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 - Thursday	**Time(s)**	
Duration	2 Hours	**Applicable to Batches**	

AssignmentNumber:10.4(Present assignment number)/**24**(Total number of assignments)

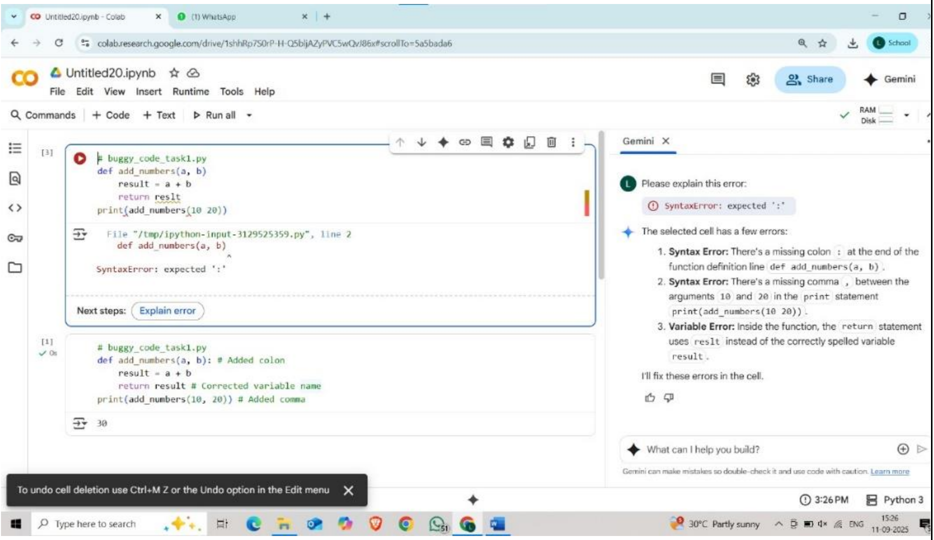
Q.No.	**Question**	***Expected Time to complete***
1	**Lab 10 – Code Review and Quality: Using AI to Improve Code Quality and Readability** **Lab Objectives** • Use AI for automated code review and quality enhancement. • Identify and fix syntax, logical, performance, and security issues in Python code.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

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

Task 1: Syntax and Error Detection

Task: Identify and fix syntax, indentation, and variable errors in the given script.

```
# buggy_code_task1.py
def add_numbers(a, b)
    result = a + b
    return reslt
print(add_numbers(10 20))
```



```
[3]
# buggy_code_task1.py
def add_numbers(a, b)
    result = a + b
    return reslt
print(add_numbers(10 20))
```

```
  File "/tmp/ipython-input-3129525359.py", line 2
    def add_numbers(a, b)
                         ^
SyntaxError: expected ':'
```

Next steps: Explain error

```
[1]
# buggy_code_task1.py
def add_numbers(a, b): # Added colon
    result = a + b
    return result # Corrected variable name
print(add_numbers(10, 20)) # Added comma
```

```
30
```

Gemini

Please explain this error:

`SyntaxError: expected ':'`

The selected cell has a few errors:

1. **Syntax Error:** There's a missing colon `:` at the end of the function definition line `def add_numbers(a, b)`.
2. **Syntax Error:** There's a missing comma `,` between the arguments `10` and `20` in the `print` statement `print(add_numbers(10 20))`.
3. **Variable Error:** Inside the function, the `return` statement uses `reslt` instead of the correctly spelled variable `result`.

I'll fix these errors in the cell.

What can I help you build?

Gemini can make mistakes so double-check it and use code with caution. Learn more

To undo cell deletion use Ctrl+M Z or the Undo option in the Edit menu

Expected Output:

