Software Construction and Development

Bachelor OF Science in Computer Science
Session (2022-2026 Fall)

Submitted by:

Name	Roll No	Contact No:	Emails
Muhammad Ehsan	(B-27918)	0323-4119975	b-27918@student.usa.edu.pk

Lab Supervisor:

Supervisor:	Miss Sawera

Lecturer:

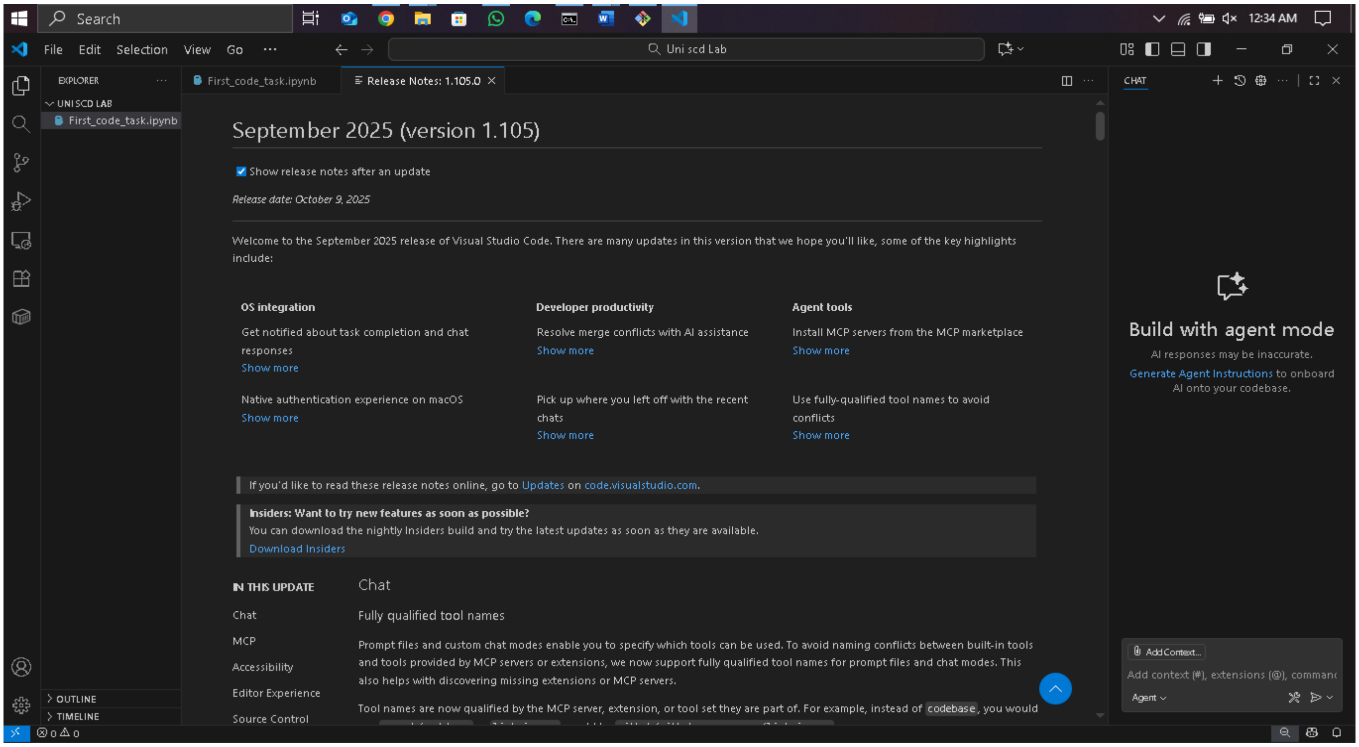
Lecturer Name:	Mr. Zain ul Abideen

Contents

Experiment 1: Lab Orientation and Setup

Take a **screenshot** of:

- Your IDE open (like VS Code)



September 2025 (version 1.105)

☑ Show release notes after an update

Release date: October 9, 2025

Welcome to the September 2025 release of Visual Studio Code. There are many updates in this version that we hope you'll like, some of the key highlights include:

OS integration

Get notified about task completion and chat responses
Show more

Native authentication experience on macOS
Show more

Developer productivity

Resolve merge conflicts with AI assistance
Show more

Pick up where you left off with the recent chats
Show more

Agent tools

Install MCP servers from the MCP marketplace
Show more

Use fully-qualified tool names to avoid conflicts
Show more

If you'd like to read these release notes online, go to Updates on code.visualstudio.com.

Insiders: Want to try new features as soon as possible?
You can download the nightly Insiders build and try the latest updates as soon as they are available.
Download Insiders

IN THIS UPDATE

Chat
MCP
Accessibility
Editor Experience
Source Control

Chat

Fully qualified tool names

Prompt files and custom chat modes enable you to specify which tools can be used. To avoid naming conflicts between built-in tools and tools provided by MCP servers or extensions, we now support fully qualified tool names for prompt files and chat modes. This also helps with discovering missing extensions or MCP servers.

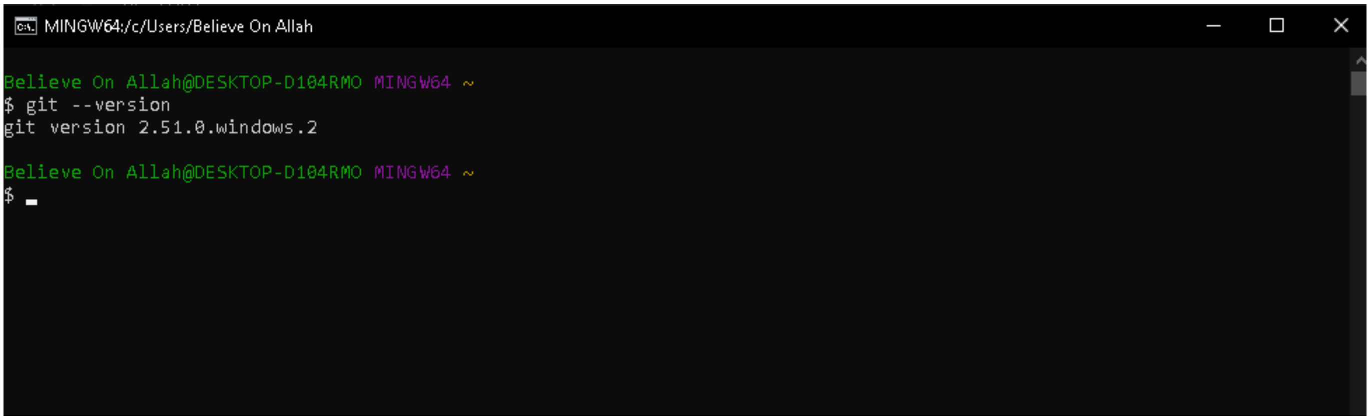
Tool names are now qualified by the MCP server, extension, or tool set they are part of. For example, instead of `codebase`, you would

Build with agent mode

AI responses may be inaccurate.

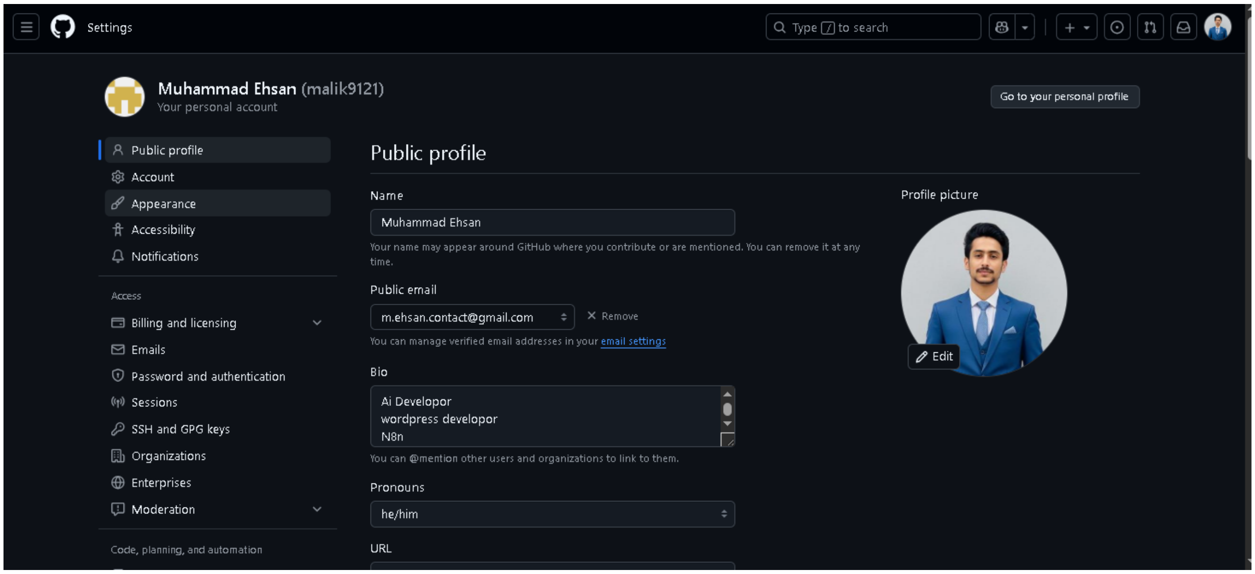
Generate Agent Instructions to onboard AI onto your codebase.

1.1 Your Git version showing in terminal



```
Believe On Allah@DESKTOP-D104RMO MINGW64 ~
$ git --version
git version 2.51.0.windows.2

Believe On Allah@DESKTOP-D104RMO MINGW64 ~
$ _
```

1.2 Your GitHub profile page



Muhammad Ehsan (malik9121)
Your personal account

Public profile

Name
Muhammad Ehsan
Your name may appear around GitHub where you contribute or are mentioned. You can remove it at any time.

Public email
m.ehsan.contact@gmail.com × Remove
You can manage verified email addresses in your email settings

Bio
Ai Developer
wordpress developor
N8n
You can @mention other users and organizations to link to them.

Pronouns
he/him

URL

Experiment 2: Introduction to Git

Objective: Learn basic Git commands.

Tasks:

git init, add, commit, push, clone

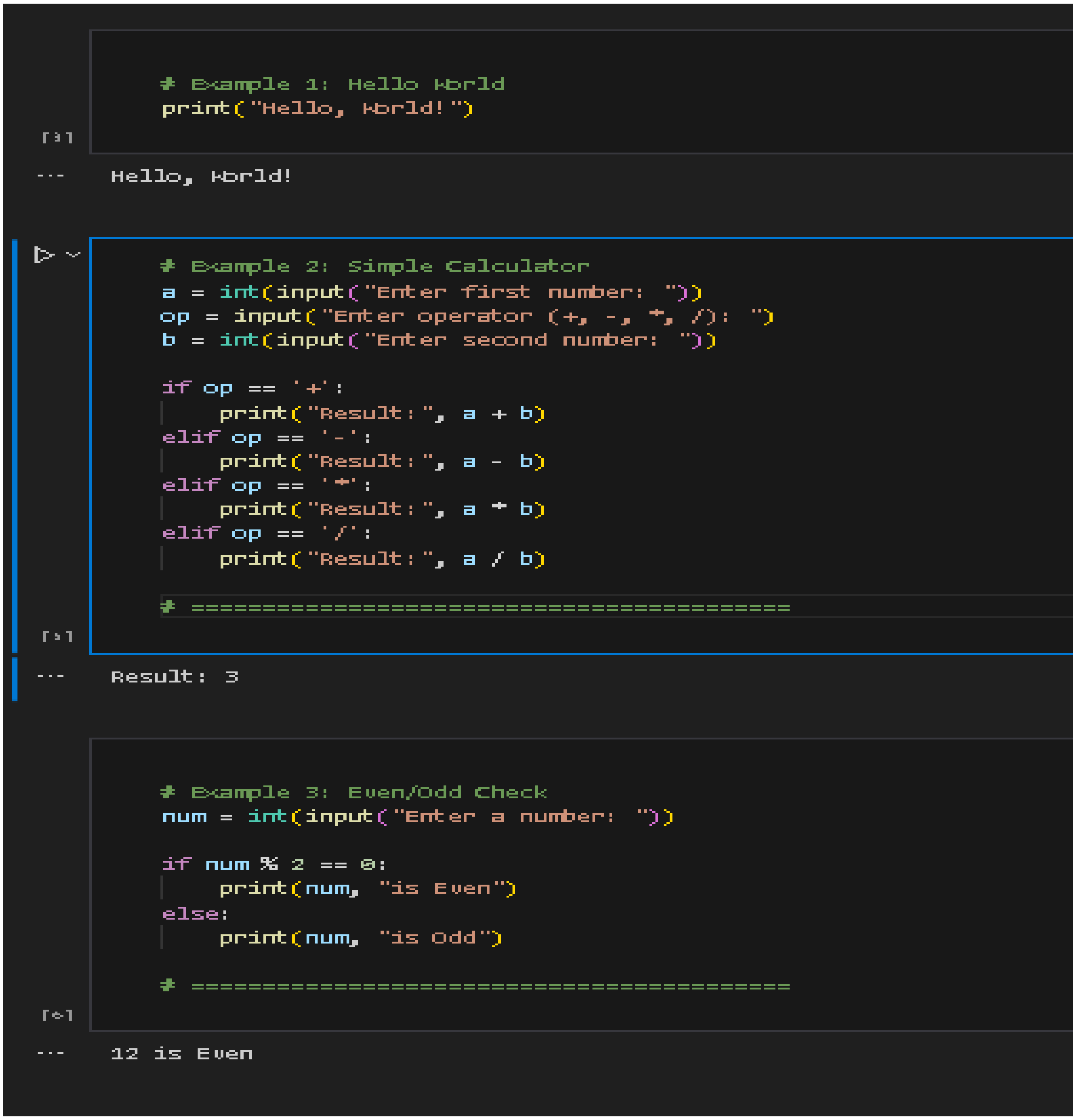
2.1 Working Tasks:

This test checks your understanding of basic Git commands used for version control. It covers initializing a repository with git init, adding files using git add, saving changes through git commit, and uploading projects to a remote repository with git push. It also includes the use of git clone for copying repositories from remote sources. The test evaluates your ability to manage, track, and share code using Git effectively.

2.2 Software Use:

VS-CODE that are link with Github.

2.3 Inputs & Outputs:



```
# Example 1: Hello World
print("Hello, World!")
```

```
Hello, World!
```

```
# Example 2: Simple Calculator
a = int(input("Enter first number: "))
op = input("Enter operator (+, -, *, /): ")
b = int(input("Enter second number: "))

if op == '+':
    print("Result:", a + b)
elif op == '-':
    print("Result:", a - b)
elif op == '*':
    print("Result:", a * b)
elif op == '/':
    print("Result:", a / b)

# ==========================================
```

```
Result: 3
```

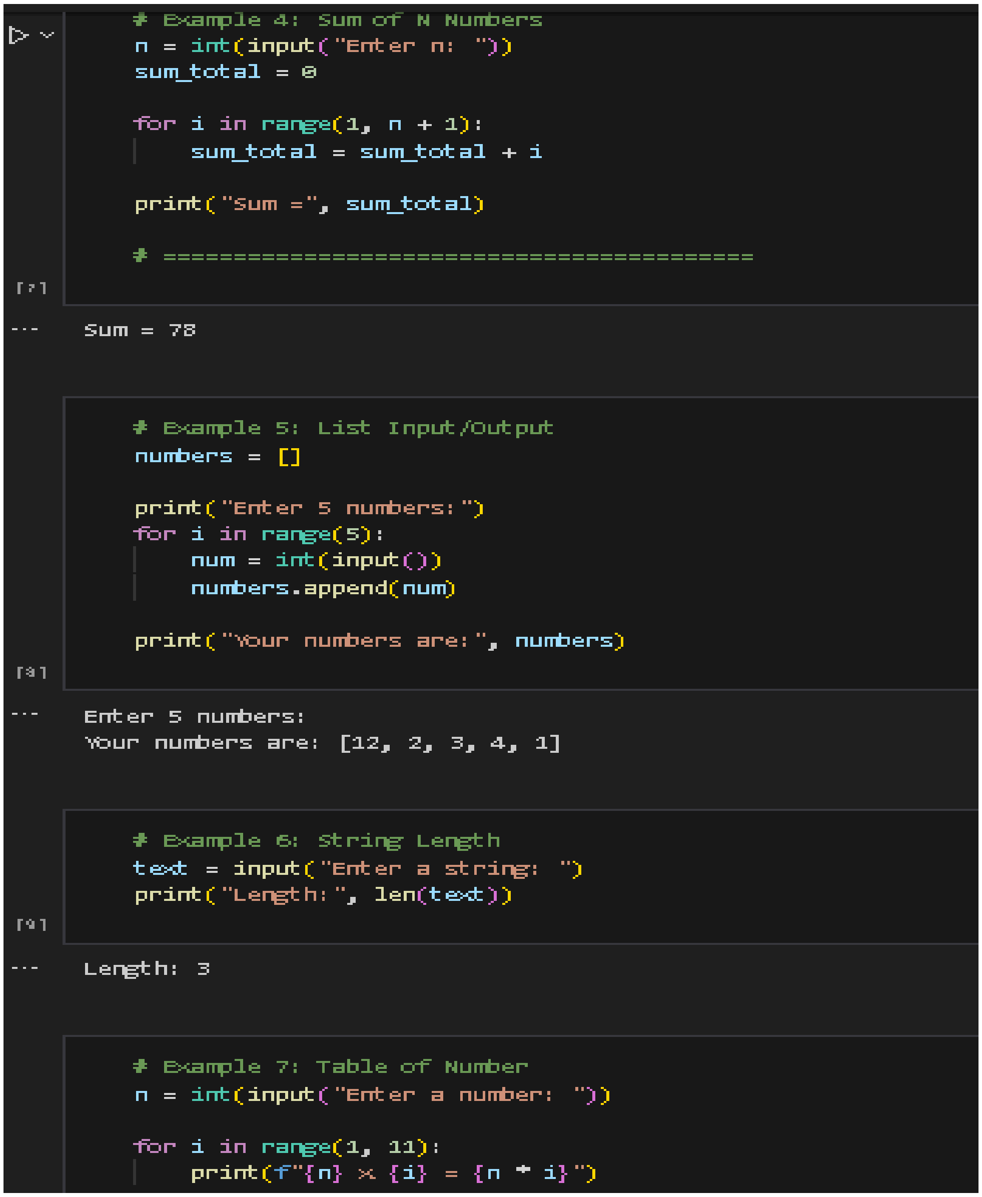
```
# Example 3: Even/Odd Check
num = int(input("Enter a number: "))

if num % 2 == 0:
    print(num, "is Even")
else:
    print(num, "is Odd")

# ==========================================
```

```
12 is Even
```

2.4 Explainable:

- ✓ This Python script includes three simple examples.
- ✓ The first program prints “Hello, World!” to show basic output.
- ✓ The second example is a simple calculator that performs addition, subtraction, multiplication, or division based on user input.
- ✓ The third program checks whether a given number is even or odd using the modulus operator.
- ✓ Together, these examples demonstrate basic input/output, conditional statements, and arithmetic operations in Python.

2.5 Inputs & Outputs:



```
# Example 4: Sum of N Numbers
n = int(input("Enter n: "))
sum_total = 0

for i in range(1, n + 1):
    sum_total = sum_total + i

print("Sum =", sum_total)

# ==========================================
```

```
Sum = 78
```

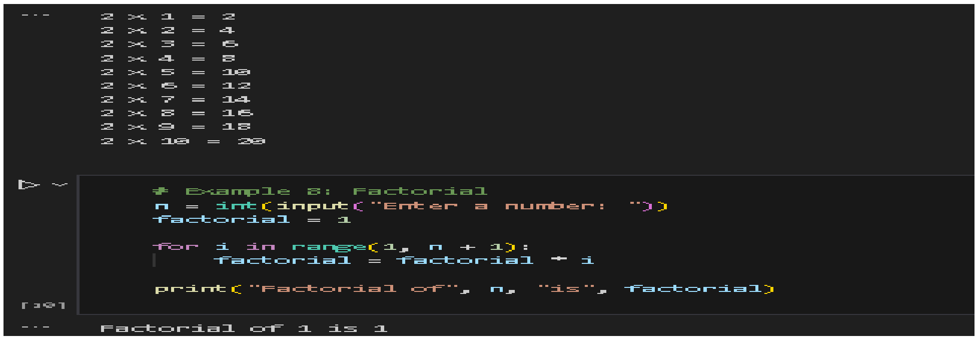
```
# Example 5: List Input/Output
numbers = []

print("Enter 5 numbers:")
for i in range(5):
    num = int(input())
    numbers.append(num)

print("Your numbers are:", numbers)
```

```
Enter 5 numbers:
Your numbers are: [12, 2, 3, 4, 1]
```

```
# Example 6: String Length
text = input("Enter a string: ")
print("Length:", len(text))
```

```
Length: 3
```

```
# Example 7: Table of Number
n = int(input("Enter a number: "))

for i in range(1, 11):
    print(f"{n} x {i} = {n * i}")
```



```
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18
2 x 10 = 20
```

```
# Example 8: Factorial
n = int(input("Enter a number: "))
factorial = 1

for i in range(1, n + 1):
    factorial = factorial * i

print("Factorial of", n, "is", factorial)
```

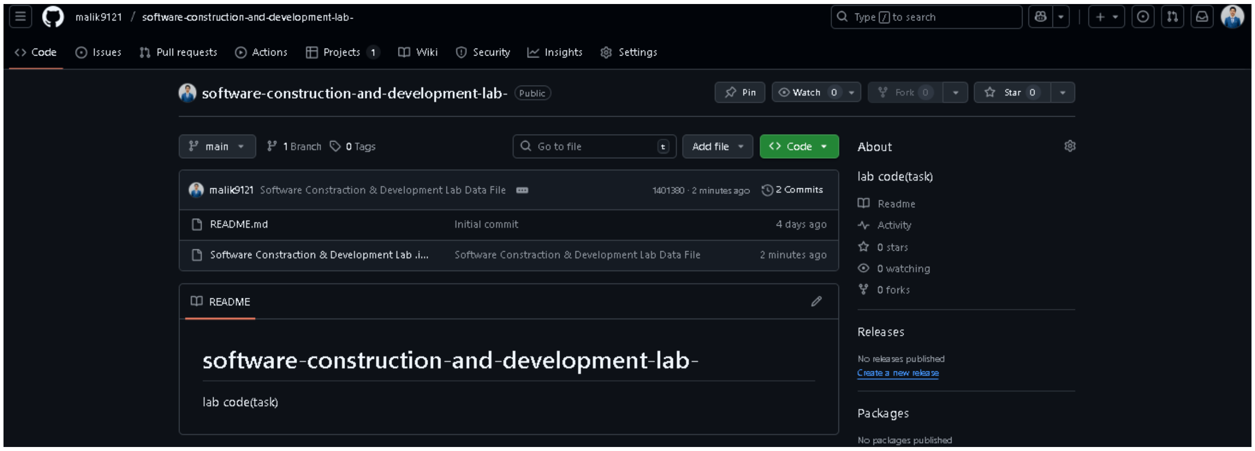
```
Factorial of 1 is 1
```

2.6 Explainable Upper-Code :

- ✓ This code contains several beginner-level Python examples.

