence AI-generated code.
- To explore how context (like comments and function names) helps AI generate relevant output.
- To evaluate the quality and accuracy of code based on prompt clarity.
- To develop effective prompting strategies for AI-assisted programming.

Lab Outcomes (LOs):

After completing this lab, students will be able to:

- Generate Python code using Google Gemini in Google Colab.
- Analyze the effectiveness of code explanations and suggestions by Gemini.
- Set up and use Cursor AI for AI-powered coding assistance.
- Evaluate and refactor code using Cursor AI features.
- Compare AI tool behavior and code quality across different platforms.

Task Description#1

- Ask AI to write a function to calculate compound interest, starting with only the function name. Then add a docstring, then input-output example

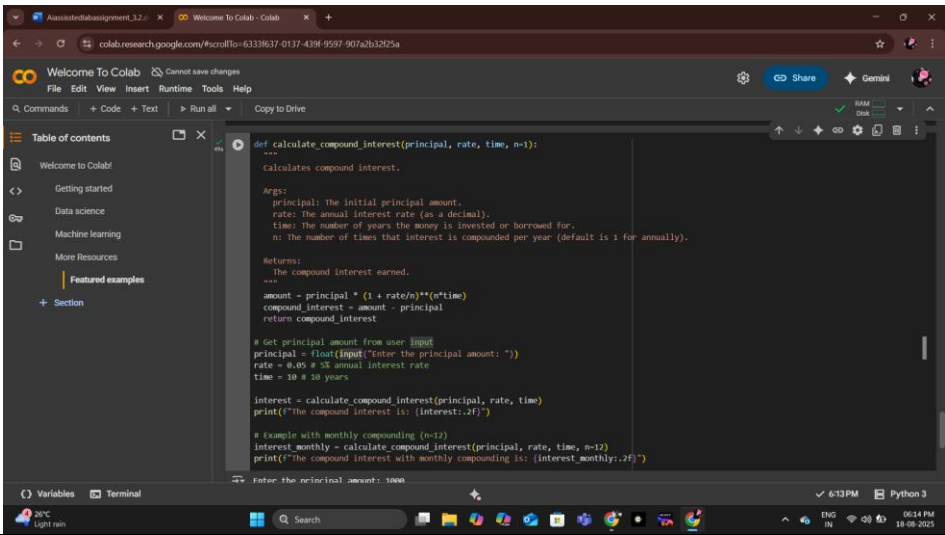
Expected Output#1

- Comparison of AI-generated code styles



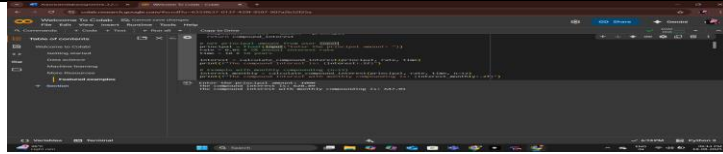
The screenshot shows the Cursor AI interface with a Python function defined. The function is named `calculate_compound_interest` and takes four parameters: `principal`, `rate`, `time`, and `n`. The function calculates the compound interest and returns it. The code is as follows:

```
def calculate_compound_interest(principal, rate, time, n=1):  
    """  
    Calculates compound interest.  
    Args:  
        principal: The initial principal amount.  
        rate: The annual interest rate (as a decimal).  
        time: The number of years the money is invested or borrowed for.  
        n: The number of times that interest is compounded per year (default is 1 for annually).  
    Returns:  
        The compound interest earned.  
    """  
    amount = principal * (1 + rate/n)**(n*time)  
    compound_interest = amount - principal  
    return compound_interest  
  
# Get principal amount from user input  
principal = float(input("Enter the principal amount: "))  
rate = 0.05 # 5% annual interest rate  
time = 10 # 10 years  
  
interest = calculate_compound_interest(principal, rate, time)  
print(f"The compound interest is: {interest:.2f}")  
  
# Example with monthly compounding (n=12)  
interest_monthly = calculate_compound_interest(principal, rate, time, n=12)  
print(f"The compound interest with monthly compounding is: {interest_monthly:.2f}")
```