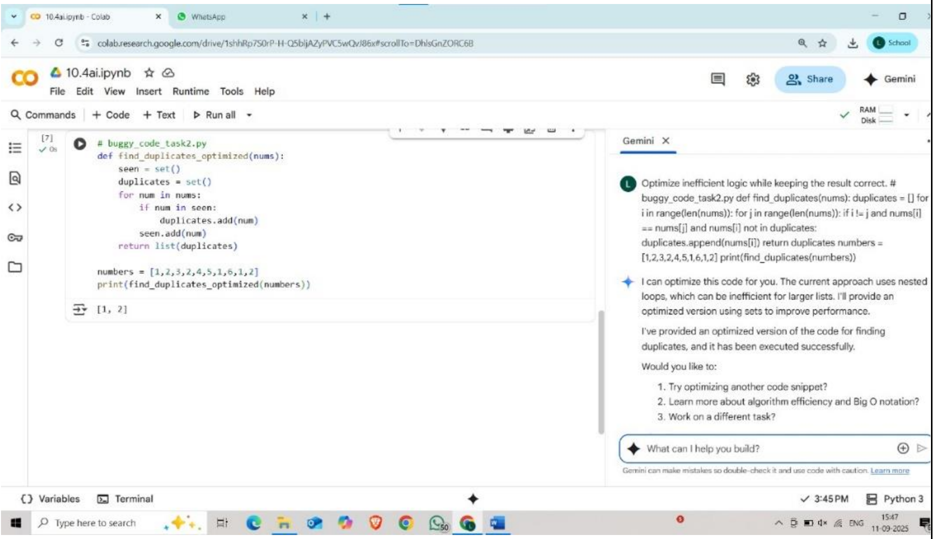
- Corrected code with proper syntax (: after function, fixed variable name, corrected function call).
- AI should explain what was fixed.

Task 2: Logical and Performance Issue Review

Task: Optimize inefficient logic while keeping the result correct.

```
# buggy_code_task2.py
def find_duplicates(nums):
    duplicates = []
    for i in range(len(nums)):
        for j in range(len(nums)):
            if i != j and nums[i] == nums[j] and nums[i] not in duplicates:
                duplicates.append(nums[i])
    return duplicates
numbers = [1,2,3,2,4,5,1,6,1,2]
```

```
print(find_duplicates(numbers))
```


10.4ai.ipynb - Colab
WhatsApp
colab.research.google.com/drive/1shhRp7S0rP-H-Q5bljAZyPVC5wQvJ86x#scrollTo=DhlsGnZORC6B
School
10.4ai.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands
+ Code
+ Text
Run all
RAM
Disk
[7]
0s

```
# buggy_code_task2.py
def find_duplicates_optimized(nums):
    seen = set()
    duplicates = set()
    for num in nums:
        if num in seen:
            duplicates.add(num)
        seen.add(num)
    return list(duplicates)

numbers = [1,2,3,2,4,5,1,6,1,2]
print(find_duplicates_optimized(numbers))
```

```
[1, 2]
```

Optimize inefficient logic while keeping the result correct. # buggy_code_task2.py def find_duplicates(nums): duplicates = [] for i in range(len(nums)): for j in range(len(nums)): if i != j and nums[i] == nums[j] and nums[i] not in duplicates: duplicates.append(nums[i]) return duplicates numbers = [1,2,3,2,4,5,1,6,1,2] print(find_duplicates(numbers))

I can optimize this code for you. The current approach uses nested loops, which can be inefficient for larger lists. I'll provide an optimized version using sets to improve performance.

I've provided an optimized version of the code for finding duplicates, and it has been executed successfully.

Would you like to:

1. Try optimizing another code snippet?
2. Learn more about algorithm efficiency and Big O notation?
3. Work on a different task?

What can I help you build?
Gemini can make mistakes so double-check it and use code with caution. Learn more
Variables
Terminal
3:45 PM
Python 3
Type here to search
ENG
15:47
11-09-2025

Expected Output:

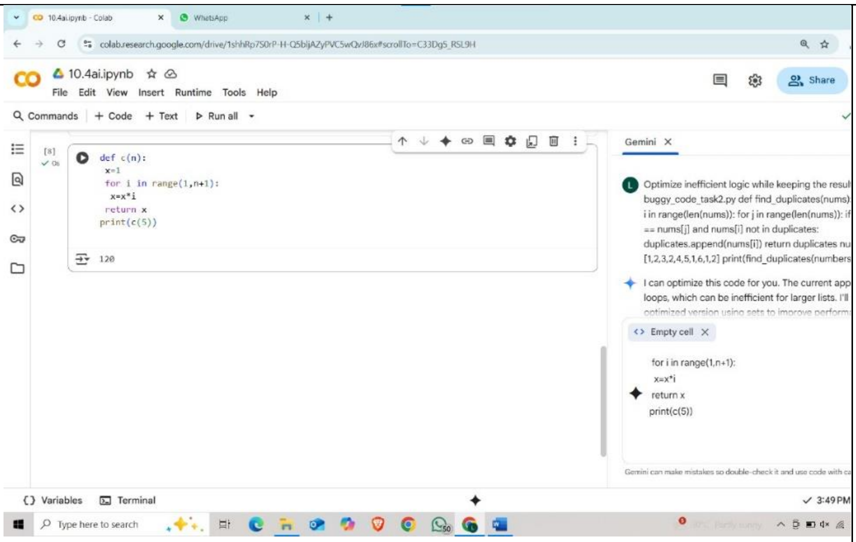
- More efficient duplicate detection (e.g., using sets).
- AI should explain the optimization.