- ✓ Example 4 calculates the sum of the first *n* natural numbers using a for loop.
- ✓ Example 5 takes five numbers from the user, stores them in a list, and prints the list.
- ✓ Example 6 finds the length of a string using the len() function.
- ✓ Example 7 displays the multiplication table of a given number.
- ✓ Example 8 calculates the factorial of a number using a loop. Together, these examples cover loops, lists, strings, and basic arithmetic logic.

2.7 Final Push/Upload On Github:



software-construction-and-development-lab- Public

malik9121 Software Constraction & Development Lab Data File 1401380 · 2 minutes ago 2 Commits

README.md	Initial commit	4 days ago
Software Constraction & Development Lab .i...	Software Constraction & Development Lab Data File	2 minutes ago

README

software-construction-and-development-lab-

lab code(task)

About

lab code(task)

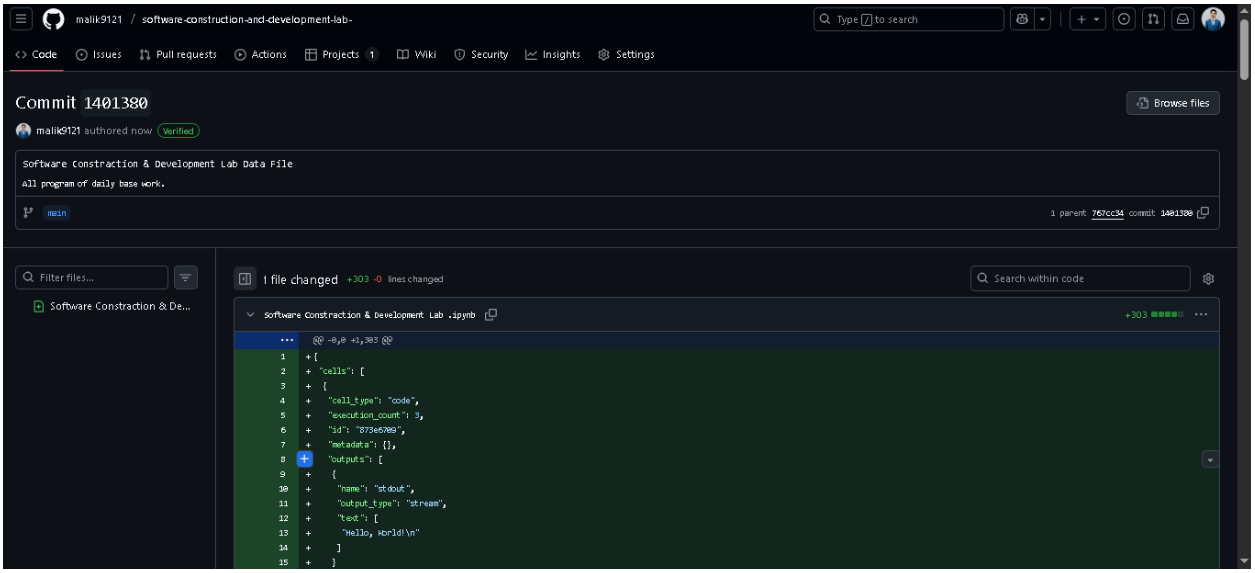
Readme
Activity
0 stars
0 watching
0 forks

Releases

No releases published
Create a new release

Packages

No packages published



Commit 1401380

malik9121 authored now Verified

Software Constraction & Development Lab Data File

All program of daily base work.

main 1 parent 767cc34 commit 1401380

1 file changed +303 -0 lines changed

Software Constraction & De...

Software Constraction & Development Lab .ipynb +303

```
@@ -0,0 +1,303 @@
+{
+ "cells": [
+  {
+   "cell_type": "code",
+   "execution_count": 3,
+   "id": "873e6709",
+   "metadata": {},
+   "outputs": [
+    {
+     "name": "stdout",
+     "output_type": "stream",
+     "text": [
+      "Hello, World!\n"
+     ]
+    }
```

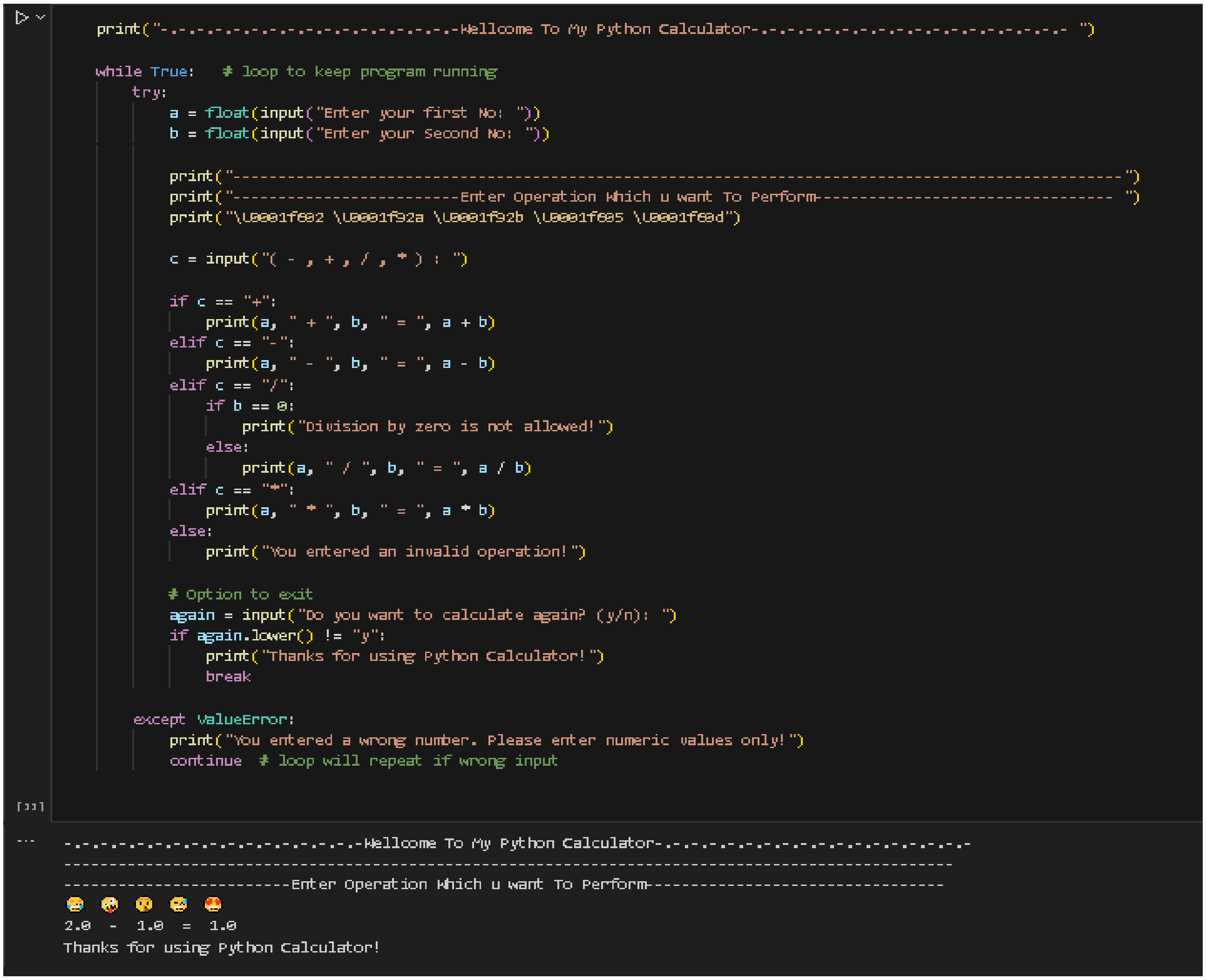
Experiment 3: Modular Programming

Objective: Organize code into modules/packages.

Tasks:

Write a simple calculator program using modules

3.1 Inputs & Outputs:



```
print("-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-Wellcome To My Python Calculator-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.- ")

while True:   # loop to keep program running
    try:
        a = float(input("Enter your first No: "))
        b = float(input("Enter your Second No: "))

        print("--------------------------------------------------------------------------------------------------")
        print("-------------------------Enter Operation Which u want To Perform-------------------------------- ")
        print("\U0001f602 \U0001f92a \U0001f92b \U0001f605 \U0001f60d")

        c = input("( - , + , / , * ) : ")

        if c == "+":
            print(a, " + ", b, " = ", a + b)
        elif c == "-":
            print(a, " - ", b, " = ", a - b)
        elif c == "/":
            if b == 0:
                print("Division by zero is not allowed!")
            else:
                print(a, " / ", b, " = ", a / b)
        elif c == "*":
            print(a, " * ", b, " = ", a * b)
        else:
            print("You entered an invalid operation!")

        # Option to exit
        again = input("Do you want to calculate again? (y/n): ")
        if again.lower() != "y":
            print("Thanks for using Python Calculator!")
            break

    except ValueError:
        print("You entered a wrong number. Please enter numeric values only!")
        continue  # loop will repeat if wrong input
```

```
-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-Wellcome To My Python Calculator-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-
--------------------------------------------------------------------------------------------------
-------------------------Enter Operation Which u want To Perform--------------------------------
😂 🤪 🤫 😅 😍
2.0  -  1.0  =  1.0
Thanks for using Python Calculator!
```

3.2 Explainable Upper-Code :

This program is a simple Python calculator that runs continuously until the user decides to exit. It takes two numeric inputs, asks for an operation (+, -, *, /), and displays the result. It also prevents division by zero and handles invalid or non-numeric input using exception handling.The loop allows multiple calculations in one run, and exits only when the user types "n."

🧪 Experiment 4: Writing Clean Code

Objective: Practice coding standards and documentation

Tasks:

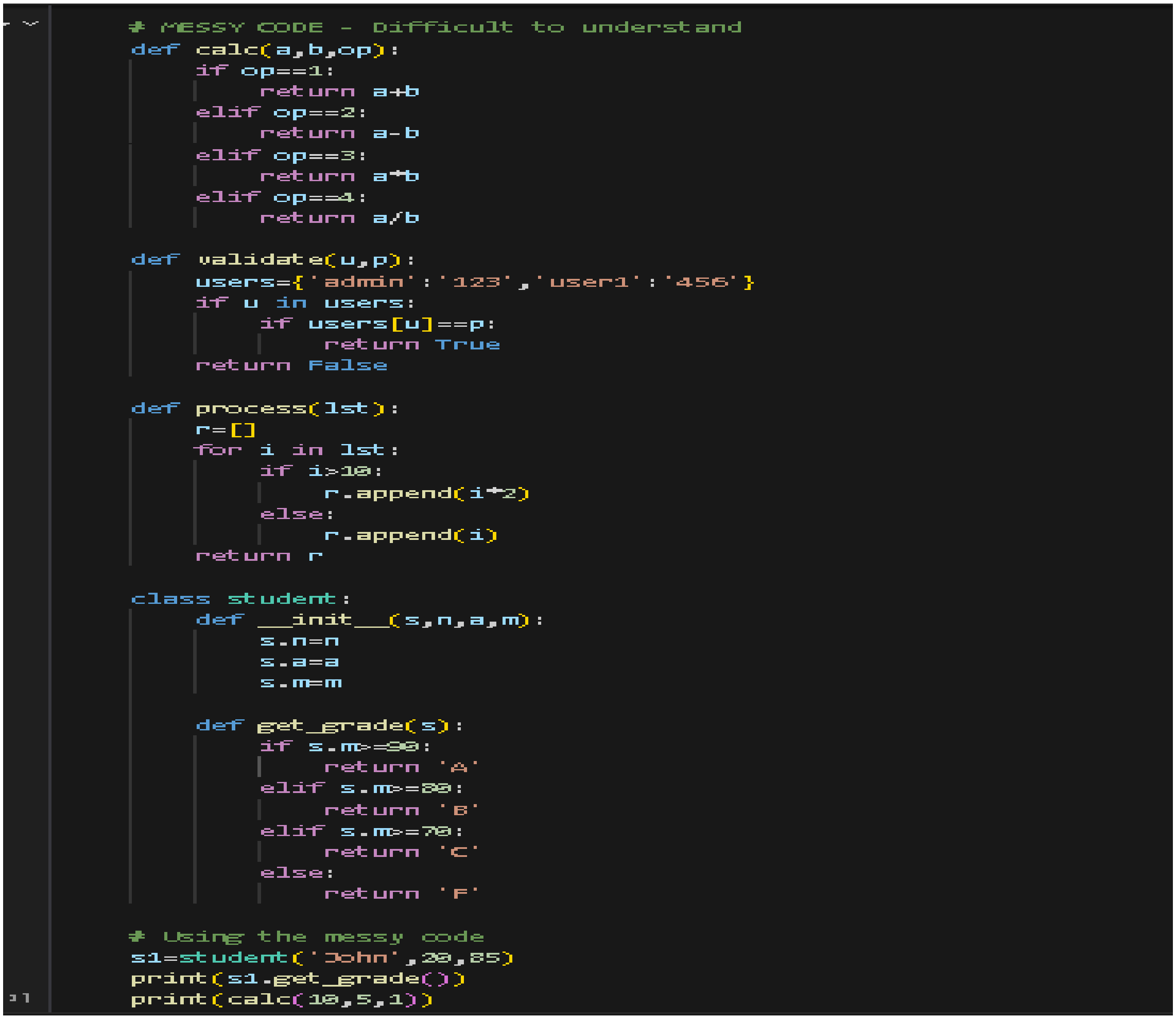
Refactor messy code

Add comments and docstrings

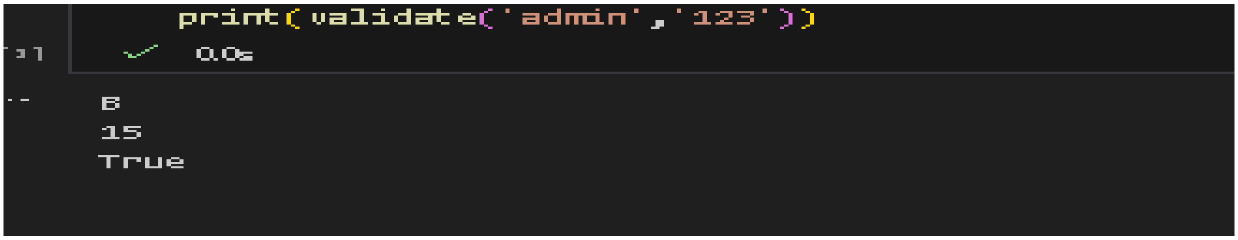
Deliverables: Cleaned and well-commented code

This is messy Code its was difficult to understand:

4.1 Inputs & Outputs:



```python
# MESSY CODE - Difficult to understand
def calc(a,b,op):
    if op==1:
        return a+b
    elif op==2:
        return a-b
    elif op==3:
        return a*b
    elif op==4:
        return a/b

def validate(u,p):
    users={'admin':'123','user1':'456'}
    if u in users:
        if users[u]==p:
            return True
    return False

def process(lst):
    r=[]
    for i in lst:
        if i>10:
            r.append(i*2)
        else:
            r.append(i)
    return r

class student:
    def __init__(s,n,a,m):
        s.n=n
        s.a=a
        s.m=m

    def get_grade(s):
        if s.m>=90:
            return 'A'
        elif s.m>=80:
            return 'B'
        elif s.m>=70:
            return 'C'
        else:
            return 'F'

# Using the messy code
s1=student('John',20,85)
print(s1.get_grade())
print(calc(10,5,1))
```

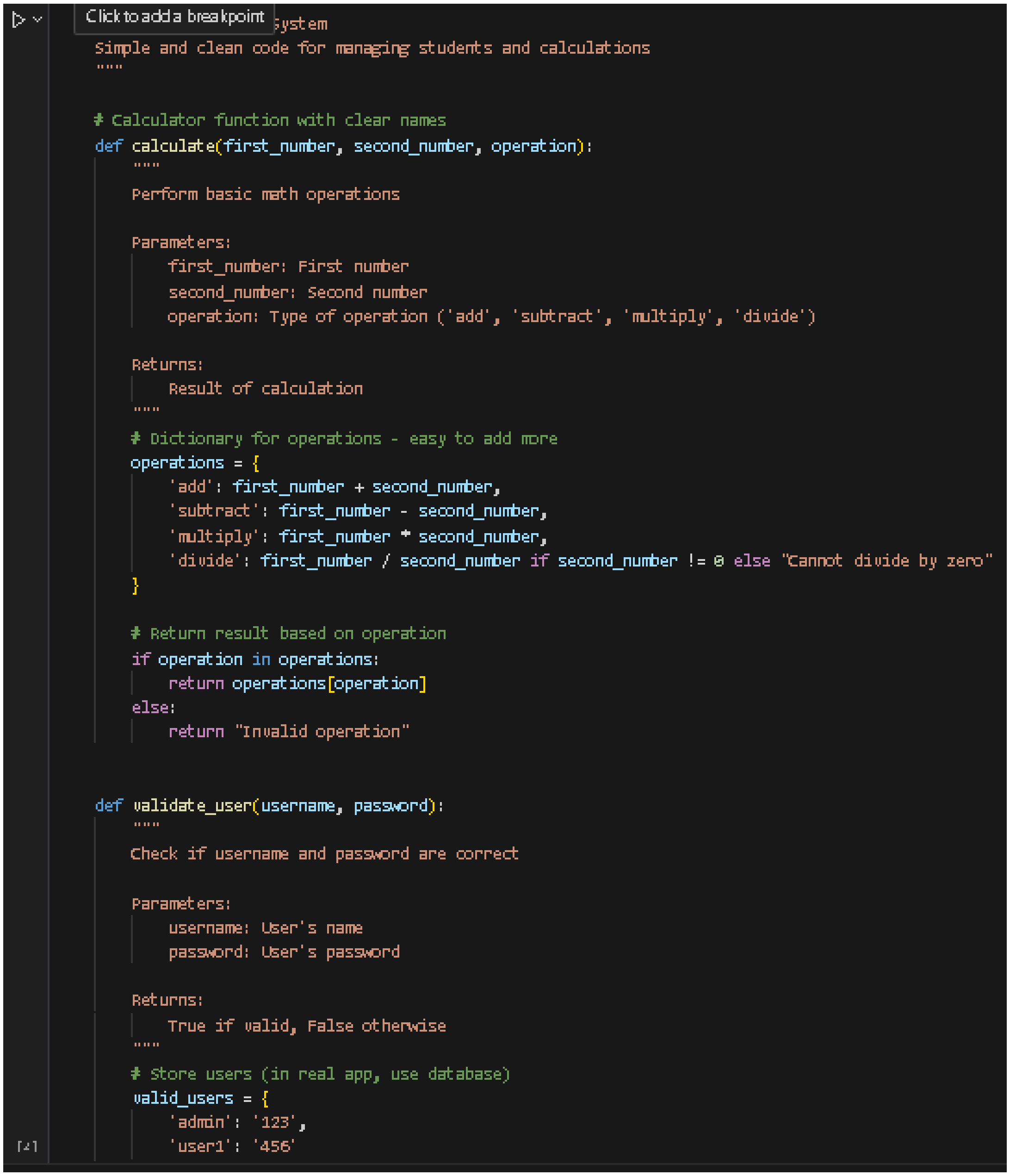


```python
print(validate('admin','123'))
```

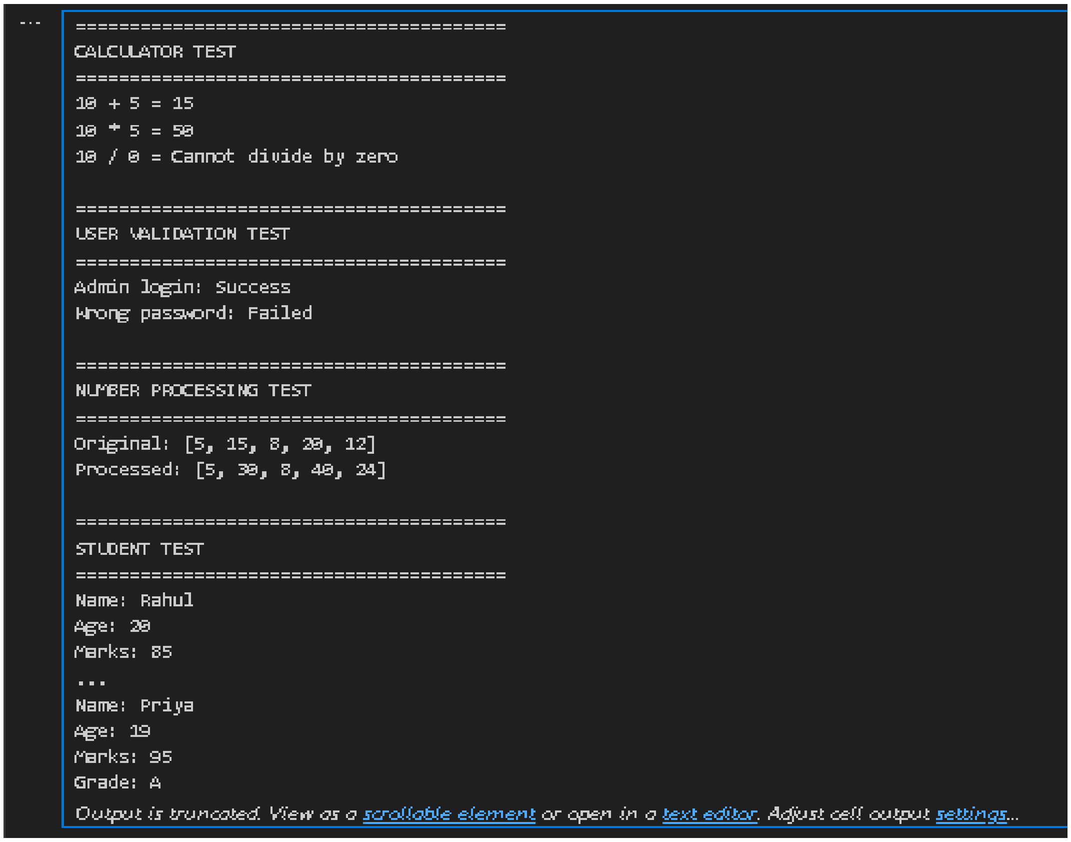
```
B
15
True
```

Upper code is messy Code:

4.2 Inputs & Outputs:



```
System
Simple and clean code for managing students and calculations
"""

# Calculator function with clear names
def calculate(first_number, second_number, operation):
    """
    Perform basic math operations

    Parameters:
        first_number: First number
        second_number: Second number
        operation: Type of operation ('add', 'subtract', 'multiply', 'divide')

    Returns:
        Result of calculation
    """
    # Dictionary for operations - easy to add more
    operations = {
        'add': first_number + second_number,
        'subtract': first_number - second_number,
        'multiply': first_number * second_number,
        'divide': first_number / second_number if second_number != 0 else "Cannot divide by zero"
    }

    # Return result based on operation
    if operation in operations:
        return operations[operation]
    else:
        return "Invalid operation"


def validate_user(username, password):
    """
    Check if username and password are correct

    Parameters:
        username: User's name
        password: User's password

    Returns:
        True if valid, False otherwise
    """
    # Store users (in real app, use database)
    valid_users = {
        'admin': '123',
        'user1': '456'
```



```
========================================
CALCULATOR TEST
========================================
10 + 5 = 15
10 * 5 = 50
10 / 0 = Cannot divide by zero

========================================
USER VALIDATION TEST
========================================
Admin login: Success
Wrong password: Failed

========================================
NUMBER PROCESSING TEST
========================================
Original: [5, 15, 8, 20, 12]
Processed: [5, 30, 8, 40, 24]

========================================
STUDENT TEST
========================================
Name: Rahul
Age: 20
Marks: 85
...
Name: Priya
Age: 19
Marks: 95
Grade: A
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

4.3 Main Improvements:

1. Better Names

- calc → calculate
- s → self
- n → name
- a → age
- m → marks

2. Comments Added

- Function descriptions
- Parameter explanations
- What each section does

3. Better Structure

- Separated code into functions
- Added main() function
- Organized related code together

4. Clear Logic

- Used dictionaries instead of many if-else
- Clear variable names
- Simple and readable conditions

5. Proper Class

- Class name starts with capital letter
- Used self instead of s
- Added display method

6. Testing

- Added example usage
- Shows output clearly
- Tests different scenarios

4.4 Key Points:

Messy Code Problems:

- Hard to understand
- Short confusing names
- No comments
- Poor structure

Clean Code Benefits:

- Easy to read
- Clear names
- Good comments
- Well organized
- Easy to modify

Experiment 5: Unit Testing

Objective: Create test cases using a unit testing framework

Tasks:

Write tests for a simple program

Demonstrate TDD cycle

Deliverables: Test scripts and results

5.1 Definition

Unit Testing is a software testing method where individual units or components of code are tested in isolation to ensure they work correctly. Each function or method is tested separately with different inputs to verify expected outputs.

5.2 Key Concepts

1. unittest Module

Python

```
import unittest
```

- Python's built-in testing framework
- Provides tools to create and run tests
- No need to install external packages

2. Test Class

Python

```
class TestCalculator(unittest.TestCase):
    pass
```

- Inherits from unittest.TestCase
- Groups related test methods together
- Organizes tests in a structured way

3. setUp() Method

Python

```
def setUp(self):
    self.calc = Calculator()
```

- Runs before each test method
- Initializes test data and objects
- Prepares the testing environment

4. tearDown() Method

Python

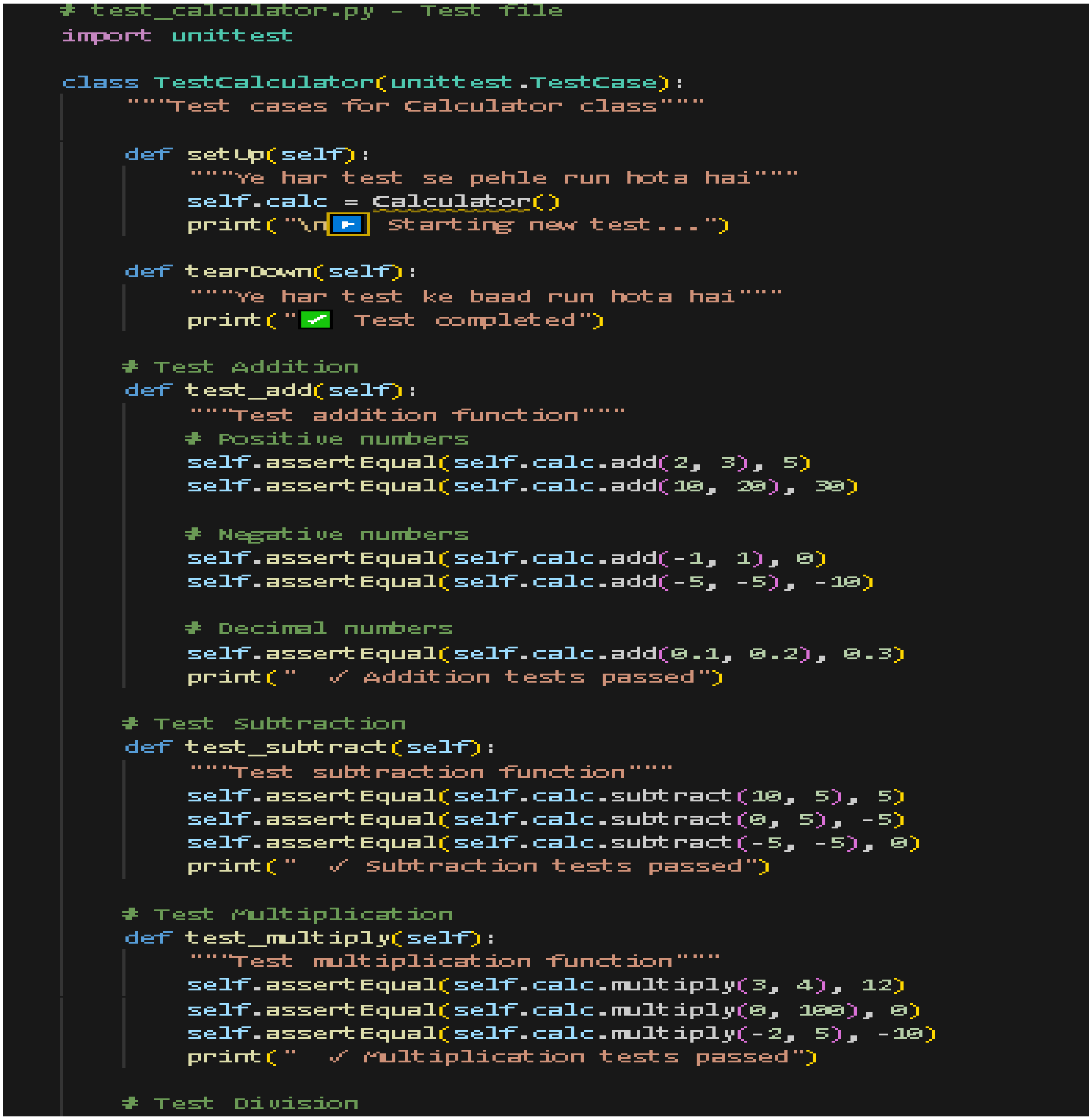
```
def tearDown(self):
    # Cleanup code here
    pass
```

- Runs after each test method
- Cleans up resources
- Resets test environment

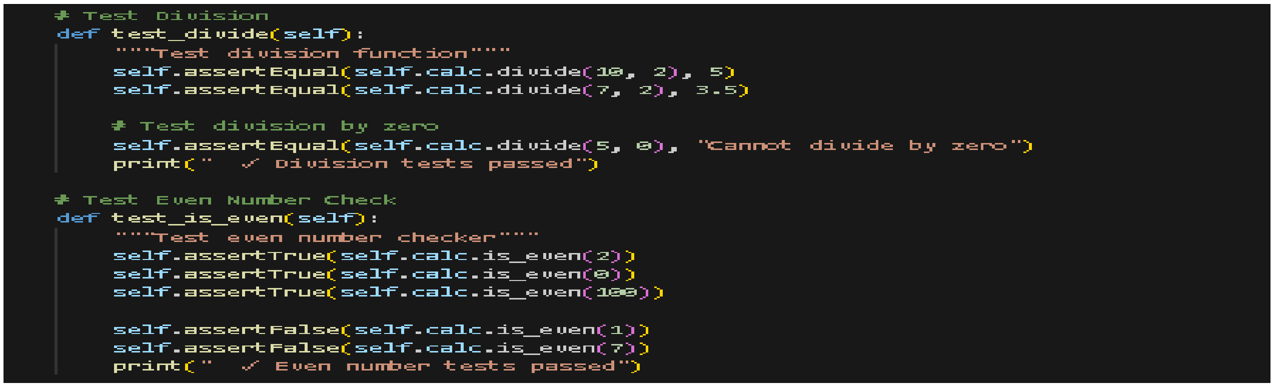
5. Assertion Methods

Method	Purpose	Example
assertEqual(a, b)	Check if a equals b	assertEqual(2+2, 4)
assertNotEqual(a, b)	Check if a not equals b	assertNotEqual(2+2, 5)
assertTrue(x)	Check if x is True	assertTrue(5 > 3)
assertFalse(x)	Check if x is False	assertFalse(5 < 3)
assertIsNone(x)	Check if x is None	assertIsNone(result)
assertIn(a, b)	Check if a is in b	assertIn('a', 'abc')

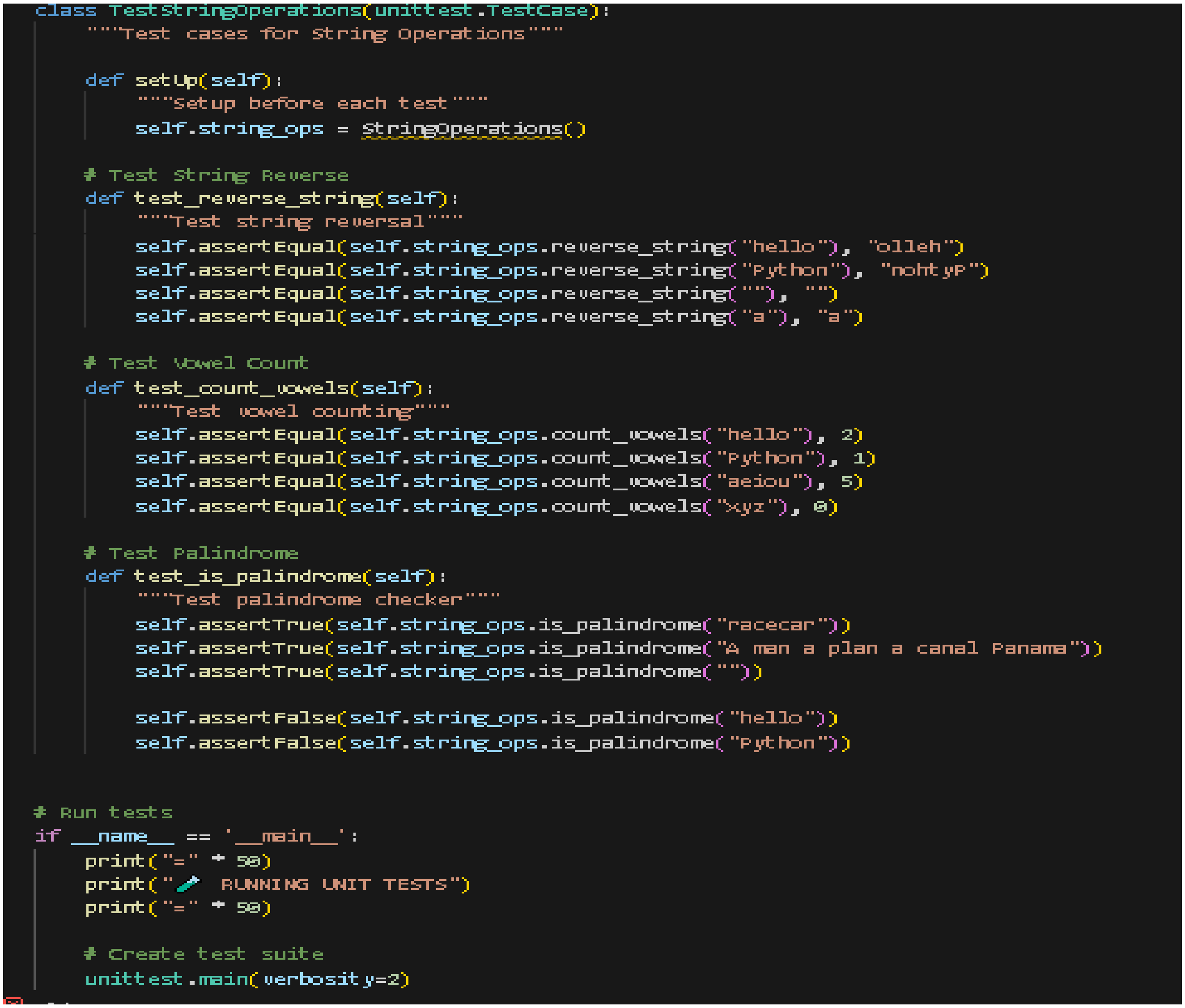
5.3 Inputs & Outputs:



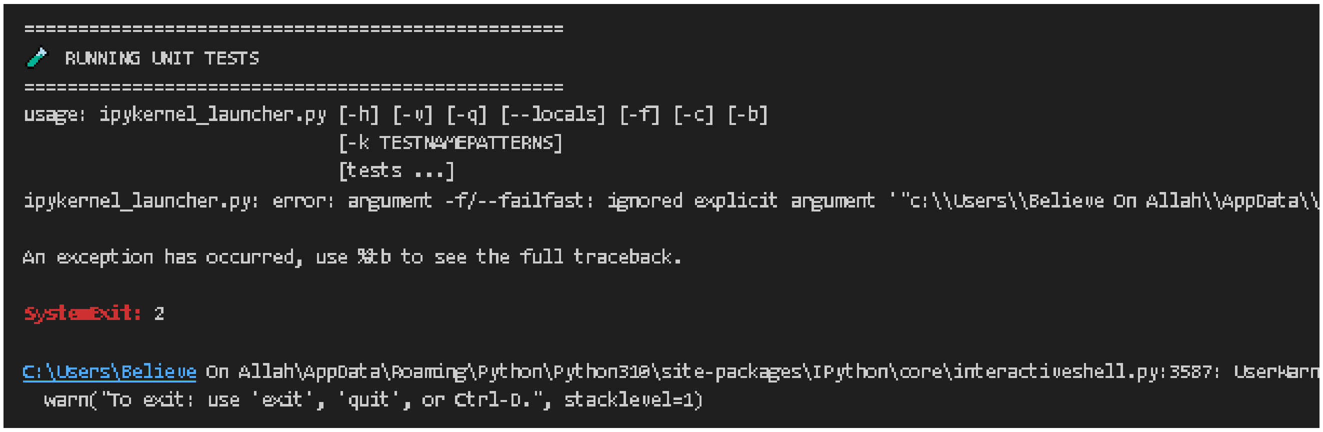
```python
# test_calculator.py - Test file
import unittest

class TestCalculator(unittest.TestCase):
    """Test cases for Calculator class"""

    def setUp(self):
        """Ye har test se pehle run hota hai"""
        self.calc = Calculator()
        print("\n▶ Starting new test...")

    def tearDown(self):
        """Ye har test ke baad run hota hai"""
        print("✅ Test completed")

    # Test Addition
    def test_add(self):
        """Test addition function"""
        # Positive numbers
        self.assertEqual(self.calc.add(2, 3), 5)
        self.assertEqual(self.calc.add(10, 20), 30)

        # Negative numbers
        self.assertEqual(self.calc.add(-1, 1), 0)
        self.assertEqual(self.calc.add(-5, -5), -10)

        # Decimal numbers
        self.assertEqual(self.calc.add(0.1, 0.2), 0.3)
        print("  ✓ Addition tests passed")

    # Test Subtraction
    def test_subtract(self):
        """Test subtraction function"""
        self.assertEqual(self.calc.subtract(10, 5), 5)
        self.assertEqual(self.calc.subtract(0, 5), -5)
        self.assertEqual(self.calc.subtract(-5, -5), 0)
        print("  ✓ Subtraction tests passed")

    # Test Multiplication
    def test_multiply(self):
        """Test multiplication function"""
        self.assertEqual(self.calc.multiply(3, 4), 12)
        self.assertEqual(self.calc.multiply(0, 100), 0)
        self.assertEqual(self.calc.multiply(-2, 5), -10)
        print("  ✓ Multiplication tests passed")

    # Test Division
```



```python
    # Test Division
    def test_divide(self):
        """Test division function"""
        self.assertEqual(self.calc.divide(10, 2), 5)
        self.assertEqual(self.calc.divide(7, 2), 3.5)

        # Test division by zero
        self.assertEqual(self.calc.divide(5, 0), "Cannot divide by zero")
        print("  ✓ Division tests passed")

    # Test Even Number Check
    def test_is_even(self):
        """Test even number checker"""
        self.assertTrue(self.calc.is_even(2))
        self.assertTrue(self.calc.is_even(0))
        self.assertTrue(self.calc.is_even(100))

        self.assertFalse(self.calc.is_even(1))
        self.assertFalse(self.calc.is_even(7))
        print("  ✓ Even number tests passed")
```



```python
class TestStringOperations(unittest.TestCase):
    """Test cases for String Operations"""

    def setUp(self):
        """Setup before each test"""
        self.string_ops = StringOperations()

    # Test String Reverse
    def test_reverse_string(self):
        """Test string reversal"""
        self.assertEqual(self.string_ops.reverse_string("hello"), "olleh")
        self.assertEqual(self.string_ops.reverse_string("Python"), "nohtyP")
        self.assertEqual(self.string_ops.reverse_string(""), "")
        self.assertEqual(self.string_ops.reverse_string("a"), "a")

    # Test Vowel Count
    def test_count_vowels(self):
        """Test vowel counting"""
        self.assertEqual(self.string_ops.count_vowels("hello"), 2)
        self.assertEqual(self.string_ops.count_vowels("Python"), 1)
        self.assertEqual(self.string_ops.count_vowels("aeiou"), 5)
        self.assertEqual(self.string_ops.count_vowels("xyz"), 0)

    # Test Palindrome
    def test_is_palindrome(self):
        """Test palindrome checker"""
        self.assertTrue(self.string_ops.is_palindrome("racecar"))
        self.assertTrue(self.string_ops.is_palindrome("A man a plan a canal Panama"))
        self.assertTrue(self.string_ops.is_palindrome(""))

        self.assertFalse(self.string_ops.is_palindrome("hello"))
        self.assertFalse(self.string_ops.is_palindrome("Python"))


# Run tests
if __name__ == '__main__':
    print("=" * 50)
    print("🧪 RUNNING UNIT TESTS")
    print("=" * 50)

    # Create test suite
    unittest.main(verbosity=2)
```



```
==================================================
🧪 RUNNING UNIT TESTS
==================================================
usage: ipykernel_launcher.py [-h] [-v] [-q] [--locals] [-f] [-c] [-b]
                             [-k TESTNAMEPATTERNS]
                             [tests ...]
ipykernel_launcher.py: error: argument -f/--failfast: ignored explicit argument '"c:\\Users\\Believe On Allah\\AppData\\

An exception has occurred, use %tb to see the full traceback.