The screenshot shows the Google Colab interface with a Python function defined. The function is named `calculate_compound_interest` and takes four parameters: `principal`, `rate`, `time`, and `n`. The function calculates the compound interest and returns it. The code is as follows:

```
def calculate_compound_interest(principal, rate, time, n=1):  
    """  
    Calculates compound interest.  
    Args:  
        principal: The initial principal amount.  
        rate: The annual interest rate (as a decimal).  
        time: The number of years the money is invested or borrowed for.  
        n: The number of times that interest is compounded per year (default is 1 for annually).  
    Returns:  
        The compound interest earned.  
    """  
    amount = principal * (1 + rate/n)**(n*time)  
    compound_interest = amount - principal  
    return compound_interest  
  
# Get principal amount from user input  
principal = float(input("Enter the principal amount: "))  
rate = 0.05 # 5% annual interest rate  
time = 10 # 10 years  
  
interest = calculate_compound_interest(principal, rate, time)  
print(f"The compound interest is: {interest:.2f}")  
  
# Example with monthly compounding (n=12)  
interest_monthly = calculate_compound_interest(principal, rate, time, n=12)  
print(f"The compound interest with monthly compounding is: {interest_monthly:.2f}")
```

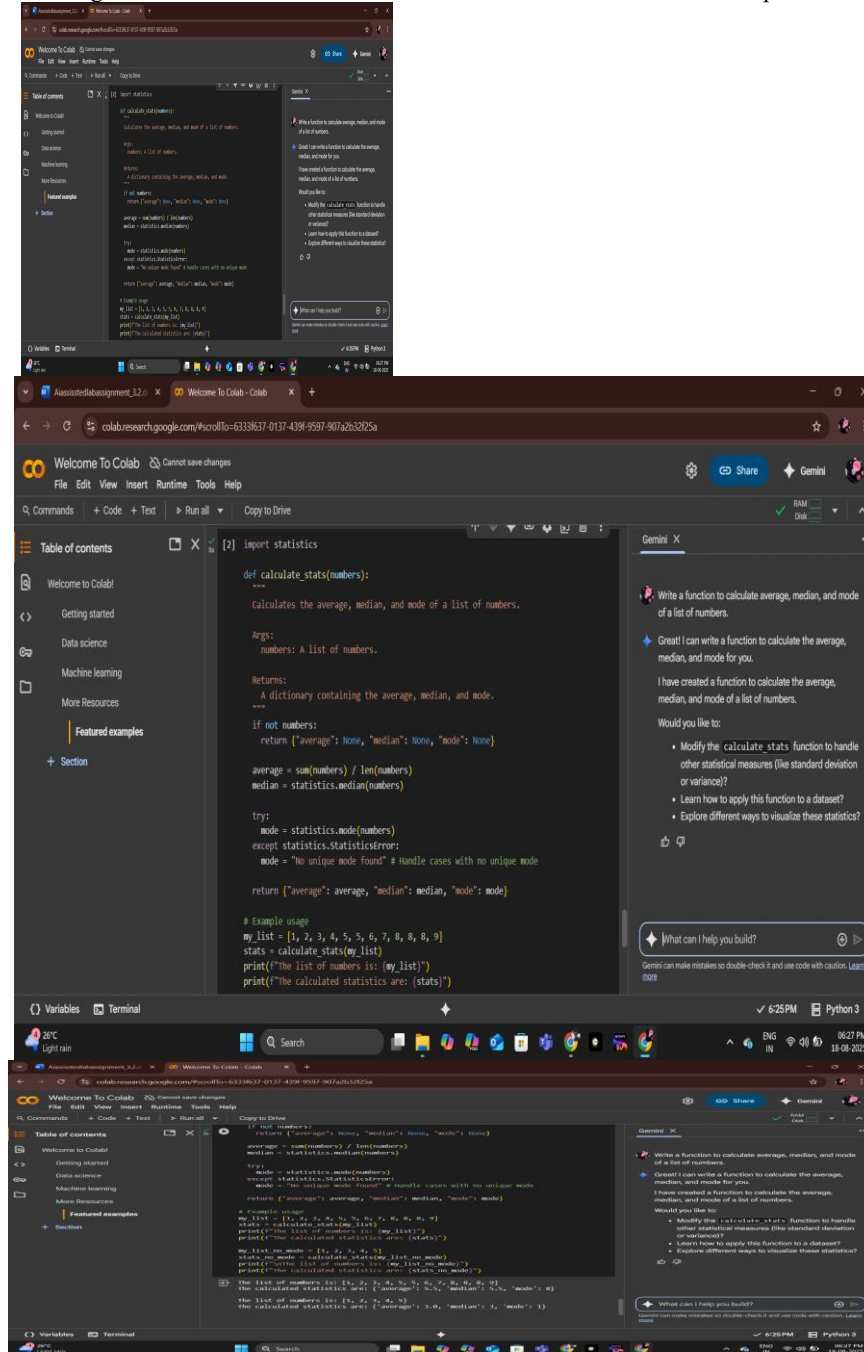


Task Description#2

- Do math stuff, then refine it to: # Write a function to calculate average, median, and mode of a list of numbers.

Expected Output#2

- AI-generated function evolves from unclear to accurate multi-statistical operation.

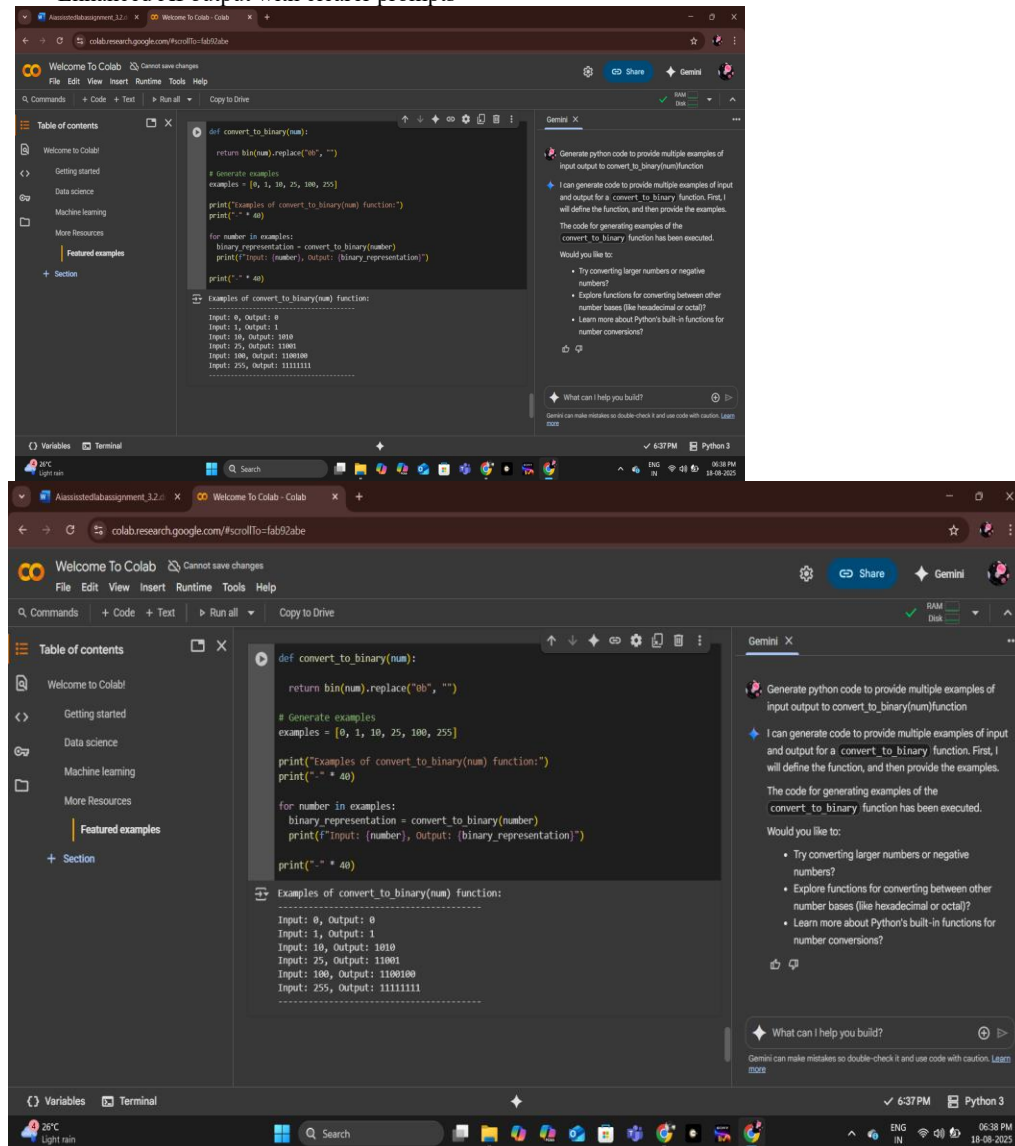


Task Description#3

- Provide multiple examples of input-output to the AI for `convert_to_binary(num)` function. Observe how AI uses few-shot prompting to generalize.