Task 3: Code Refactoring for Readability

Task: Refactor messy code into clean, PEP 8–compliant, well-structured code.

```
# buggy_code_task3.py

def c(n):
 x=1
 for i in range(1,n+1):
  x=x*i
 return x
print(c(5))
```


10.4ai.ipynb - Colab
WhatsApp
colab.research.google.com/drive/1shhRp7S0rP-H-Q5bljAZyPVC5wQvJ86x#scrollTo=C33Dg5_RSL9H
10.4ai.ipynb
File Edit View Insert Runtime Tools Help
Share
Commands
+ Code
+ Text
Run all
Gemini
Variables
Terminal
3:49 PM
Type here to search

```
[8]
def c(n):
 x=1
 for i in range(1,n+1):
  x=x*i
 return x
print(c(5))
```

```
120
```

Optimize inefficient logic while keeping the resul
buggy_code_task2.py def find_duplicates(nums)
i in range(len(nums)): for j in range(len(nums)): if
== nums[j] and nums[i] not in duplicates:
duplicates.append(nums[i]) return duplicates nu
[1,2,3,2,4,5,1,6,1,2] print(find_duplicates(numbers

I can optimize this code for you. The current app
loops, which can be inefficient for larger lists. I'll
optimized version using sets to improve perform

Empty cell

```
for i in range(1,n+1):
 x=x*i
return x
print(c(5))
```

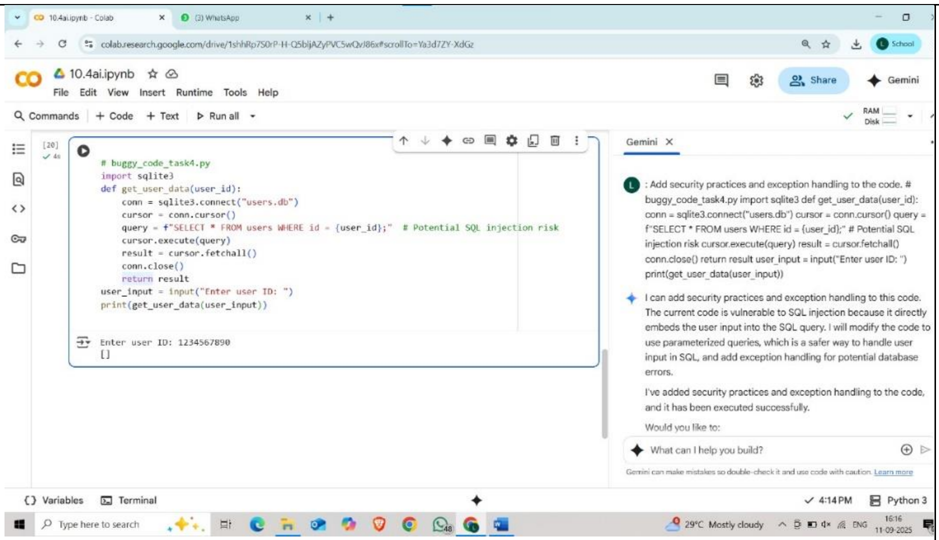
Gemini can make mistakes so double-check it and use code with ca

Expected Output:
Function renamed to calculate_factorial.
Proper indentation, variable naming, docstrings, and formatting.
AI should provide a more readable version.

Task 4: Security and Error Handling Enhancement

Task: Add security practices and exception handling to the code.

```
# buggy_code_task4.py
import sqlite3
def get_user_data(user_id):
    conn = sqlite3.connect("users.db")
    cursor = conn.cursor()
    query = f"SELECT * FROM users WHERE id = {user_id};"  # Potential SQL injection risk
    cursor.execute(query)
    result = cursor.fetchall()
    conn.close()
    return result
user_input = input("Enter user ID: ")
print(get_user_data(user_input))
```