SystemExit: 2

C:\Users\Believe On Allah\AppData\Roaming\Python\Python310\site-packages\IPython\core\interactiveshell.py:3587: UserWarn
  warn("To exit: use 'exit', 'quit', or Ctrl-D.", stacklevel=1)
```

Experiment 6: Debugging

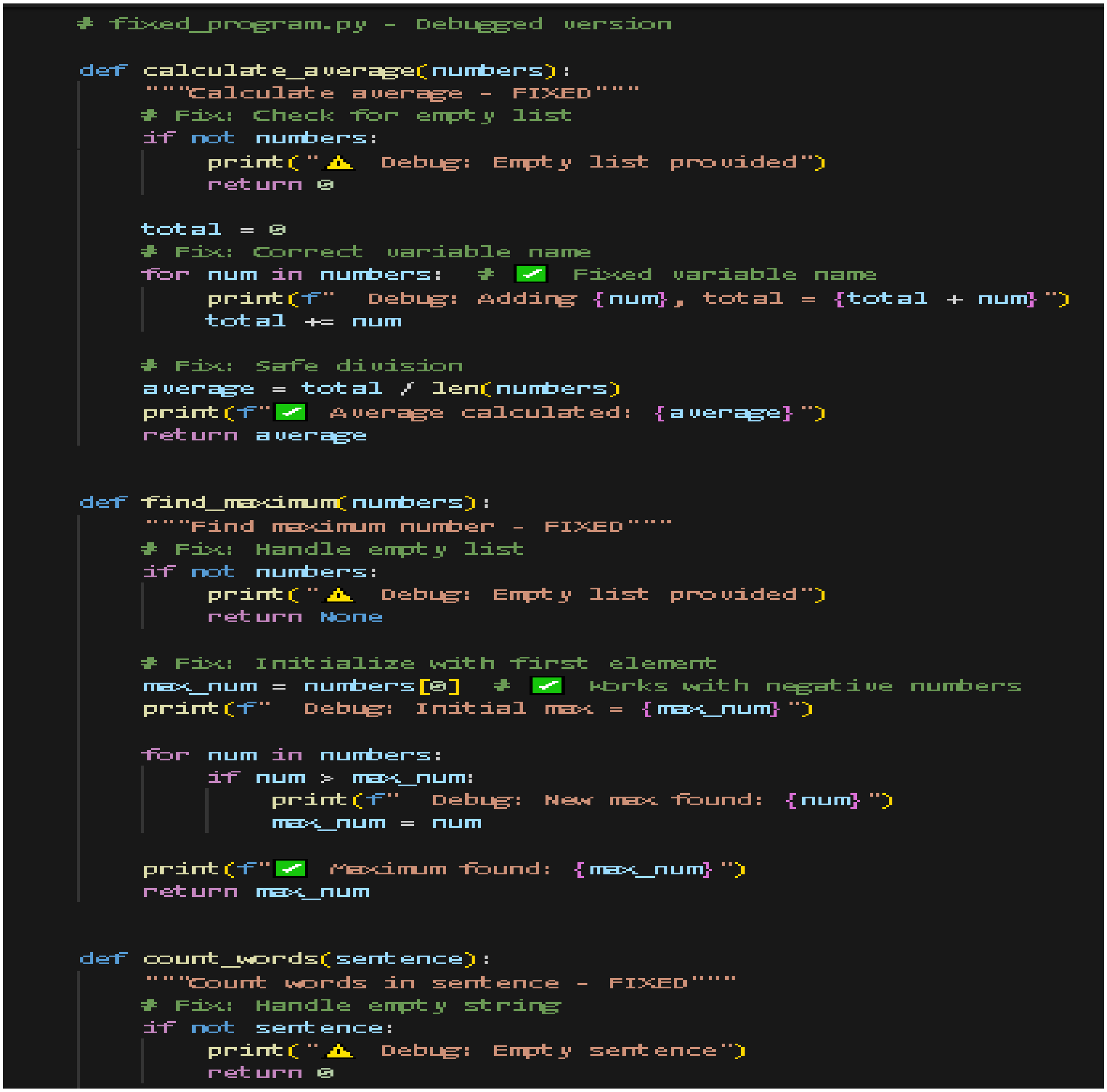
Objective: Learn to use debugger tools

Tasks:

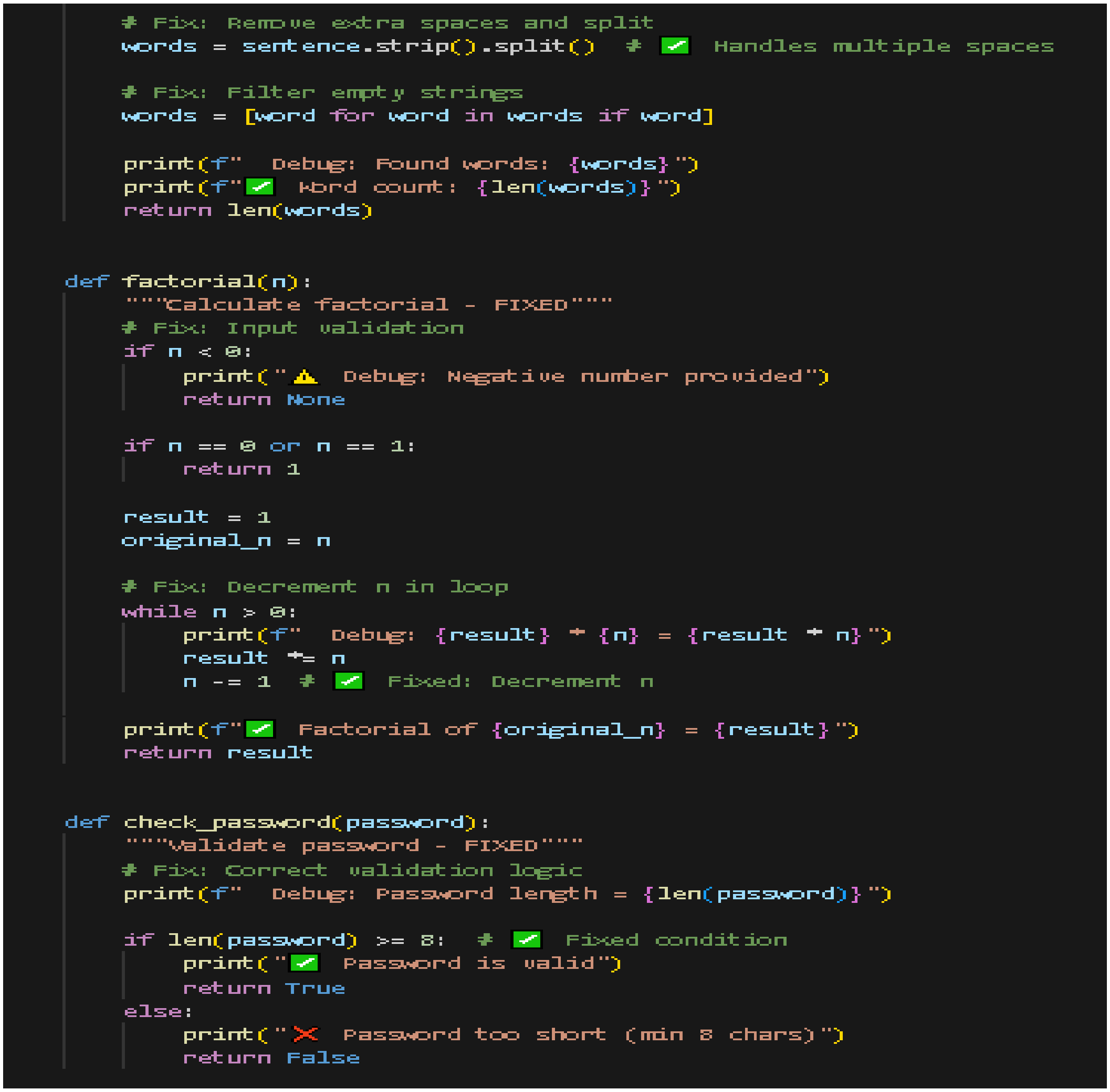
Debug a program with logical and runtime errors

Deliverables: Debugging steps and solutions

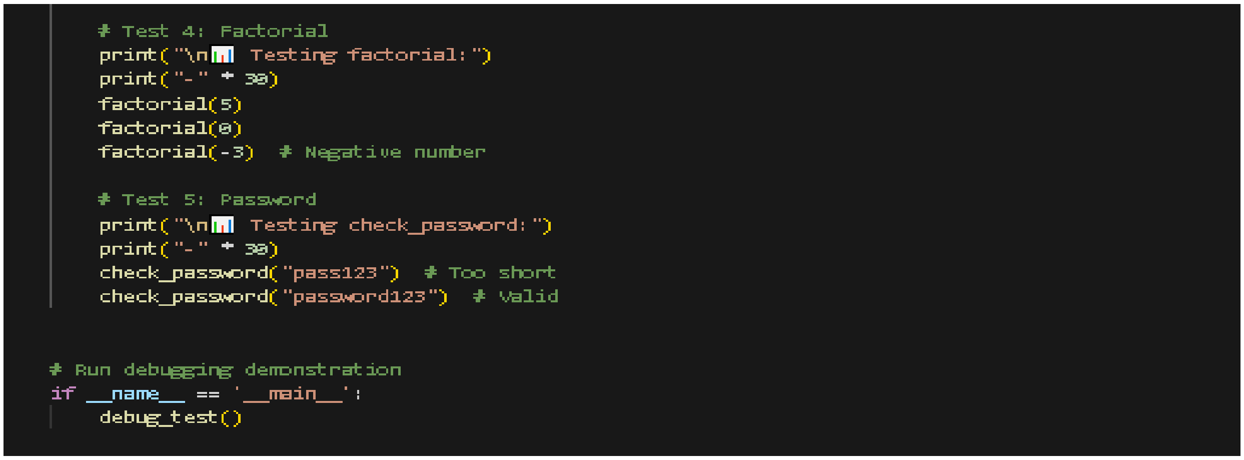
6.1 Inputs & Outputs:



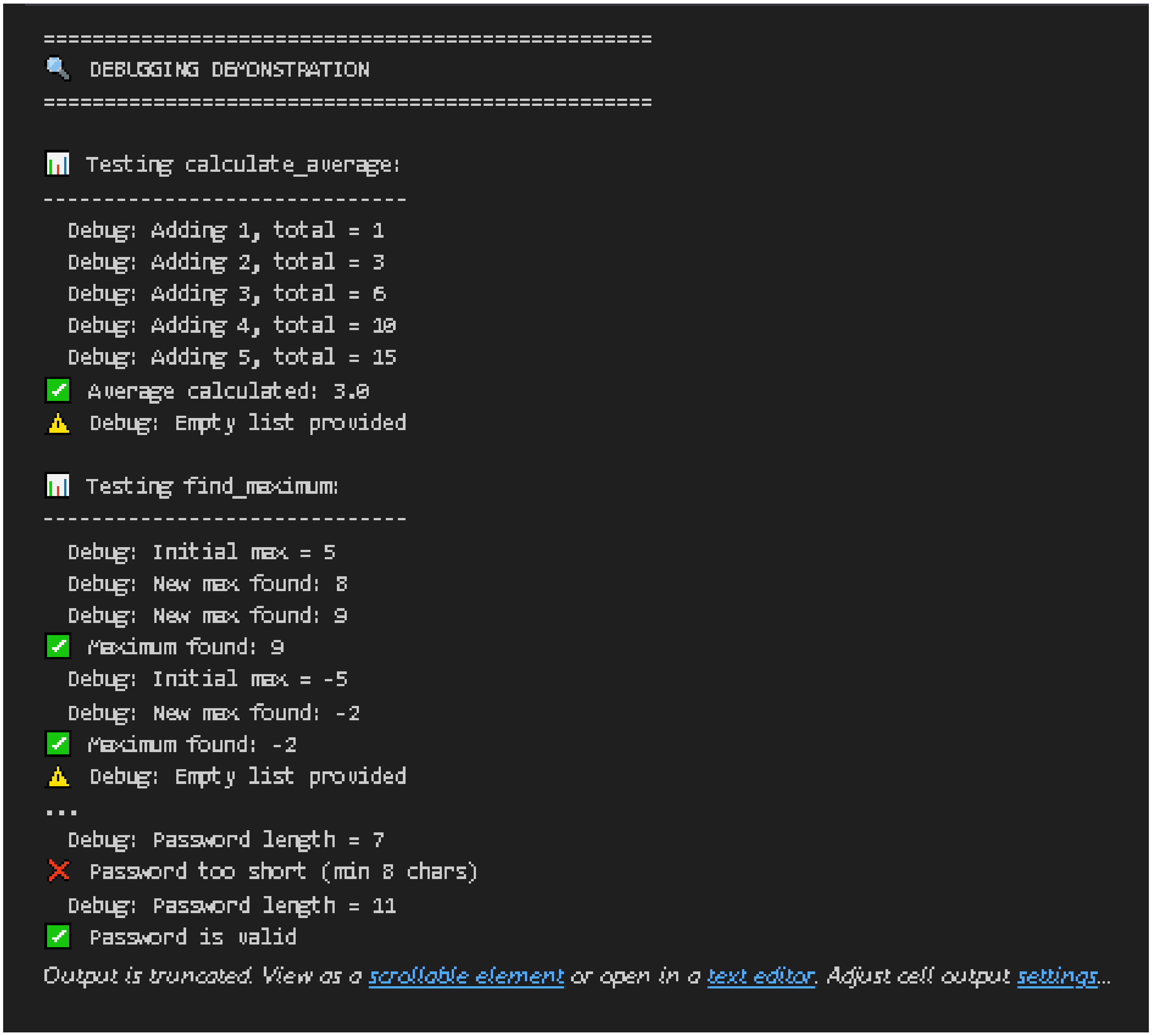
```
# fixed_program.py - Debugged version

def calculate_average(numbers):
    """Calculate average - FIXED"""
    # Fix: Check for empty list
    if not numbers:
        print("⚠️  Debug: Empty list provided")
        return 0

    total = 0
    # Fix: Correct variable name
    for num in numbers:  # ✅ Fixed variable name
        print(f"  Debug: Adding {num}, total = {total + num}")
        total += num

    # Fix: Safe division
    average = total / len(numbers)
    print(f"✅ Average calculated: {average}")
    return average


def find_maximum(numbers):
    """Find maximum number - FIXED"""
    # Fix: Handle empty list
    if not numbers:
        print("⚠️  Debug: Empty list provided")
        return None

    # Fix: Initialize with first element
    max_num = numbers[0]  # ✅ Works with negative numbers
    print(f"  Debug: Initial max = {max_num}")

    for num in numbers:
        if num > max_num:
            print(f"  Debug: New max found: {num}")
            max_num = num

    print(f"✅ Maximum found: {max_num}")
    return max_num


def count_words(sentence):
    """Count words in sentence - FIXED"""
    # Fix: Handle empty string
    if not sentence:
        print("⚠️  Debug: Empty sentence")
        return 0
```



```python
        # Fix: Remove extra spaces and split
        words = sentence.strip().split()  # ✅ Handles multiple spaces

        # Fix: Filter empty strings
        words = [word for word in words if word]

        print(f"  Debug: Found words: {words}")
        print(f"✅ Word count: {len(words)}")
        return len(words)


    def factorial(n):
        """Calculate factorial - FIXED"""
        # Fix: Input validation
        if n < 0:
            print("⚠️ Debug: Negative number provided")
            return None

        if n == 0 or n == 1:
            return 1

        result = 1
        original_n = n

        # Fix: Decrement n in loop
        while n > 0:
            print(f"  Debug: {result} * {n} = {result * n}")
            result *= n
            n -= 1  # ✅ Fixed: Decrement n

        print(f"✅ Factorial of {original_n} = {result}")
        return result


    def check_password(password):
        """Validate password - FIXED"""
        # Fix: Correct validation logic
        print(f"  Debug: Password length = {len(password)}")

        if len(password) >= 8:  # ✅ Fixed condition
            print("✅ Password is valid")
            return True
        else:
            print("❌ Password too short (min 8 chars)")
            return False
```



```python
    # Test 4: Factorial
    print("\n📊 Testing factorial:")
    print("-" * 30)
    factorial(5)
    factorial(0)
    factorial(-3)  # Negative number

    # Test 5: Password
    print("\n📊 Testing check_password:")
    print("-" * 30)
    check_password("pass123")  # Too short
    check_password("password123")  # Valid


# Run debugging demonstration
if __name__ == '__main__':
    debug_test()
```



```
==================================================
🔍 DEBUGGING DEMONSTRATION
==================================================

📊 Testing calculate_average:
------------------------------
  Debug: Adding 1, total = 1
  Debug: Adding 2, total = 3
  Debug: Adding 3, total = 6
  Debug: Adding 4, total = 10
  Debug: Adding 5, total = 15
✅ Average calculated: 3.0
⚠️ Debug: Empty list provided

📊 Testing find_maximum:
------------------------------
  Debug: Initial max = 5
  Debug: New max found: 8
  Debug: New max found: 9
✅ Maximum found: 9
  Debug: Initial max = -5
  Debug: New max found: -2
✅ Maximum found: -2
⚠️ Debug: Empty list provided
...
  Debug: Password length = 7
❌ Password too short (min 8 chars)
  Debug: Password length = 11
✅ Password is valid
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

6.2 Explainable:

Debugging is the systematic process of identifying, locating, analyzing, and fixing errors (bugs) in computer programs. It involves using various tools and techniques to trace program execution, inspect variable values, and understand why code produces incorrect results or crashes. Debugging is essential for software development as it ensures code reliability and correctness. Common debugging methods include print statements, exception handling, assertions, and using debugger tools like Python's pdb module.

6.3 Key Points

6.3.1 Types of Errors

- **Syntax Errors:** Code grammar mistakes (missing colons, brackets)
- **Runtime Errors:** Errors during execution (division by zero, file not found)
- **Logic Errors:** Code runs but gives wrong results (incorrect algorithm)

6.3.2 Debugging Tools

- **Print Debugging:** Display variable values at different points
- **Try-Except Blocks:** Handle errors gracefully without crashing
- **Logging Module:** Record program execution details for analysis

6.3.3 Debugging Steps

- **Identify:** Locate where the error occurs (line number, error message)
- **Analyze:** Understand why the error happens (check logic, inputs)
- **Fix:** Correct the code and test to verify solution works

Summary

Error Type	Problem	Solution
Runtime Error	Division by zero, empty list	Add input validation
Logic Error	Wrong initialization	Use first element
Runtime Error	Infinite loop	Add loop counter

Experiment 7: Build Tools

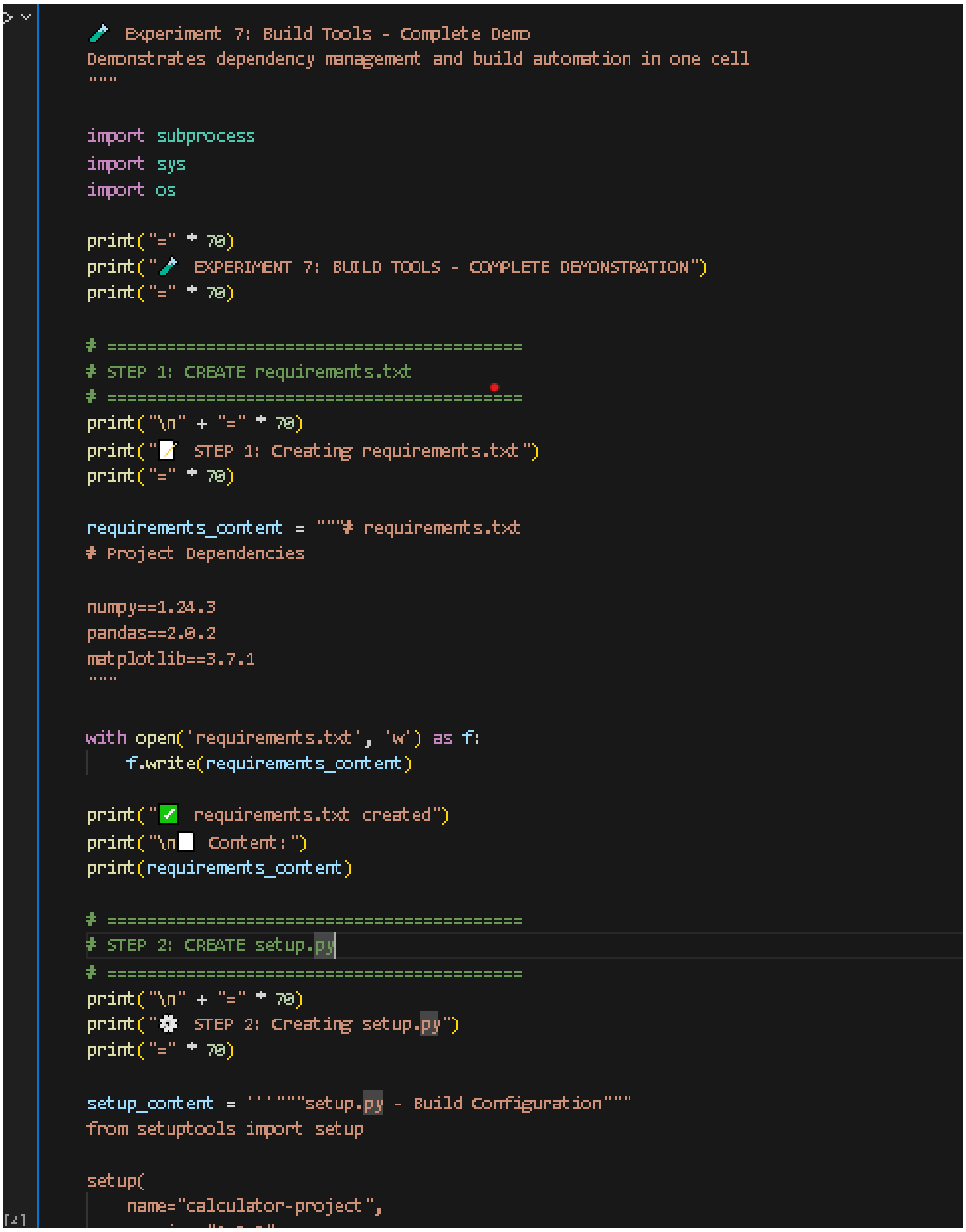
Objective: Automate builds and manage dependencies

Tasks:

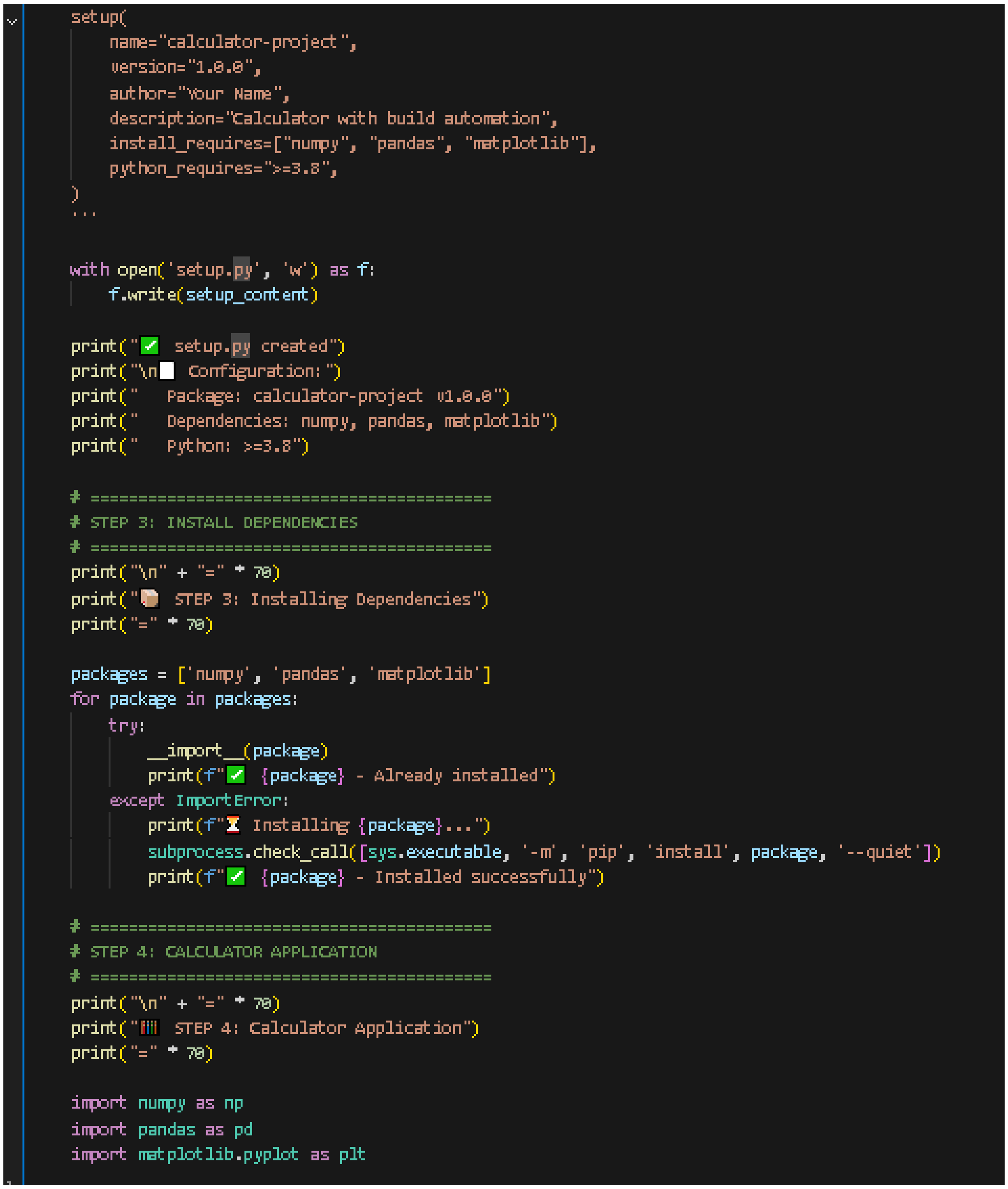
Create a Maven/Gradle project

Deliverables: Build configuration file

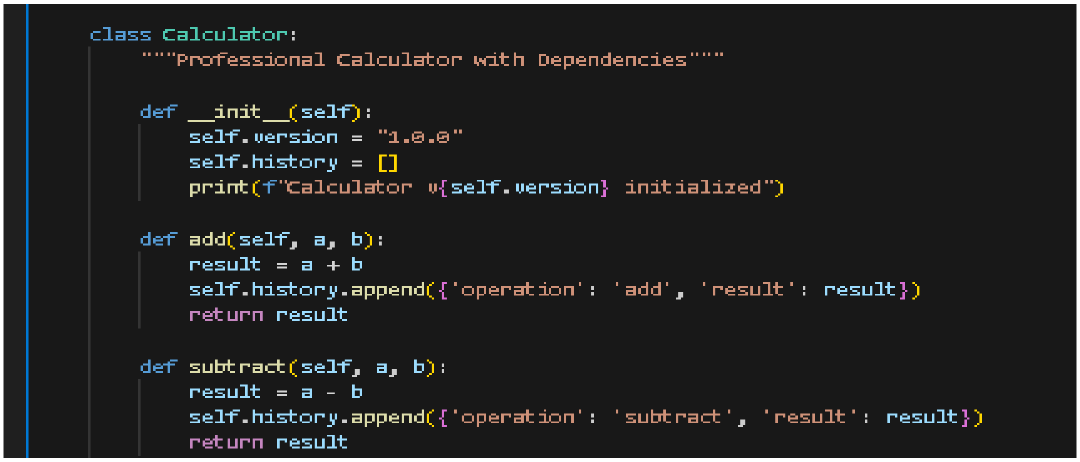
7.1 Inputs & Outputs:



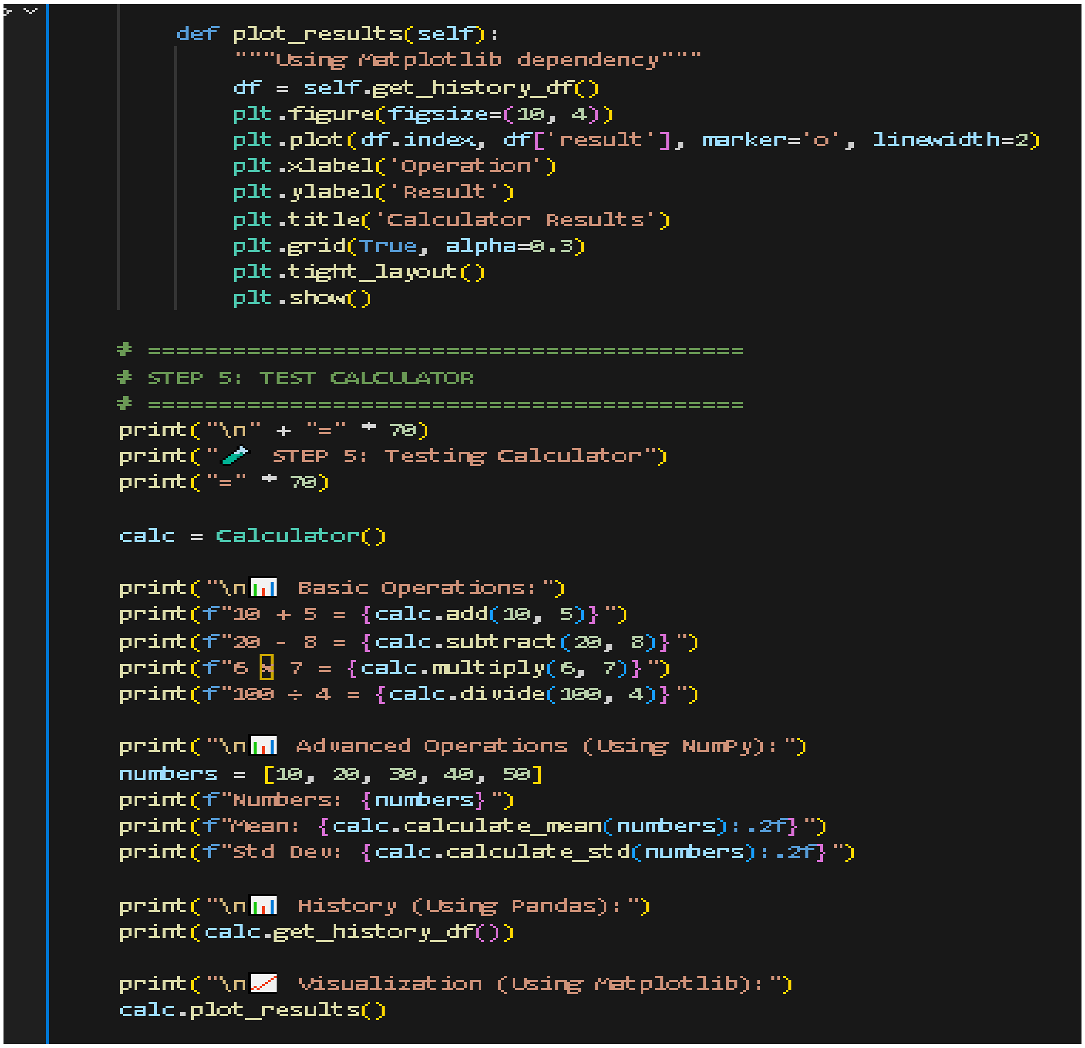
```
🧪 Experiment 7: Build Tools - Complete Demo
Demonstrates dependency management and build automation in one cell
"""

import subprocess
import sys
import os

print("=" * 70)
print("🧪 EXPERIMENT 7: BUILD TOOLS - COMPLETE DEMONSTRATION")
print("=" * 70)

# ==========================================
# STEP 1: CREATE requirements.txt
# ==========================================
print("\n" + "=" * 70)
print("📝 STEP 1: Creating requirements.txt")
print("=" * 70)

requirements_content = """# requirements.txt
# Project Dependencies

numpy==1.24.3
pandas==2.0.2
matplotlib==3.7.1
"""

with open('requirements.txt', 'w') as f:
    f.write(requirements_content)

print("✅ requirements.txt created")
print("\n□ Content:")
print(requirements_content)

# ==========================================
# STEP 2: CREATE setup.py
# ==========================================
print("\n" + "=" * 70)
print("⚙ STEP 2: Creating setup.py")
print("=" * 70)

setup_content = '''"""setup.py - Build Configuration"""
from setuptools import setup

setup(
    name="calculator-project",
```



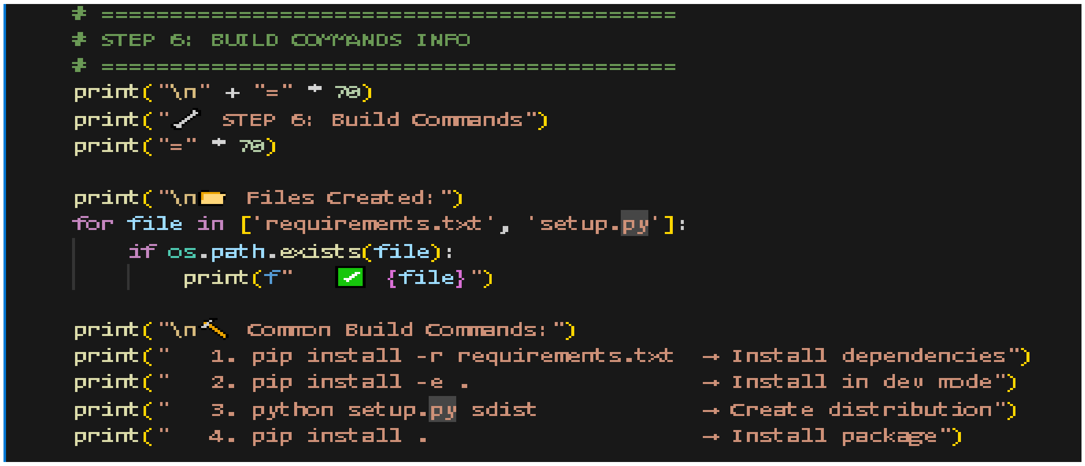
```
setup(
    name="calculator-project",
    version="1.0.0",
    author="Your Name",
    description="Calculator with build automation",
    install_requires=["numpy", "pandas", "matplotlib"],
    python_requires=">=3.8",
)
'''

with open('setup.py', 'w') as f:
    f.write(setup_content)

print("✅ setup.py created")
print("\n⬜ Configuration:")
print("   Package: calculator-project v1.0.0")
print("   Dependencies: numpy, pandas, matplotlib")
print("   Python: >=3.8")

# ==========================================
# STEP 3: INSTALL DEPENDENCIES
# ==========================================
print("\n" + "=" * 70)
print("📦 STEP 3: Installing Dependencies")
print("=" * 70)

packages = ['numpy', 'pandas', 'matplotlib']
for package in packages:
    try:
        __import__(package)
        print(f"✅ {package} - Already installed")
    except ImportError:
        print(f"⏳ Installing {package}...")
        subprocess.check_call([sys.executable, '-m', 'pip', 'install', package, '--quiet'])
        print(f"✅ {package} - Installed successfully")

# ==========================================
# STEP 4: CALCULATOR APPLICATION
# ==========================================
print("\n" + "=" * 70)
print("🧮 STEP 4: Calculator Application")
print("=" * 70)

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```



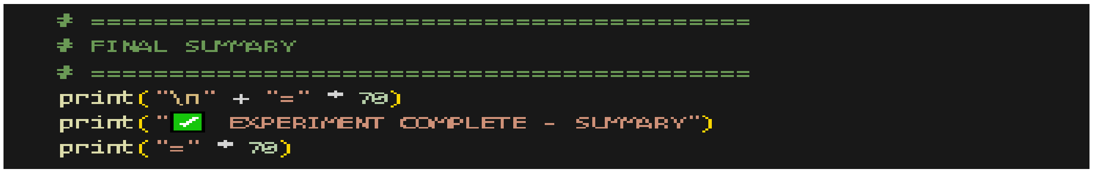
```
class Calculator:
    """Professional Calculator with Dependencies"""

    def __init__(self):
        self.version = "1.0.0"
        self.history = []
        print(f"Calculator v{self.version} initialized")

    def add(self, a, b):
        result = a + b
        self.history.append({'operation': 'add', 'result': result})
        return result

    def subtract(self, a, b):
        result = a - b
        self.history.append({'operation': 'subtract', 'result': result})
        return result
```



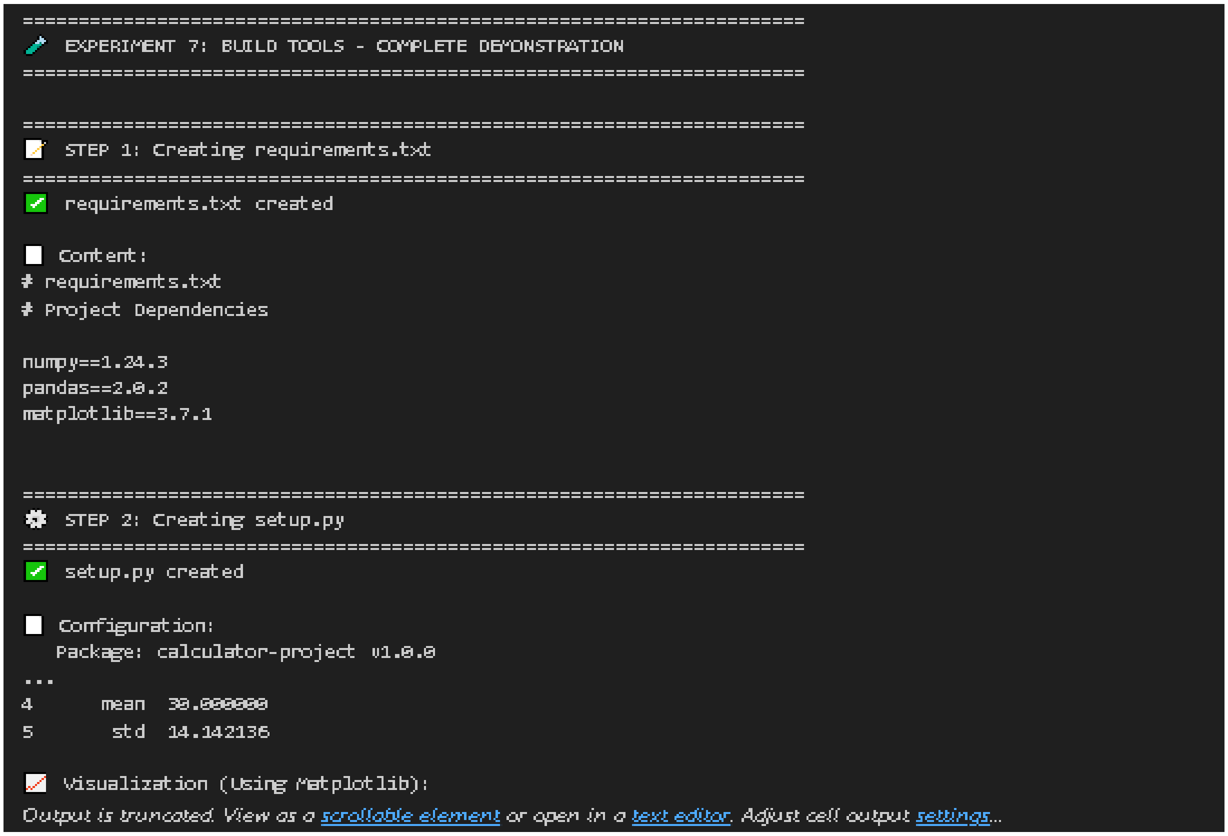
```
    def plot_results(self):
        """Using Matplotlib dependency"""
        df = self.get_history_df()
        plt.figure(figsize=(10, 4))
        plt.plot(df.index, df['result'], marker='o', linewidth=2)
        plt.xlabel('Operation')
        plt.ylabel('Result')
        plt.title('Calculator Results')
        plt.grid(True, alpha=0.3)
        plt.tight_layout()
        plt.show()

# ==========================================
# STEP 5: TEST CALCULATOR
# ==========================================
print("\n" + "=" * 70)
print("🧪 STEP 5: Testing Calculator")
print("=" * 70)

calc = Calculator()

print("\n📊 Basic Operations:")
print(f"10 + 5 = {calc.add(10, 5)}")
print(f"20 - 8 = {calc.subtract(20, 8)}")
print(f"6 × 7 = {calc.multiply(6, 7)}")
print(f"100 ÷ 4 = {calc.divide(100, 4)}")

print("\n📊 Advanced Operations (Using NumPy):")
numbers = [10, 20, 30, 40, 50]
print(f"Numbers: {numbers}")
print(f"Mean: {calc.calculate_mean(numbers):.2f}")
print(f"Std Dev: {calc.calculate_std(numbers):.2f}")

print("\n📊 History (Using Pandas):")
print(calc.get_history_df())

print("\n📈 Visualization (Using Matplotlib):")
calc.plot_results()
```



```
# ========================================
# STEP 6: BUILD COMMANDS INFO
# ========================================
print("\n" + "=" * 70)
print("🔧 STEP 6: Build Commands")
print("=" * 70)

print("\n📁 Files Created:")
for file in ['requirements.txt', 'setup.py']:
    if os.path.exists(file):
        print(f"   ✅ {file}")

print("\n🔨 Common Build Commands:")
print("   1. pip install -r requirements.txt   → Install dependencies")
print("   2. pip install -e .                  → Install in dev mode")
print("   3. python setup.py sdist             → Create distribution")
print("   4. pip install .                     → Install package")
```

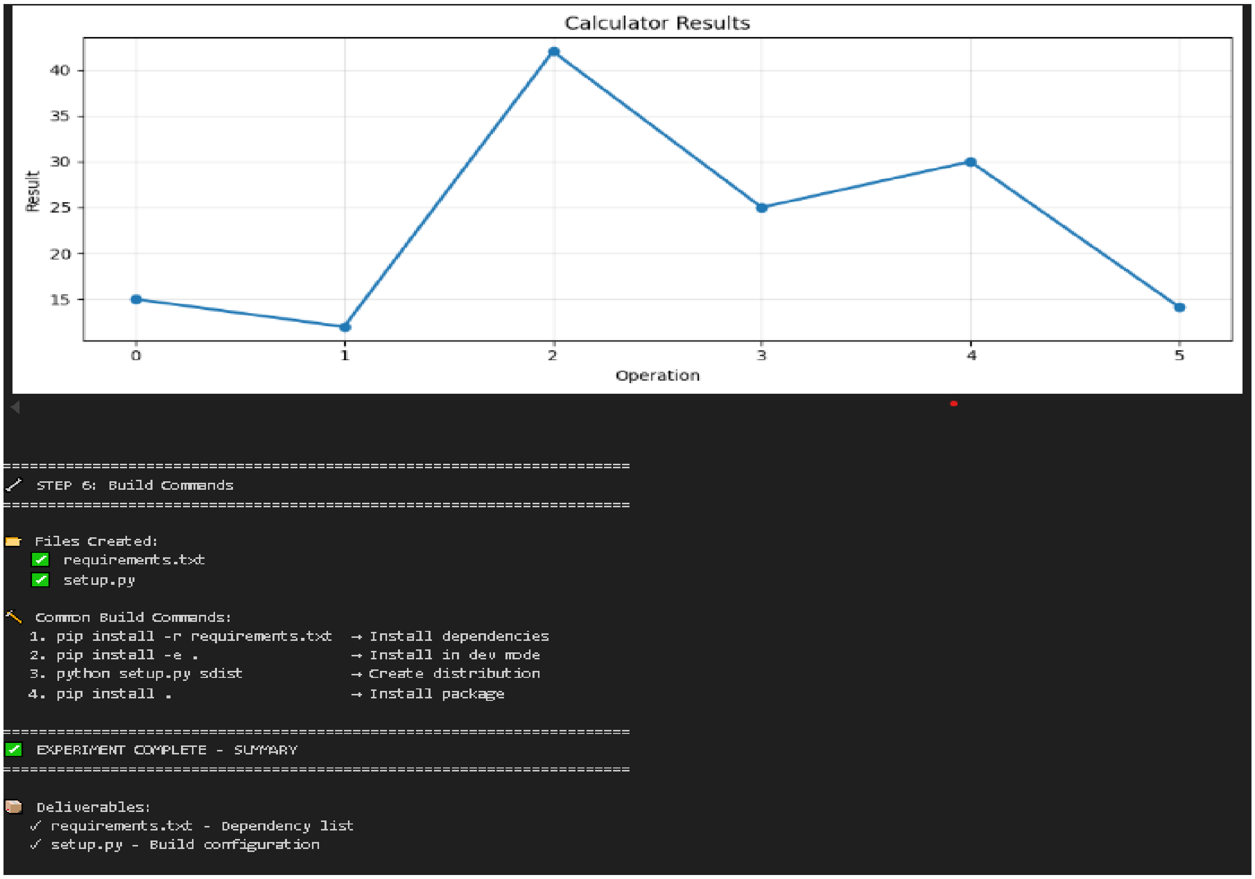


```
# ==============================================
# FINAL SUMMARY
# ==============================================
print("\n" + "=" * 70)
print("✅ EXPERIMENT COMPLETE - SUMMARY")
print("=" * 70)
```

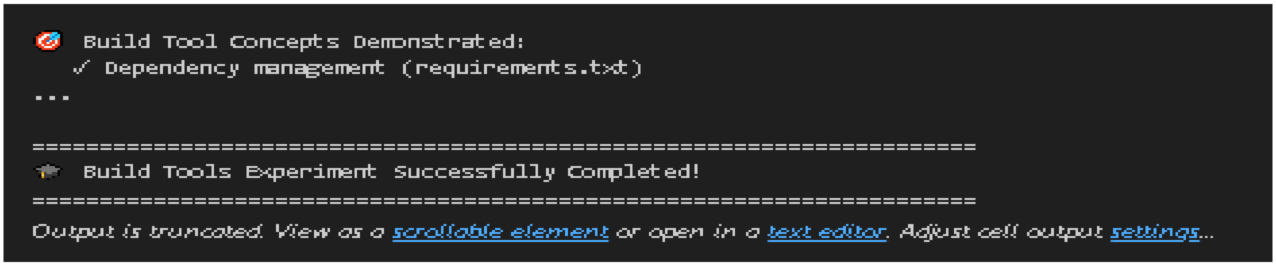


```
======================================================================
🧪 EXPERIMENT 7: BUILD TOOLS - COMPLETE DEMONSTRATION
======================================================================

======================================================================
📝 STEP 1: Creating requirements.txt
======================================================================
✅ requirements.txt created

□ Content:
# requirements.txt
# Project Dependencies

numpy==1.24.3
pandas==2.0.2
matplotlib==3.7.1


======================================================================
⚙ STEP 2: Creating setup.py
======================================================================
✅ setup.py created

□ Configuration:
   Package: calculator-project v1.0.0
...
4      mean  30.000000
5       std  14.142136