Expected Output#3

- Enhanced AI output with clearer prompts



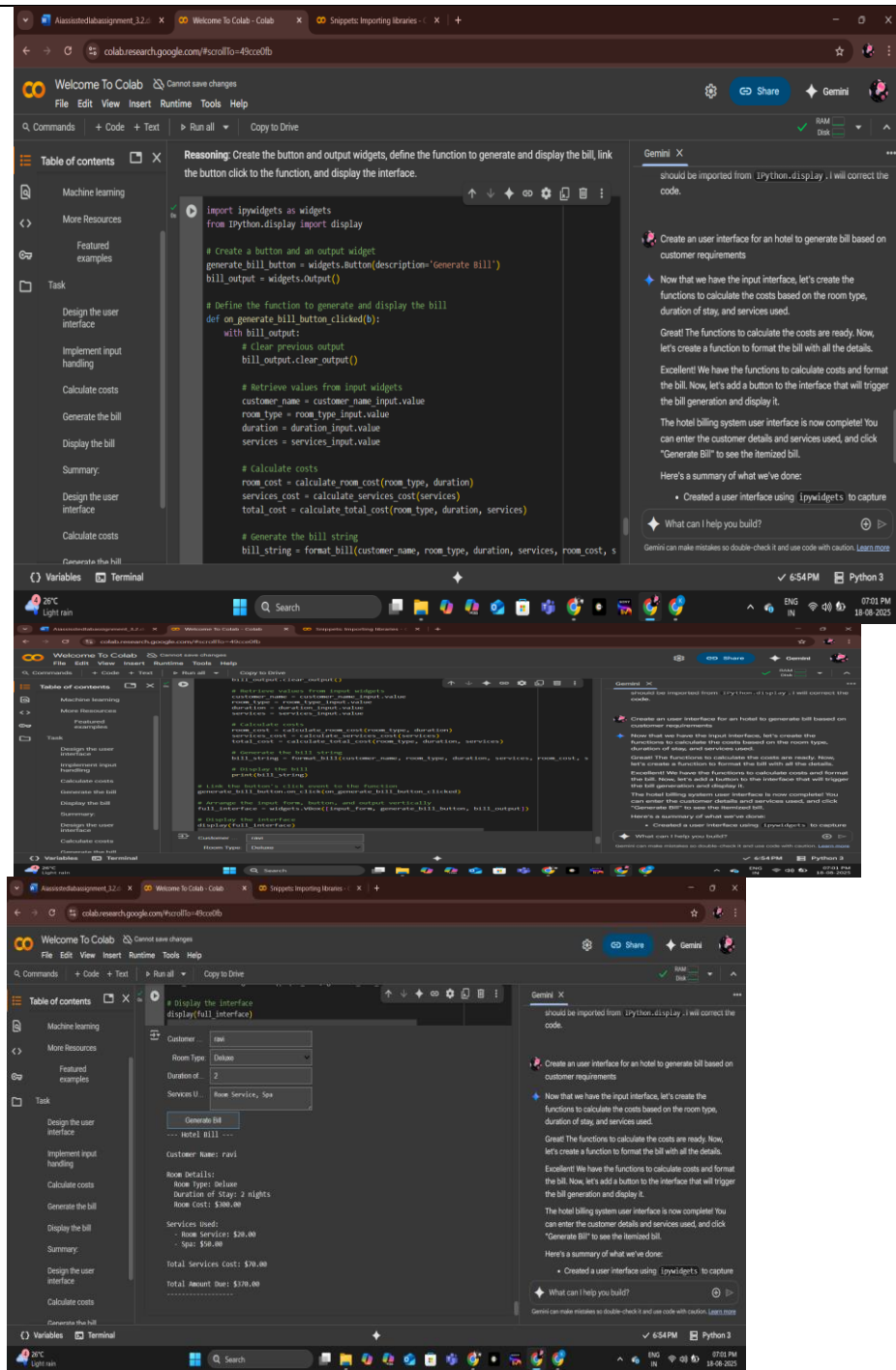
Task Description#4

- Create an user interface for an hotel to generate bill based on customer requirements