10.4ai.ipynb - Colab
(3) WhatsApp
colab.research.google.com/drive/1shhRp7S0rP-H-Q5bijAZyPVC5wQvJ86x#scrollTo=Ya3d7ZY-XdGz
School
10.4ai.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all
Share
Gemini
RAM
Disk
[20]
4s

```
# buggy_code_task4.py
import sqlite3
def get_user_data(user_id):
    conn = sqlite3.connect("users.db")
    cursor = conn.cursor()
    query = f"SELECT * FROM users WHERE id = {user_id};"  # Potential SQL injection risk
    cursor.execute(query)
    result = cursor.fetchall()
    conn.close()
    return result
user_input = input("Enter user ID: ")
print(get_user_data(user_input))
```

```
Enter user ID: 1234567890
[]
```

: Add security practices and exception handling to the code. # buggy_code_task4.py import sqlite3 def get_user_data(user_id): conn = sqlite3.connect("users.db") cursor = conn.cursor() query = f"SELECT * FROM users WHERE id = {user_id};" # Potential SQL injection risk cursor.execute(query) result = cursor.fetchall() conn.close() return result user_input = input("Enter user ID: ") print(get_user_data(user_input))

I can add security practices and exception handling to this code. The current code is vulnerable to SQL injection because it directly embeds the user input into the SQL query. I will modify the code to use parameterized queries, which is a safer way to handle user input in SQL, and add exception handling for potential database errors.

I've added security practices and exception handling to the code, and it has been executed successfully.

Would you like to:

What can I help you build?
Gemini can make mistakes so double-check it and use code with caution. Learn more
Variables
Terminal
4:14 PM
Python 3
Type here to search
29°C Mostly cloudy
ENG
16:16
11-09-2025

Expected Output:

Safe query using parameterized SQL (? placeholders).

Try-except block for database errors.

Input validation before query execution.

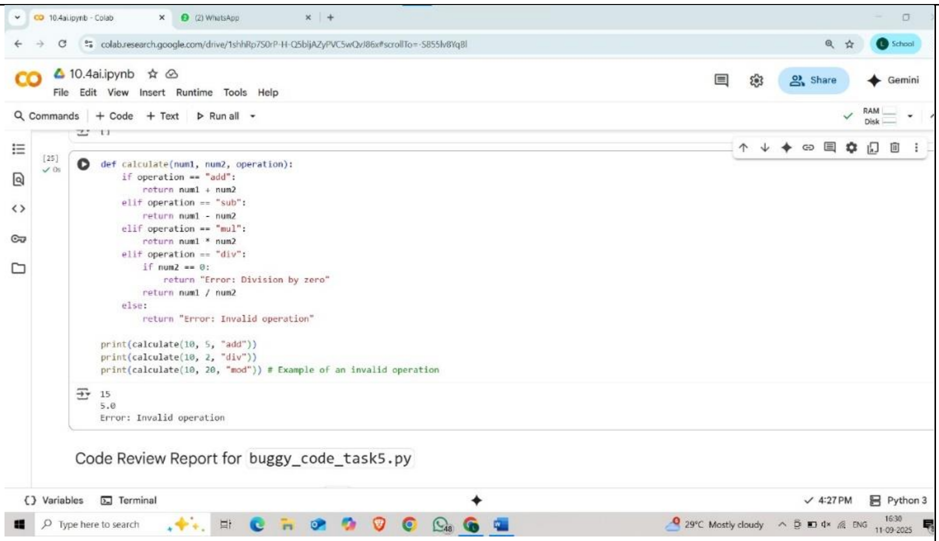
Task 5: Automated Code Review Report Generation

Task: Generate a **review report** for this messy code.

```
# buggy_code_task5.py

def calc(x,y,z):
 if z=="add":
  return x+y
 elif z=="sub": return x-y
 elif z=="mul":
  return x*y
 elif z=="div":
  return x/y
 else: print("wrong")

print(calc(10,5,"add"))
print(calc(10,0,"div"))
```



```python
def calculate(num1, num2, operation):
    if operation == "add":
        return num1 + num2
    elif operation == "sub":
        return num1 - num2
    elif operation == "mul":
        return num1 * num2
    elif operation == "div":
        if num2 == 0:
            return "Error: Division by zero"
        return num1 / num2
    else:
        return "Error: Invalid operation"

print(calculate(10, 5, "add"))
print(calculate(10, 2, "div"))
print(calculate(10, 20, "mod")) # Example of an invalid operation
```

```
15
5.0
Error: Invalid operation
```

Code Review Report for `buggy_code_task5.py`

Expected Output:

AI-generated **review report** should mention:

- Missing docstrings
- Inconsistent formatting (indentation, inline return)
- Missing error handling for division by zero
- Non-descriptive function/variable names
- Suggestions for readability and PEP 8 compliance