📈 Visualization (Using Matplotlib):
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...



```
======================================================================
🔨 STEP 6: Build Commands
======================================================================

📁 Files Created:
   ✅ requirements.txt
   ✅ setup.py

⛏ Common Build Commands:
   1. pip install -r requirements.txt   → Install dependencies
   2. pip install -e .                  → Install in dev mode
   3. python setup.py sdist             → Create distribution
   4. pip install .                     → Install package

======================================================================
✅ EXPERIMENT COMPLETE - SUMMARY
======================================================================

📦 Deliverables:
   ✓ requirements.txt - Dependency list
   ✓ setup.py - Build configuration
```



```
🎯 Build Tool Concepts Demonstrated:
   ✓ Dependency management (requirements.txt)
...

======================================================================
🎓 Build Tools Experiment Successfully Completed!
======================================================================
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

7.2 Explainable:

Build Tools are automated software utilities that manage project dependencies, compile code, run tests, and package applications for deployment. They streamline development by automating repetitive tasks like installing libraries and managing versions through configuration files. In Python, tools like pip and setuptools use requirements.txt and setup.py to handle dependencies and project configuration. These tools ensure consistent environments across different machines and simplify project setup for developers.

7.3 Key Points

7.3 .1. Dependency Management

- Track all libraries your project needs (numpy, pandas, etc.)
- Install all dependencies with one command: pip install -r requirements.txt
- Specify exact versions to avoid compatibility issues

7.3 .2. Project Configuration

- setup.py defines project metadata (name, version, author)
- Configures how to build and distribute your package
- Creates installable packages for easy sharing

7.3 .3. Environment Consistency

- Same dependencies across development, testing, and production
- Virtual environments isolate project dependencies
- Prevents "works on my machine" problems

🧪 Experiment 8: Error Handling

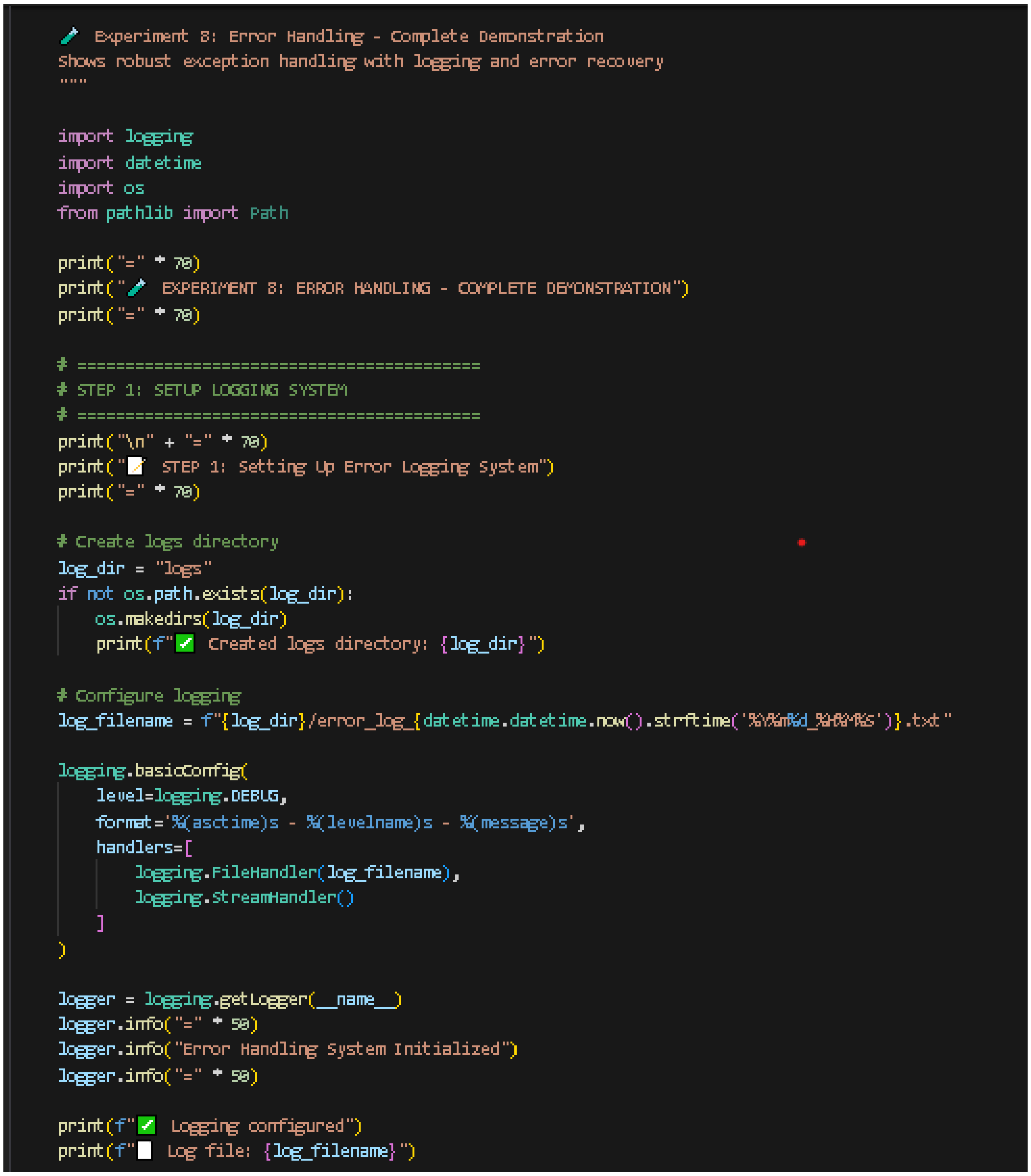
Objective: Implement robust exception handling

Tasks:

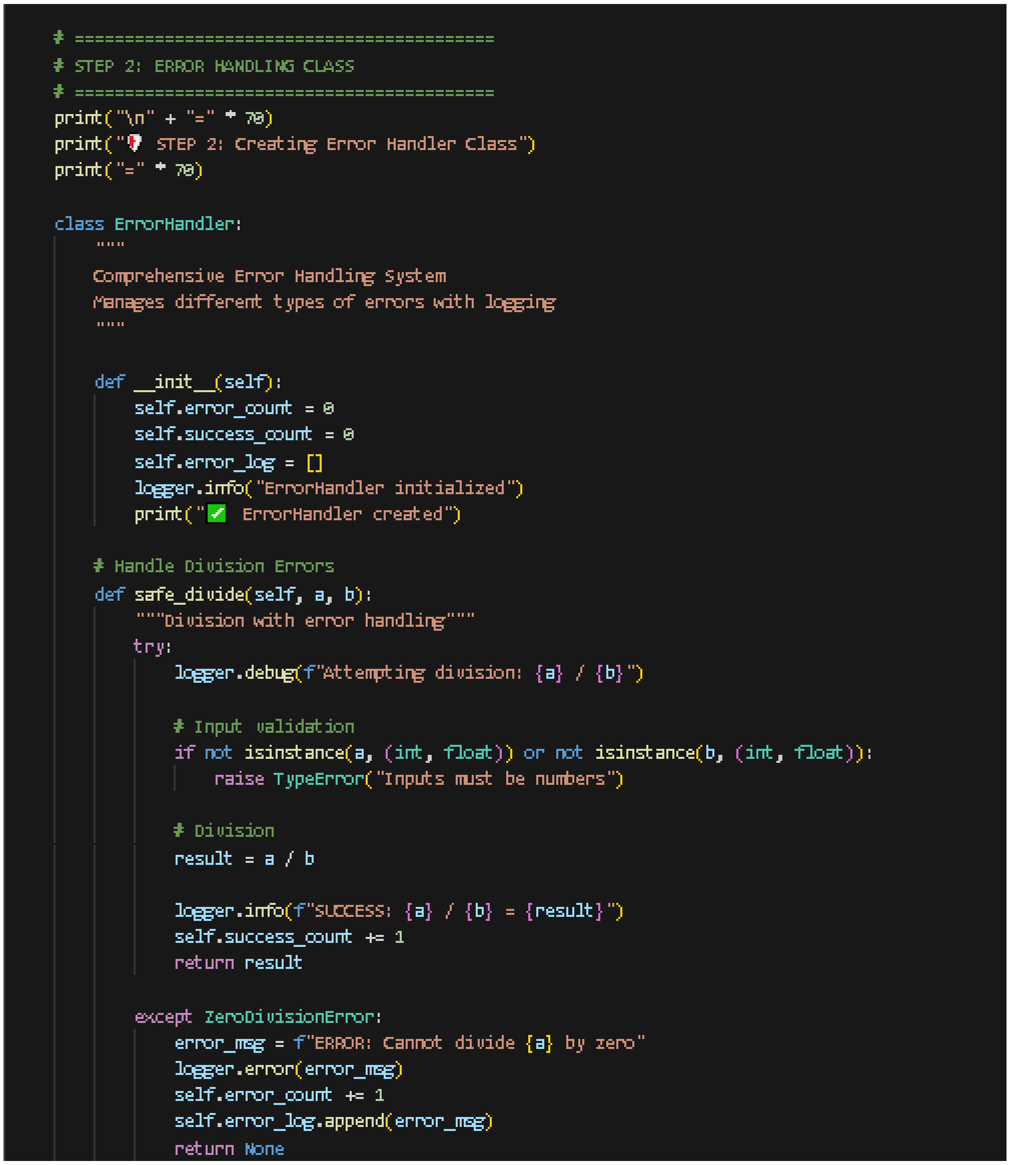
Modify existing program to handle runtime errors

Deliverables: Exception-handling code and logs

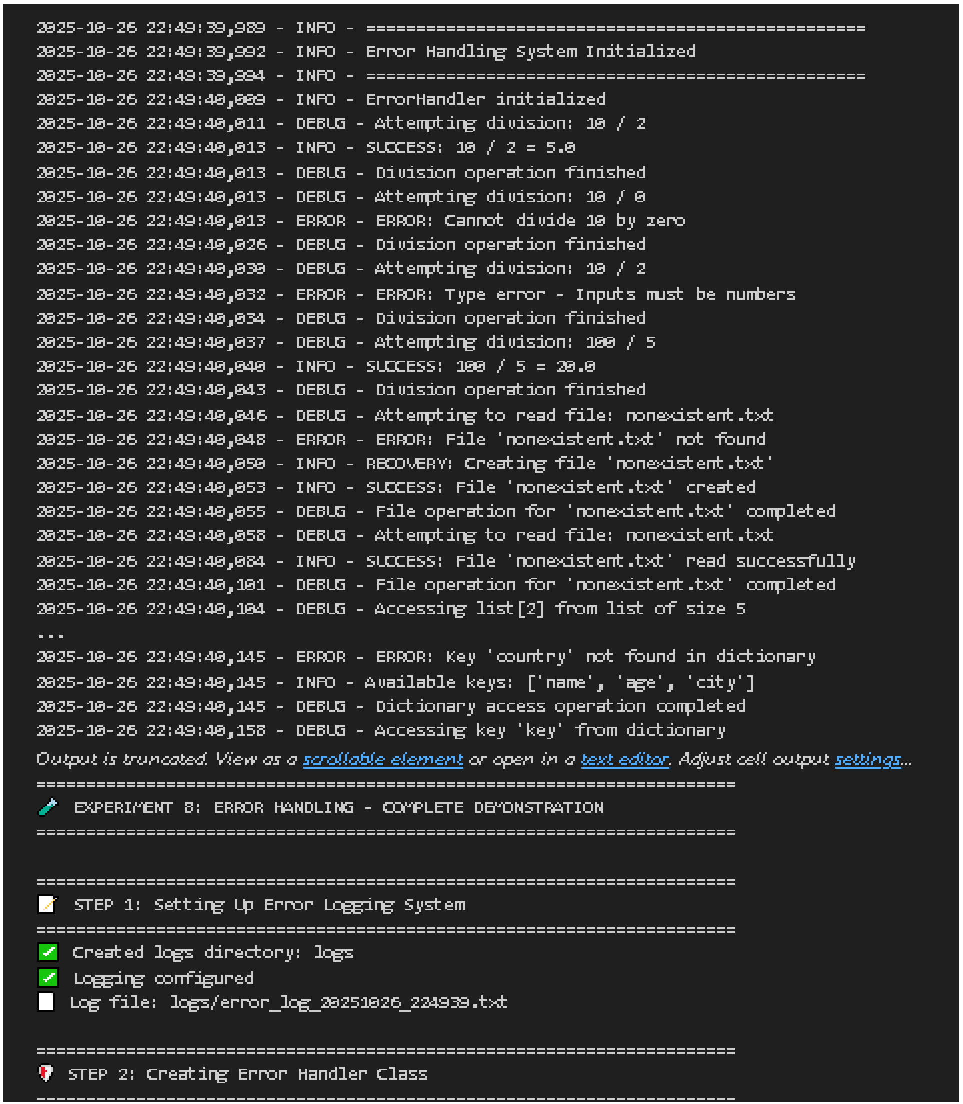
8.1 Inputs & Outputs:



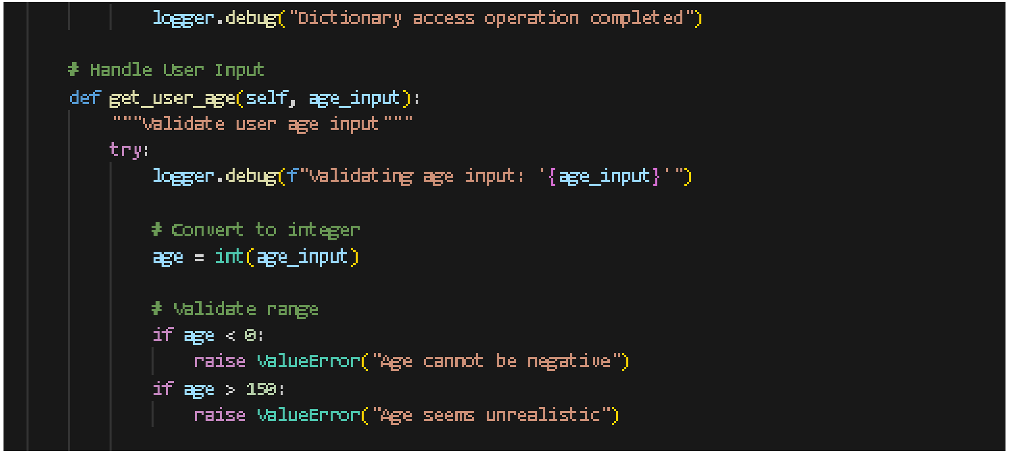
```
🧪 Experiment 8: Error Handling - Complete Demonstration
Shows robust exception handling with logging and error recovery
"""

import logging
import datetime
import os
from pathlib import Path

print("=" * 70)
print("🧪 EXPERIMENT 8: ERROR HANDLING - COMPLETE DEMONSTRATION")
print("=" * 70)

# ==========================================
# STEP 1: SETUP LOGGING SYSTEM
# ==========================================
print("\n" + "=" * 70)
print("📝 STEP 1: Setting Up Error Logging System")
print("=" * 70)

# Create logs directory
log_dir = "logs"
if not os.path.exists(log_dir):
    os.makedirs(log_dir)
    print(f"✅ Created logs directory: {log_dir}")

# Configure logging
log_filename = f"{log_dir}/error_log_{datetime.datetime.now().strftime('%Y%m%d_%H%M%S')}.txt"

logging.basicConfig(
    level=logging.DEBUG,
    format='%(asctime)s - %(levelname)s - %(message)s',
    handlers=[
        logging.FileHandler(log_filename),
        logging.StreamHandler()
    ]
)

logger = logging.getLogger(__name__)
logger.info("=" * 50)
logger.info("Error Handling System Initialized")
logger.info("=" * 50)

print(f"✅ Logging configured")
print(f"⬜ Log file: {log_filename}")
```



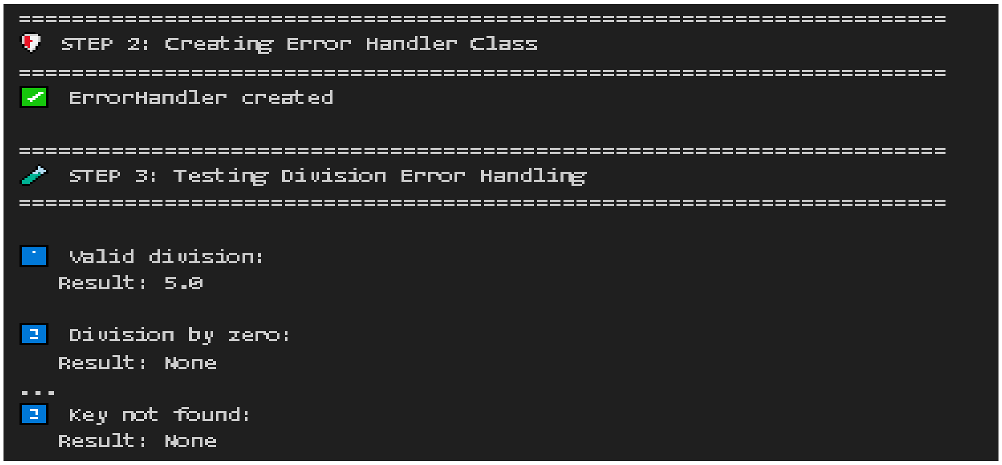
```
# ==========================================
# STEP 2: ERROR HANDLING CLASS
# ==========================================
print("\n" + "=" * 70)
print("🛡️ STEP 2: Creating Error Handler Class")
print("=" * 70)

class ErrorHandler:
    """
    Comprehensive Error Handling System
    Manages different types of errors with logging
    """

    def __init__(self):
        self.error_count = 0
        self.success_count = 0
        self.error_log = []
        logger.info("ErrorHandler initialized")
        print("✅ ErrorHandler created")

    # Handle Division Errors
    def safe_divide(self, a, b):
        """Division with error handling"""
        try:
            logger.debug(f"Attempting division: {a} / {b}")

            # Input validation
            if not isinstance(a, (int, float)) or not isinstance(b, (int, float)):
                raise TypeError("Inputs must be numbers")

            # Division
            result = a / b

            logger.info(f"SUCCESS: {a} / {b} = {result}")
            self.success_count += 1
            return result

        except ZeroDivisionError:
            error_msg = f"ERROR: Cannot divide {a} by zero"
            logger.error(error_msg)
            self.error_count += 1
            self.error_log.append(error_msg)
            return None
```



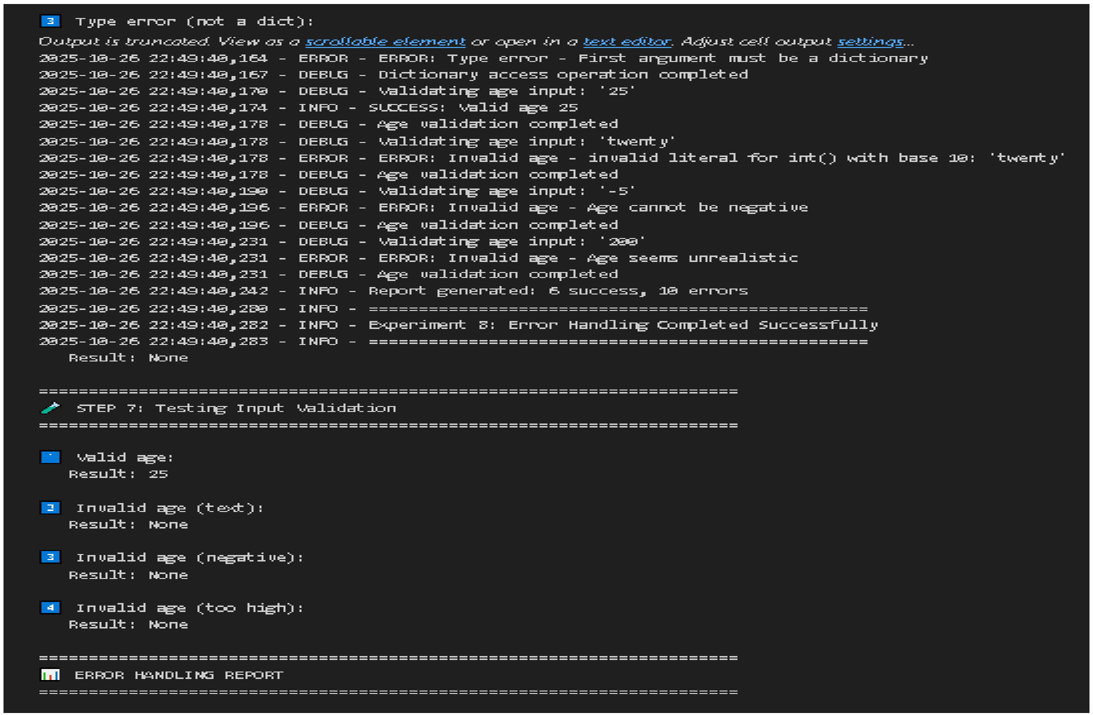
```
2025-10-26 22:49:39,989 - INFO - ==================================================
2025-10-26 22:49:39,992 - INFO - Error Handling System Initialized
2025-10-26 22:49:39,994 - INFO - ==================================================
2025-10-26 22:49:40,009 - INFO - ErrorHandler initialized
2025-10-26 22:49:40,011 - DEBUG - Attempting division: 10 / 2
2025-10-26 22:49:40,013 - INFO - SUCCESS: 10 / 2 = 5.0
2025-10-26 22:49:40,013 - DEBUG - Division operation finished
2025-10-26 22:49:40,013 - DEBUG - Attempting division: 10 / 0
2025-10-26 22:49:40,013 - ERROR - ERROR: Cannot divide 10 by zero
2025-10-26 22:49:40,026 - DEBUG - Division operation finished
2025-10-26 22:49:40,030 - DEBUG - Attempting division: 10 / 2
2025-10-26 22:49:40,032 - ERROR - ERROR: Type error - Inputs must be numbers
2025-10-26 22:49:40,034 - DEBUG - Division operation finished
2025-10-26 22:49:40,037 - DEBUG - Attempting division: 100 / 5
2025-10-26 22:49:40,040 - INFO - SUCCESS: 100 / 5 = 20.0
2025-10-26 22:49:40,043 - DEBUG - Division operation finished
2025-10-26 22:49:40,046 - DEBUG - Attempting to read file: nonexistent.txt
2025-10-26 22:49:40,048 - ERROR - ERROR: File 'nonexistent.txt' not found
2025-10-26 22:49:40,050 - INFO - RECOVERY: Creating file 'nonexistent.txt'
2025-10-26 22:49:40,053 - INFO - SUCCESS: File 'nonexistent.txt' created
2025-10-26 22:49:40,055 - DEBUG - File operation for 'nonexistent.txt' completed
2025-10-26 22:49:40,058 - DEBUG - Attempting to read file: nonexistent.txt
2025-10-26 22:49:40,084 - INFO - SUCCESS: File 'nonexistent.txt' read successfully
2025-10-26 22:49:40,101 - DEBUG - File operation for 'nonexistent.txt' completed
2025-10-26 22:49:40,104 - DEBUG - Accessing list[2] from list of size 5
...
2025-10-26 22:49:40,145 - ERROR - ERROR: Key 'country' not found in dictionary
2025-10-26 22:49:40,145 - INFO - Available keys: ['name', 'age', 'city']
2025-10-26 22:49:40,145 - DEBUG - Dictionary access operation completed
2025-10-26 22:49:40,158 - DEBUG - Accessing key 'key' from dictionary
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
======================================================================
🧪 EXPERIMENT 8: ERROR HANDLING - COMPLETE DEMONSTRATION
======================================================================