Expected Output#4

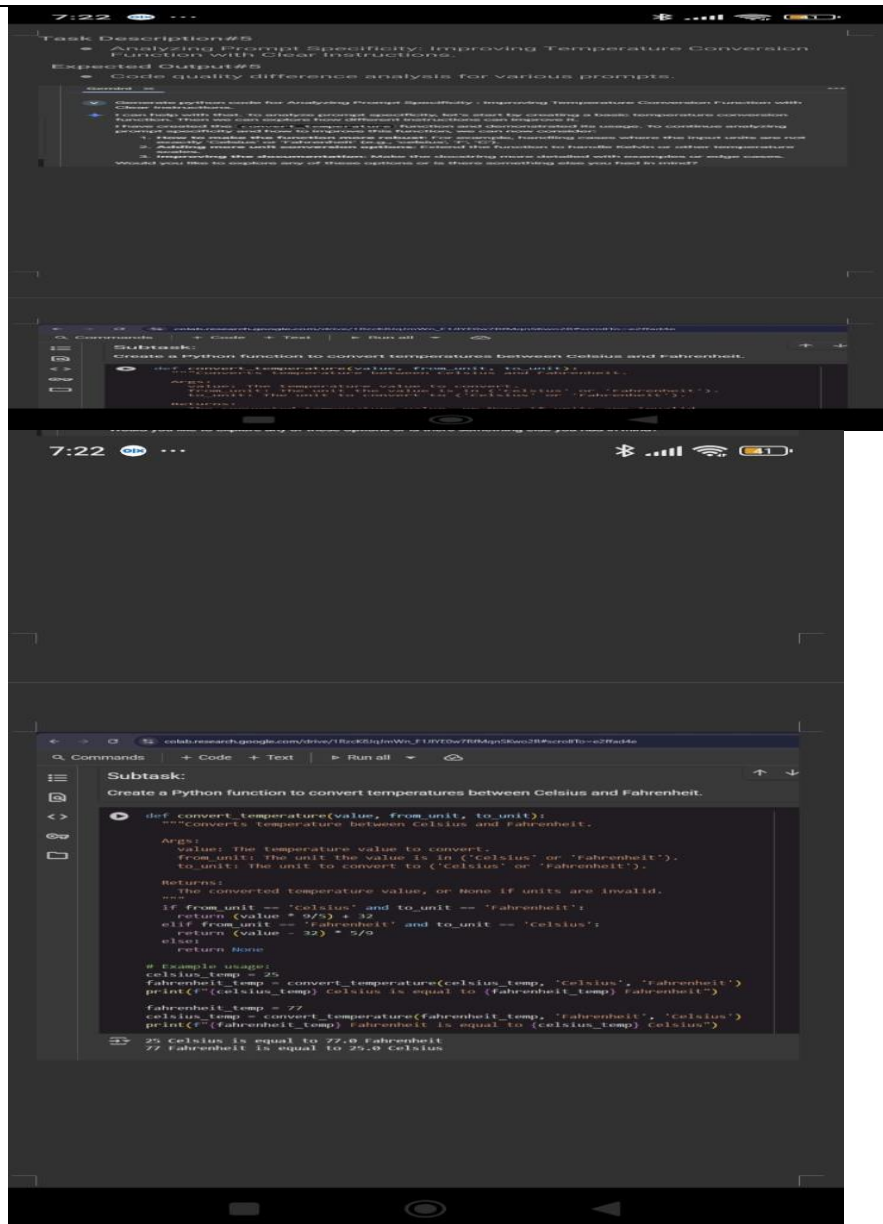
- Consistent functions with shared logic





Task Description#5

- Analyzing Prompt Specificity: Improving Temperature Conversion Function with Clear Instructions
- Expected Output#5
- Code quality difference analysis for various prompts



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

Evaluation Criteria:

Criteria	Max Marks
Task#1	0.5
Task#2	0.5
Task #3	0.5
Task #4	0.5
Task #5	0.5
Total	2.5 Marks