======================================================================
📝 STEP 1: Setting Up Error Logging System
======================================================================
✅ Created logs directory: logs
✅ Logging configured
□ Log file: logs/error_log_20251026_224939.txt

======================================================================
🛡️ STEP 2: Creating Error Handler Class
======================================================================
```



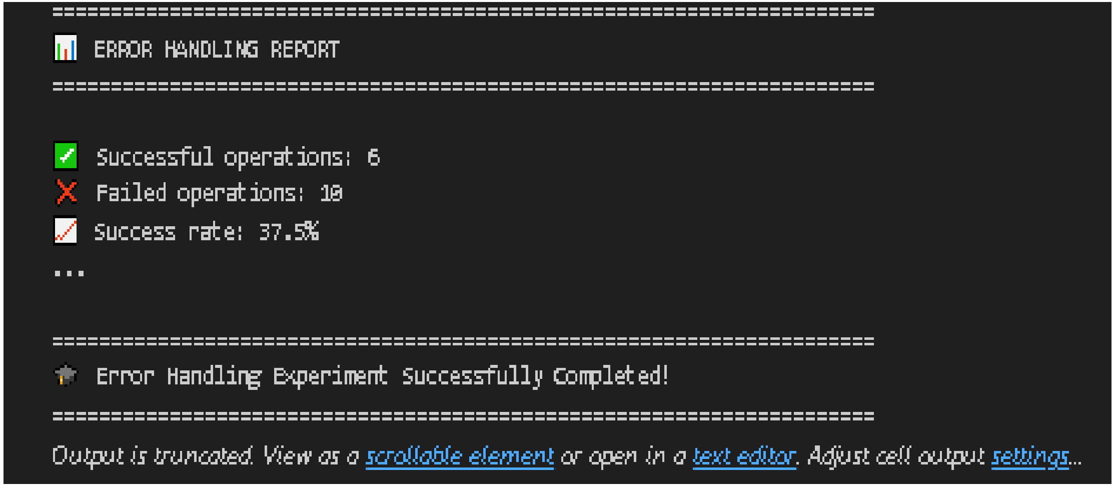
```python
            logger.debug("Dictionary access operation completed")

    # Handle User Input
    def get_user_age(self, age_input):
        """Validate user age input"""
        try:
            logger.debug(f"Validating age input: '{age_input}'")

            # Convert to integer
            age = int(age_input)

            # Validate range
            if age < 0:
                raise ValueError("Age cannot be negative")
            if age > 150:
                raise ValueError("Age seems unrealistic")
```



```
======================================================================
🛡 STEP 2: Creating Error Handler Class
======================================================================
✅ ErrorHandler created

======================================================================
🧪 STEP 3: Testing Division Error Handling
======================================================================

1️⃣ Valid division:
   Result: 5.0

2️⃣ Division by zero:
   Result: None
...
2️⃣ Key not found:
   Result: None
```



```
3️⃣ Type error (not a dict):
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
2025-10-26 22:49:40,164 - ERROR - ERROR: Type error - First argument must be a dictionary
2025-10-26 22:49:40,167 - DEBUG - Dictionary access operation completed
2025-10-26 22:49:40,170 - DEBUG - Validating age input: '25'
2025-10-26 22:49:40,174 - INFO - SUCCESS: Valid age 25
2025-10-26 22:49:40,178 - DEBUG - Age validation completed
2025-10-26 22:49:40,178 - DEBUG - Validating age input: 'twenty'
2025-10-26 22:49:40,178 - ERROR - ERROR: Invalid age - invalid literal for int() with base 10: 'twenty'
2025-10-26 22:49:40,178 - DEBUG - Age validation completed
2025-10-26 22:49:40,190 - DEBUG - Validating age input: '-5'
2025-10-26 22:49:40,196 - ERROR - ERROR: Invalid age - Age cannot be negative
2025-10-26 22:49:40,196 - DEBUG - Age validation completed
2025-10-26 22:49:40,231 - DEBUG - Validating age input: '200'
2025-10-26 22:49:40,231 - ERROR - ERROR: Invalid age - Age seems unrealistic
2025-10-26 22:49:40,231 - DEBUG - Age validation completed
2025-10-26 22:49:40,242 - INFO - Report generated: 6 success, 10 errors
2025-10-26 22:49:40,280 - INFO - ==================================================
2025-10-26 22:49:40,282 - INFO - Experiment 8: Error Handling Completed Successfully
2025-10-26 22:49:40,283 - INFO - ==================================================
   Result: None

======================================================================
🧪 STEP 7: Testing Input Validation
======================================================================

1️⃣ Valid age:
   Result: 25

2️⃣ Invalid age (text):
   Result: None

3️⃣ Invalid age (negative):
   Result: None

4️⃣ Invalid age (too high):
   Result: None

======================================================================
📊 ERROR HANDLING REPORT
======================================================================
```



```
======================================================================
📊 ERROR HANDLING REPORT
======================================================================

✅ Successful operations: 6
❌ Failed operations: 10
📈 Success rate: 37.5%
...

======================================================================
🎓 Error Handling Experiment Successfully Completed!
======================================================================
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

8.2 Explainable:

Error Handling is the process of anticipating, detecting, and responding to errors that occur during program execution to prevent crashes and ensure smooth operation. It uses try-except blocks to catch exceptions, validate inputs, and provide meaningful error messages to users. Proper error handling makes programs robust, user-friendly, and easier to debug by logging errors and handling edge cases gracefully. Python's exception handling mechanism allows developers to manage runtime errors, file operations, network issues, and invalid user inputs systematically.

8.3 Key Points

8.3.1. Exception Types

- **ValueError**: Invalid value/input (converting "abc" to integer)
- **ZeroDivisionError**: Division by zero
- **FileNotFoundError**: File doesn't exist
- **TypeError**: Wrong data type operation
- KeyError: Dictionary key not found

8.3 .2. Try-Except Structure

- try: Code that might cause error
- except: Handle specific exceptions
- else: Runs if no error occurs
- finally: Always executes (cleanup code)
- Logging errors for debugging

8.3 .3. Best Practices

- Catch specific exceptions, not generic ones
- Provide meaningful error messages to users
- Log errors with timestamp and details
- Validate inputs before processing
- Use finally for cleanup (close files, connections)

Experiment 9: UML Diagrams

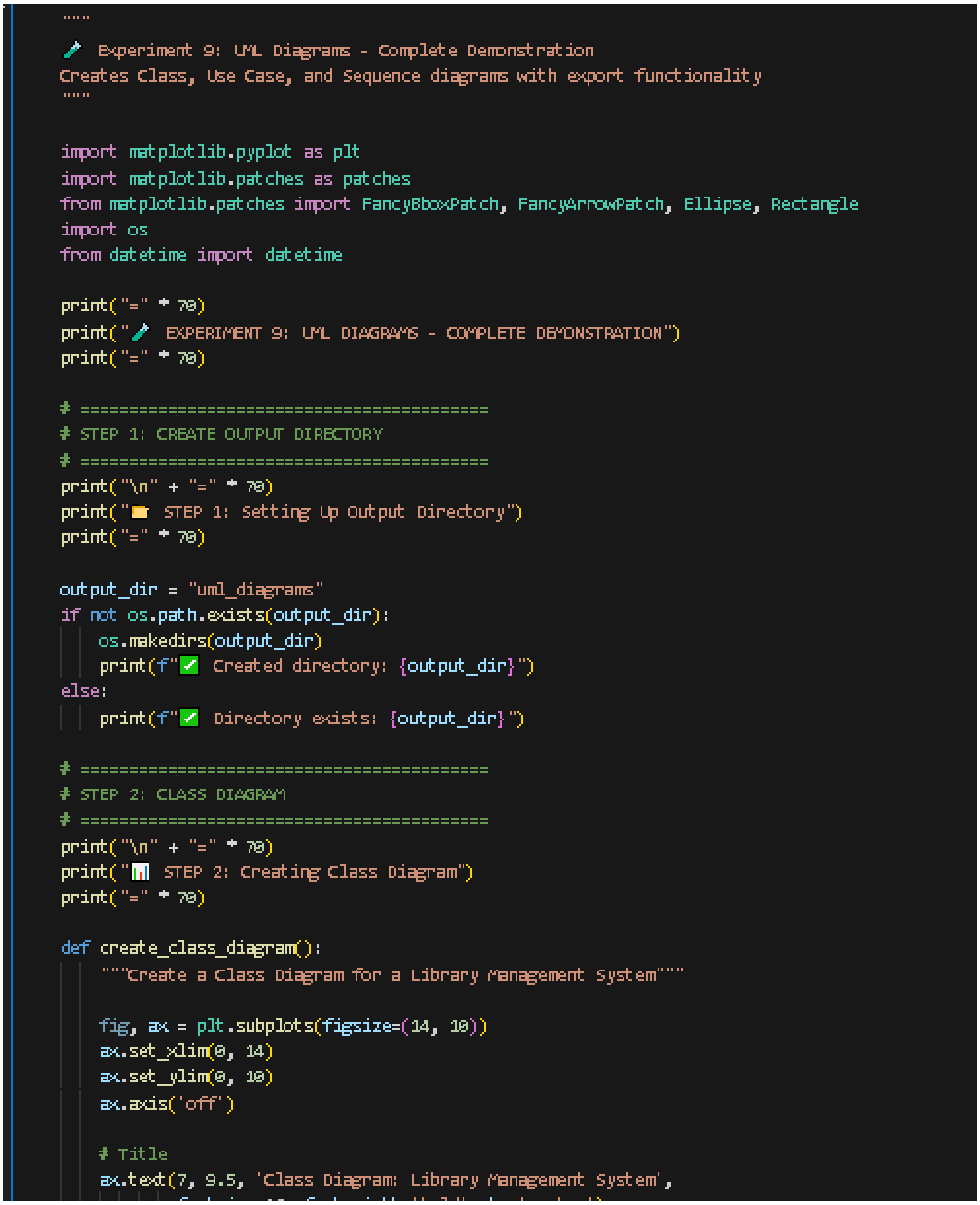
Objective: Design using UML

Tasks:

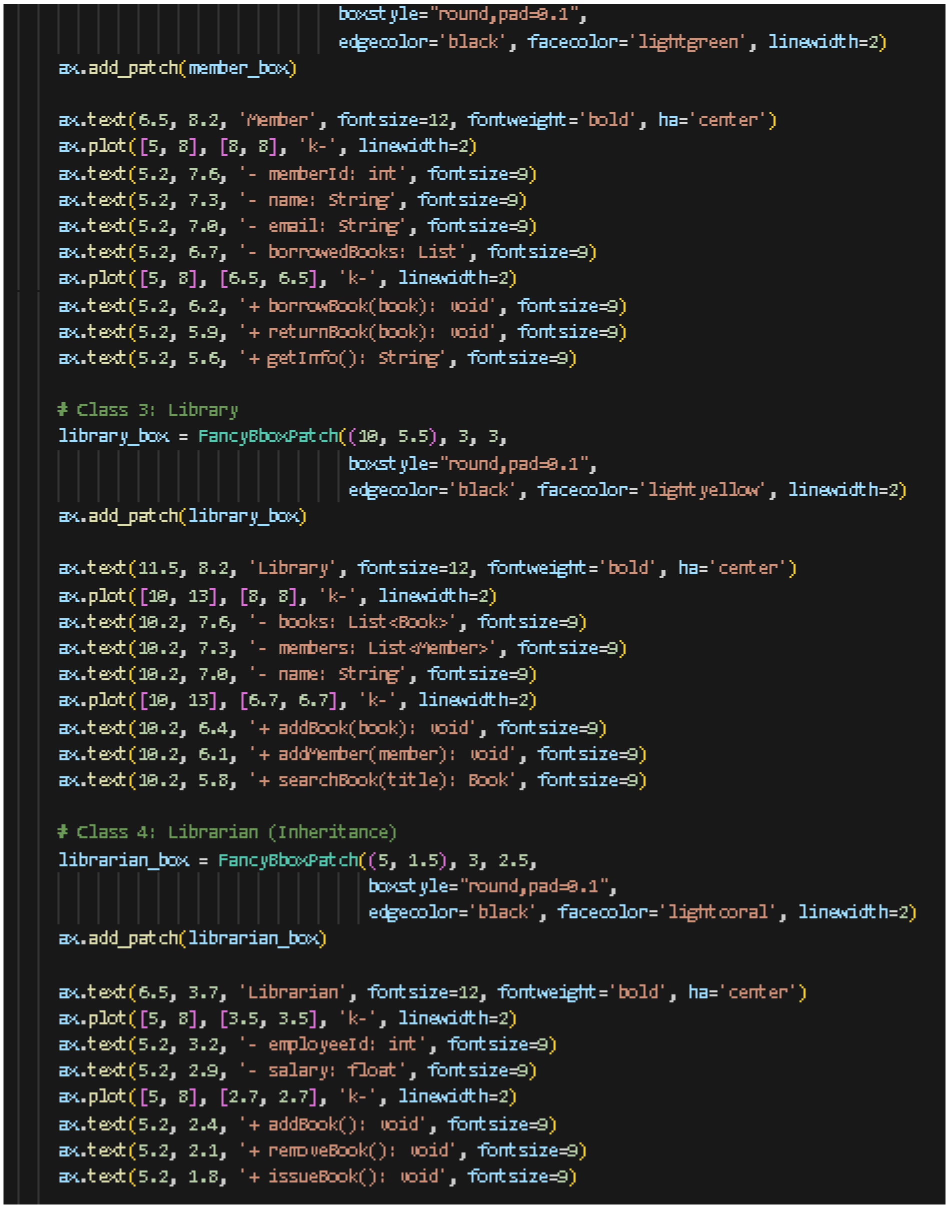
Draw class, use case, and sequence diagrams

Deliverables: Diagram images or PDFs

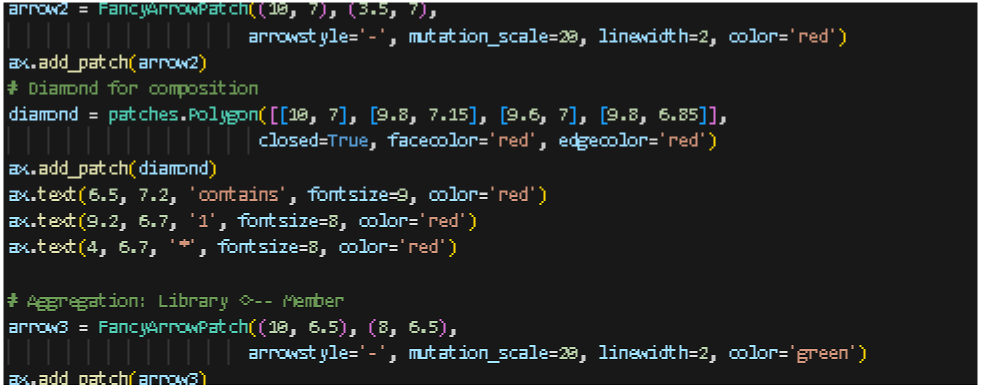
9.1 Inputs & Outputs:



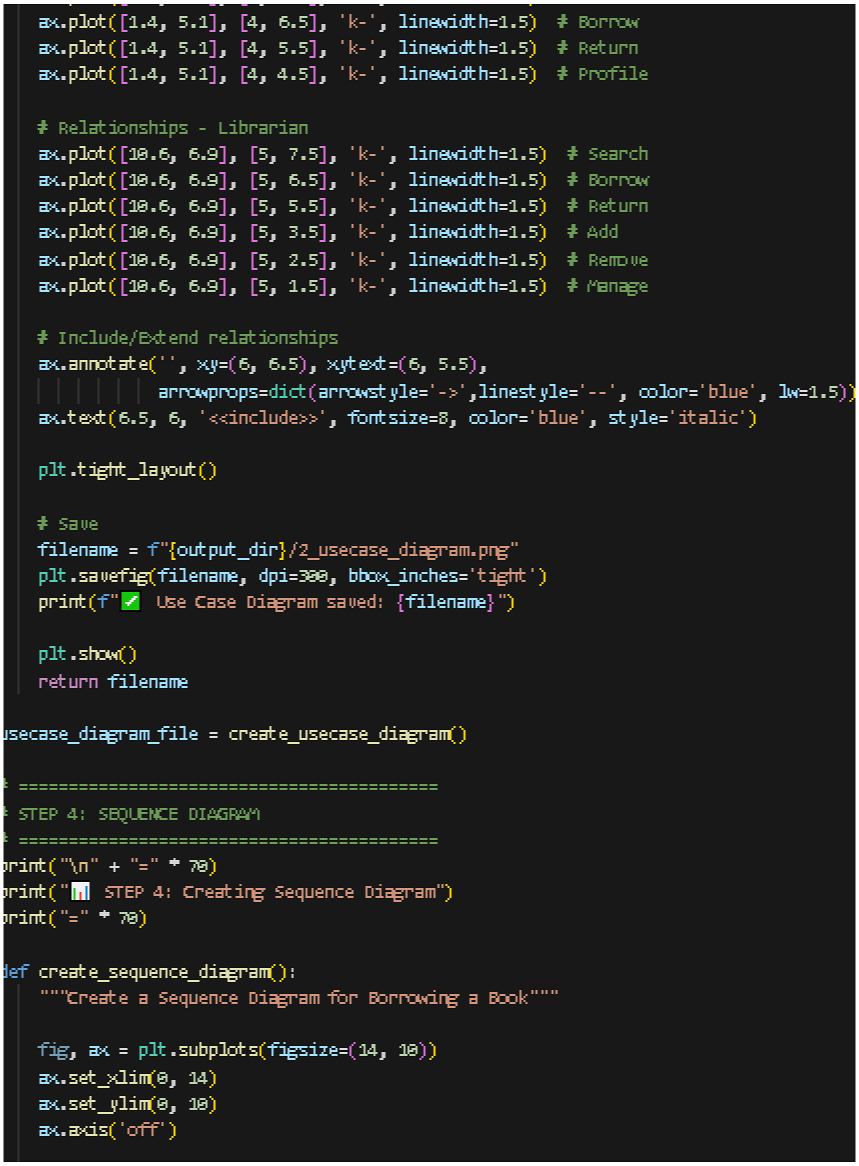
```python
"""
🧪 Experiment 9: UML Diagrams - Complete Demonstration
Creates Class, Use Case, and Sequence diagrams with export functionality
"""

import matplotlib.pyplot as plt
import matplotlib.patches as patches
from matplotlib.patches import FancyBboxPatch, FancyArrowPatch, Ellipse, Rectangle
import os
from datetime import datetime

print("=" * 70)
print("🧪 EXPERIMENT 9: UML DIAGRAMS - COMPLETE DEMONSTRATION")
print("=" * 70)

# ============================================
# STEP 1: CREATE OUTPUT DIRECTORY
# ============================================
print("\n" + "=" * 70)
print("📁 STEP 1: Setting Up Output Directory")
print("=" * 70)

output_dir = "uml_diagrams"
if not os.path.exists(output_dir):
    os.makedirs(output_dir)
    print(f"✅ Created directory: {output_dir}")
else:
    print(f"✅ Directory exists: {output_dir}")

# ============================================
# STEP 2: CLASS DIAGRAM
# ============================================
print("\n" + "=" * 70)
print("📊 STEP 2: Creating Class Diagram")
print("=" * 70)

def create_class_diagram():
    """Create a Class Diagram for a Library Management System"""

    fig, ax = plt.subplots(figsize=(14, 10))
    ax.set_xlim(0, 14)
    ax.set_ylim(0, 10)
    ax.axis('off')

    # Title
    ax.text(7, 9.5, 'Class Diagram: Library Management System',
```



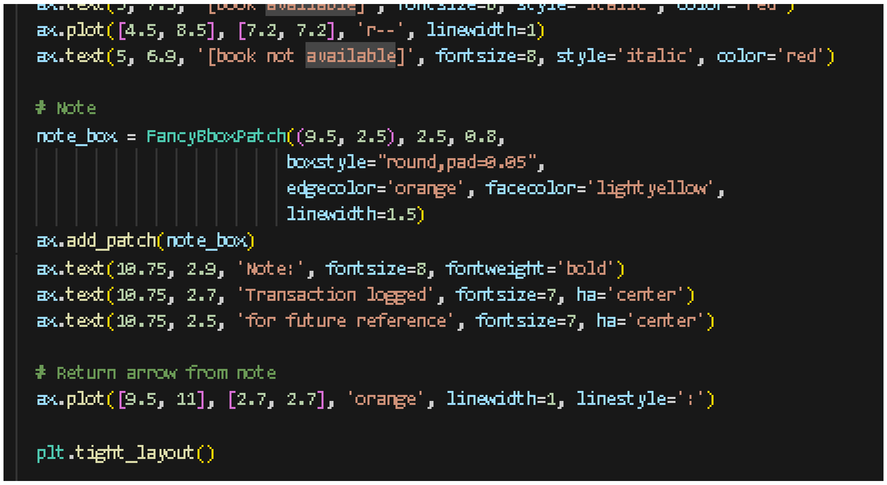
```
                              boxstyle="round,pad=0.1",
                              edgecolor='black', facecolor='lightgreen', linewidth=2)
    ax.add_patch(member_box)

    ax.text(6.5, 8.2, 'Member', fontsize=12, fontweight='bold', ha='center')
    ax.plot([5, 8], [8, 8], 'k-', linewidth=2)
    ax.text(5.2, 7.6, '- memberId: int', fontsize=9)
    ax.text(5.2, 7.3, '- name: String', fontsize=9)
    ax.text(5.2, 7.0, '- email: String', fontsize=9)
    ax.text(5.2, 6.7, '- borrowedBooks: List', fontsize=9)
    ax.plot([5, 8], [6.5, 6.5], 'k-', linewidth=2)
    ax.text(5.2, 6.2, '+ borrowBook(book): void', fontsize=9)
    ax.text(5.2, 5.9, '+ returnBook(book): void', fontsize=9)
    ax.text(5.2, 5.6, '+ getInfo(): String', fontsize=9)

    # Class 3: Library
    library_box = FancyBboxPatch((10, 5.5), 3, 3,
                                boxstyle="round,pad=0.1",
                                edgecolor='black', facecolor='lightyellow', linewidth=2)
    ax.add_patch(library_box)

    ax.text(11.5, 8.2, 'Library', fontsize=12, fontweight='bold', ha='center')
    ax.plot([10, 13], [8, 8], 'k-', linewidth=2)
    ax.text(10.2, 7.6, '- books: List<Book>', fontsize=9)
    ax.text(10.2, 7.3, '- members: List<Member>', fontsize=9)
    ax.text(10.2, 7.0, '- name: String', fontsize=9)
    ax.plot([10, 13], [6.7, 6.7], 'k-', linewidth=2)
    ax.text(10.2, 6.4, '+ addBook(book): void', fontsize=9)
    ax.text(10.2, 6.1, '+ addMember(member): void', fontsize=9)
    ax.text(10.2, 5.8, '+ searchBook(title): Book', fontsize=9)

    # Class 4: Librarian (Inheritance)
    librarian_box = FancyBboxPatch((5, 1.5), 3, 2.5,
                                  boxstyle="round,pad=0.1",
                                  edgecolor='black', facecolor='lightcoral', linewidth=2)
    ax.add_patch(librarian_box)

    ax.text(6.5, 3.7, 'Librarian', fontsize=12, fontweight='bold', ha='center')
    ax.plot([5, 8], [3.5, 3.5], 'k-', linewidth=2)
    ax.text(5.2, 3.2, '- employeeId: int', fontsize=9)
    ax.text(5.2, 2.9, '- salary: float', fontsize=9)
    ax.plot([5, 8], [2.7, 2.7], 'k-', linewidth=2)
    ax.text(5.2, 2.4, '+ addBook(): void', fontsize=9)
    ax.text(5.2, 2.1, '+ removeBook(): void', fontsize=9)
    ax.text(5.2, 1.8, '+ issueBook(): void', fontsize=9)
```



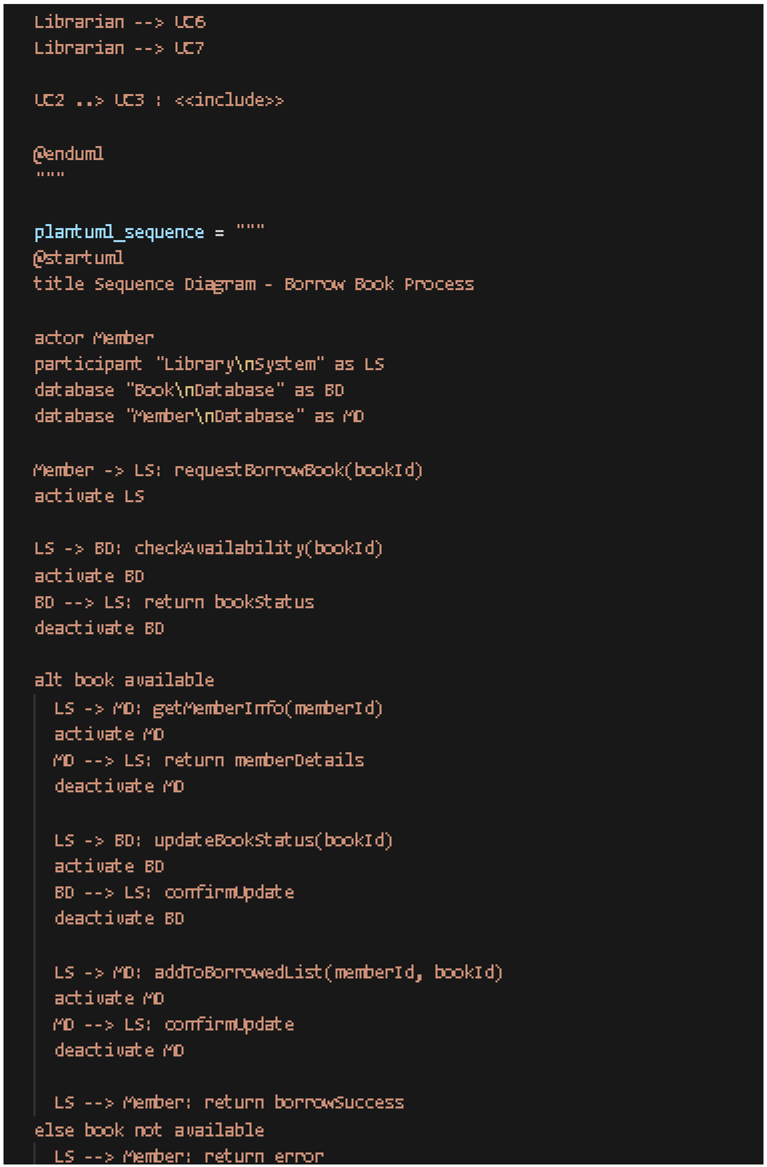
```
arrow2 = FancyArrowPatch((10, 7), (3.5, 7),
                        arrowstyle='-', mutation_scale=20, linewidth=2, color='red')
ax.add_patch(arrow2)
# Diamond for composition
diamond = patches.Polygon([[10, 7], [9.8, 7.15], [9.6, 7], [9.8, 6.85]],
                         closed=True, facecolor='red', edgecolor='red')
ax.add_patch(diamond)
ax.text(6.5, 7.2, 'contains', fontsize=9, color='red')
ax.text(9.2, 6.7, '1', fontsize=8, color='red')
ax.text(4, 6.7, '*', fontsize=8, color='red')

# Aggregation: Library <>-- Member
arrow3 = FancyArrowPatch((10, 6.5), (8, 6.5),
                        arrowstyle='-', mutation_scale=20, linewidth=2, color='green')
ax.add_patch(arrow3)
```



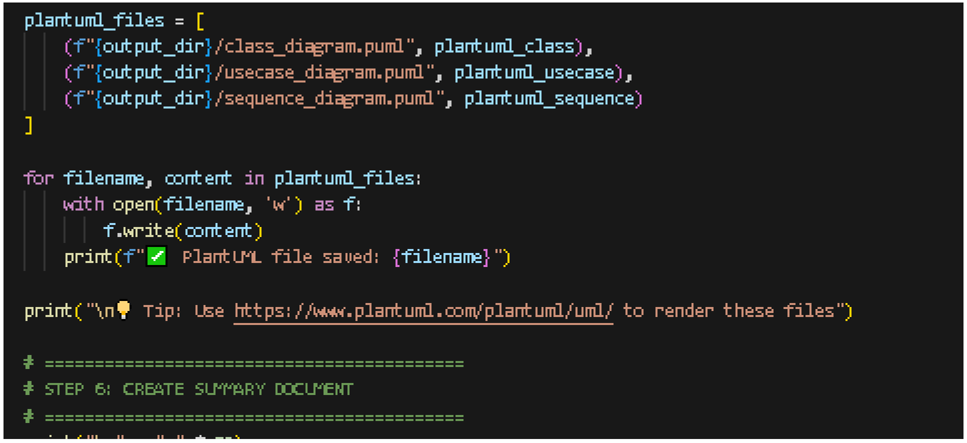
```
    ax.plot([1.4, 5.1], [4, 6.5], 'k-', linewidth=1.5)  # Borrow
    ax.plot([1.4, 5.1], [4, 5.5], 'k-', linewidth=1.5)  # Return
    ax.plot([1.4, 5.1], [4, 4.5], 'k-', linewidth=1.5)  # Profile

    # Relationships - Librarian
    ax.plot([10.6, 6.9], [5, 7.5], 'k-', linewidth=1.5)  # Search
    ax.plot([10.6, 6.9], [5, 6.5], 'k-', linewidth=1.5)  # Borrow
    ax.plot([10.6, 6.9], [5, 5.5], 'k-', linewidth=1.5)  # Return
    ax.plot([10.6, 6.9], [5, 3.5], 'k-', linewidth=1.5)  # Add
    ax.plot([10.6, 6.9], [5, 2.5], 'k-', linewidth=1.5)  # Remove
    ax.plot([10.6, 6.9], [5, 1.5], 'k-', linewidth=1.5)  # Manage

    # Include/Extend relationships
    ax.annotate('', xy=(6, 6.5), xytext=(6, 5.5),
                arrowprops=dict(arrowstyle='->',linestyle='--', color='blue', lw=1.5))
    ax.text(6.5, 6, '<<include>>', fontsize=8, color='blue', style='italic')

    plt.tight_layout()

    # Save
    filename = f"{output_dir}/2_usecase_diagram.png"
    plt.savefig(filename, dpi=300, bbox_inches='tight')
    print(f"✅ Use Case Diagram saved: {filename}")

    plt.show()
    return filename

usecase_diagram_file = create_usecase_diagram()

# ============================================
# STEP 4: SEQUENCE DIAGRAM
# ============================================
print("\n" + "=" * 70)
print("📊 STEP 4: Creating Sequence Diagram")
print("=" * 70)

def create_sequence_diagram():
    """Create a Sequence Diagram for Borrowing a Book"""

    fig, ax = plt.subplots(figsize=(14, 10))
    ax.set_xlim(0, 14)
    ax.set_ylim(0, 10)
    ax.axis('off')
```



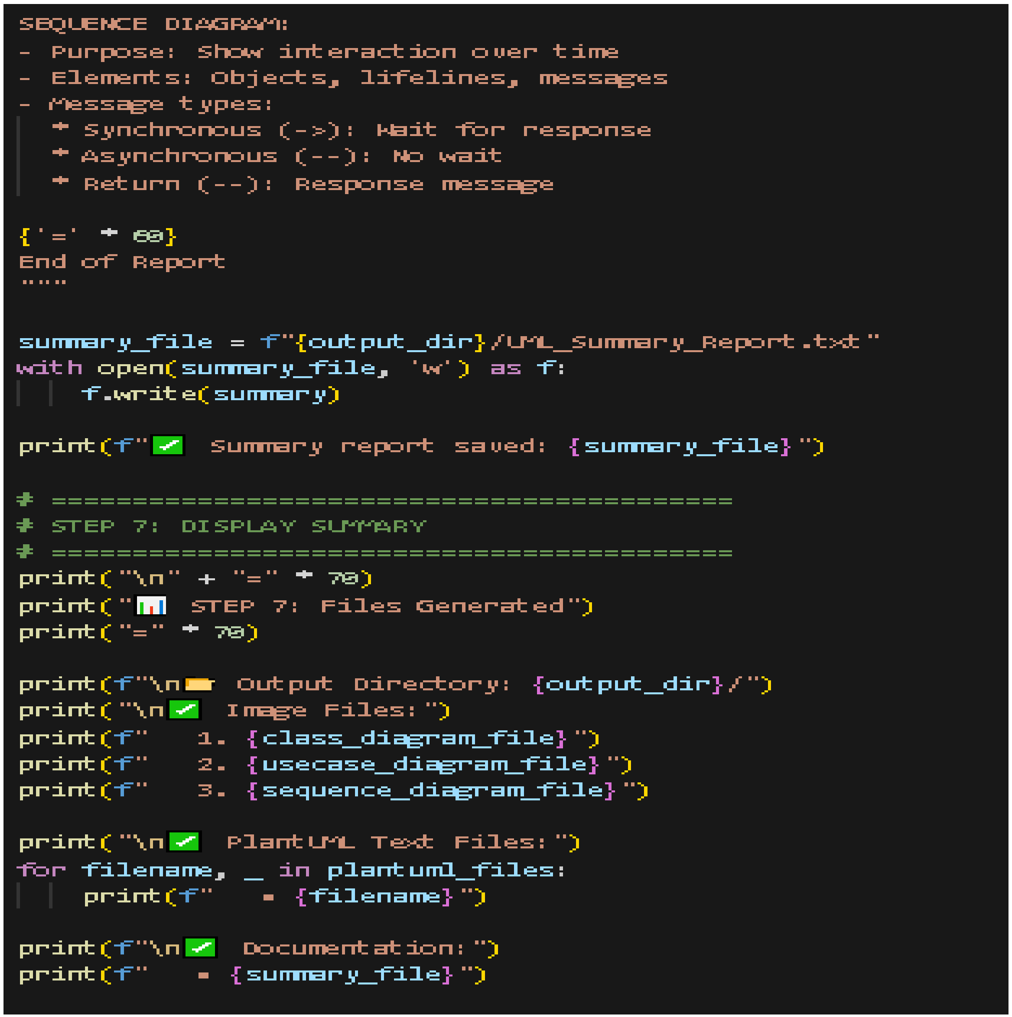
```
    ax.plot([4.5, 8.5], [7.2, 7.2], 'r--', linewidth=1)
    ax.text(5, 6.9, '[book not available]', fontsize=8, style='italic', color='red')

    # Note
    note_box = FancyBboxPatch((9.5, 2.5), 2.5, 0.8,
                              boxstyle="round,pad=0.05",
                              edgecolor='orange', facecolor='lightyellow',
                              linewidth=1.5)
    ax.add_patch(note_box)
    ax.text(10.75, 2.9, 'Note:', fontsize=8, fontweight='bold')
    ax.text(10.75, 2.7, 'Transaction logged', fontsize=7, ha='center')
    ax.text(10.75, 2.5, 'for future reference', fontsize=7, ha='center')

    # Return arrow from note
    ax.plot([9.5, 11], [2.7, 2.7], 'orange', linewidth=1, linestyle=':')

    plt.tight_layout()
```



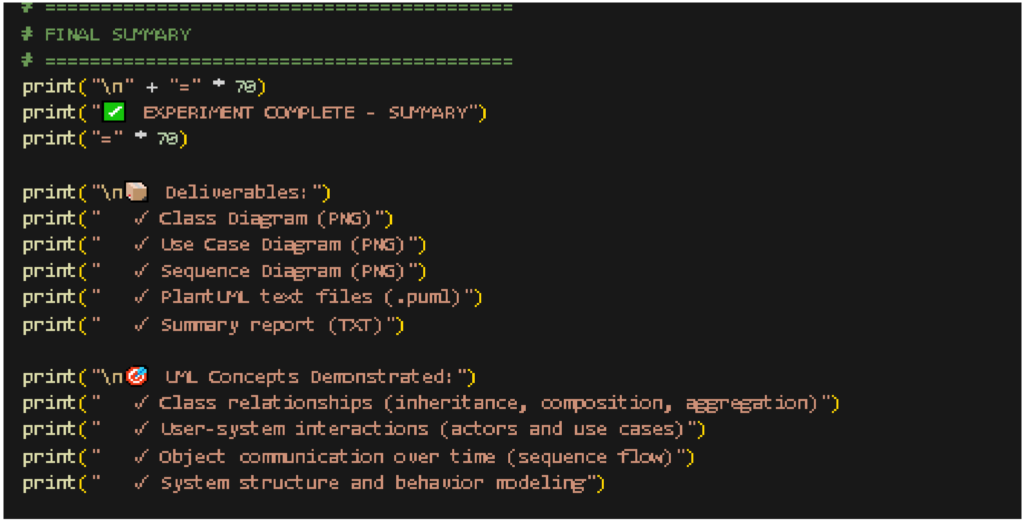
```
Librarian --> UC6
Librarian --> UC7

UC2 ..> UC3 : <<include>>

@enduml
"""

plantuml_sequence = """
@startuml
title Sequence Diagram - Borrow Book Process

actor Member
participant "Library\nSystem" as LS
database "Book\nDatabase" as BD
database "Member\nDatabase" as MD

Member -> LS: requestBorrowBook(bookId)
activate LS

LS -> BD: checkAvailability(bookId)
activate BD
BD --> LS: return bookStatus
deactivate BD

alt book available
  LS -> MD: getMemberInfo(memberId)
  activate MD
  MD --> LS: return memberDetails
  deactivate MD

  LS -> BD: updateBookStatus(bookId)
  activate BD
  BD --> LS: confirmUpdate
  deactivate BD

  LS -> MD: addToBorrowedList(memberId, bookId)
  activate MD
  MD --> LS: confirmUpdate
  deactivate MD

  LS --> Member: return borrowSuccess
else book not available
  LS --> Member: return error
```



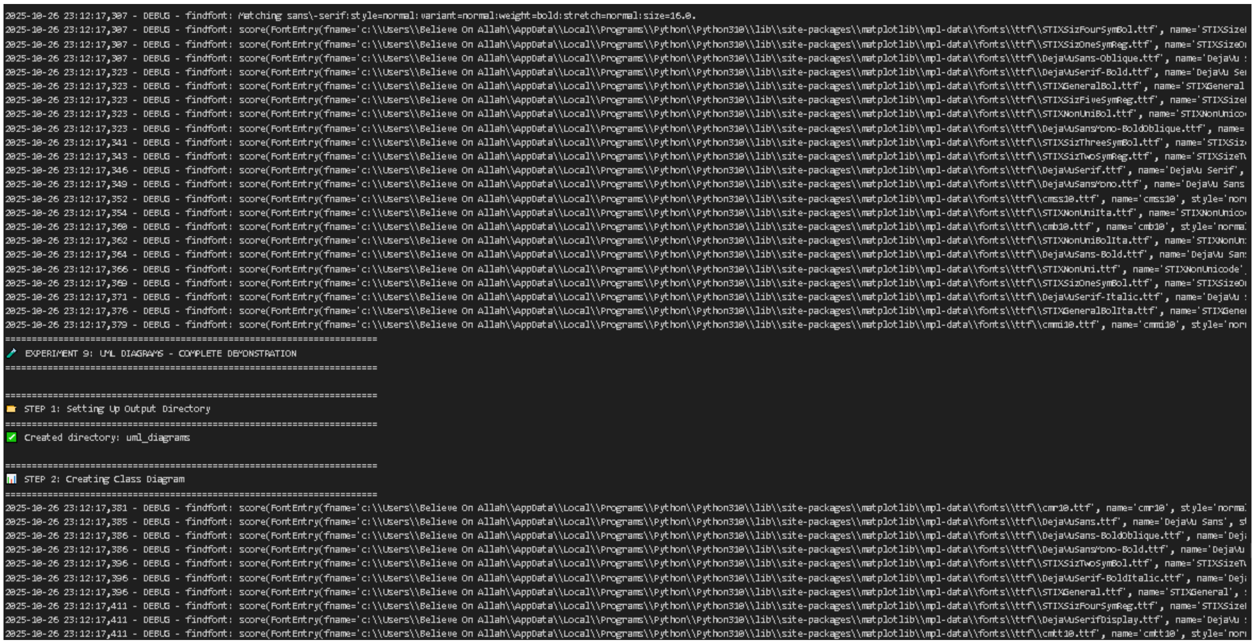
```
plantuml_files = [
    (f"{output_dir}/class_diagram.puml", plantuml_class),
    (f"{output_dir}/usecase_diagram.puml", plantuml_usecase),
    (f"{output_dir}/sequence_diagram.puml", plantuml_sequence)
]

for filename, content in plantuml_files:
    with open(filename, 'w') as f:
        f.write(content)
    print(f"✅ PlantUML file saved: {filename}")

print("\n💡 Tip: Use https://www.plantuml.com/plantuml/uml/ to render these files")

# ========================================
# STEP 6: CREATE SUMMARY DOCUMENT
# ========================================
```



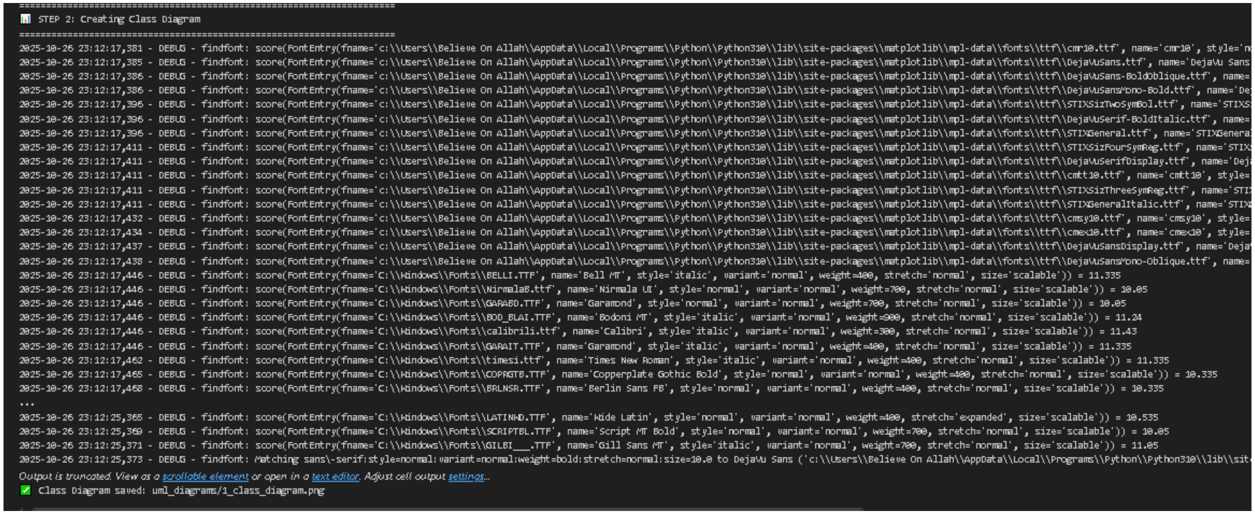
```
SEQUENCE DIAGRAM:
- Purpose: Show interaction over time
- Elements: Objects, lifelines, messages
- Message types:
  * Synchronous (->): Wait for response
  * Asynchronous (--): No wait
  * Return (--): Response message

{'=' * 60}
End of Report
"""

summary_file = f"{output_dir}/UML_Summary_Report.txt"
with open(summary_file, 'w') as f:
    f.write(summary)

print(f"✅ Summary report saved: {summary_file}")

# ============================================
# STEP 7: DISPLAY SUMMARY
# ============================================
print("\n" + "=" * 70)
print("📊 STEP 7: Files Generated")
print("=" * 70)

print(f"\n📁 Output Directory: {output_dir}/")
print("\n✅ Image Files:")
print(f"   1. {class_diagram_file}")
print(f"   2. {usecase_diagram_file}")
print(f"   3. {sequence_diagram_file}")

print("\n✅ PlantUML Text Files:")
for filename, _ in plantuml_files:
    print(f"   • {filename}")

print(f"\n✅ Documentation:")
print(f"   • {summary_file}")
```



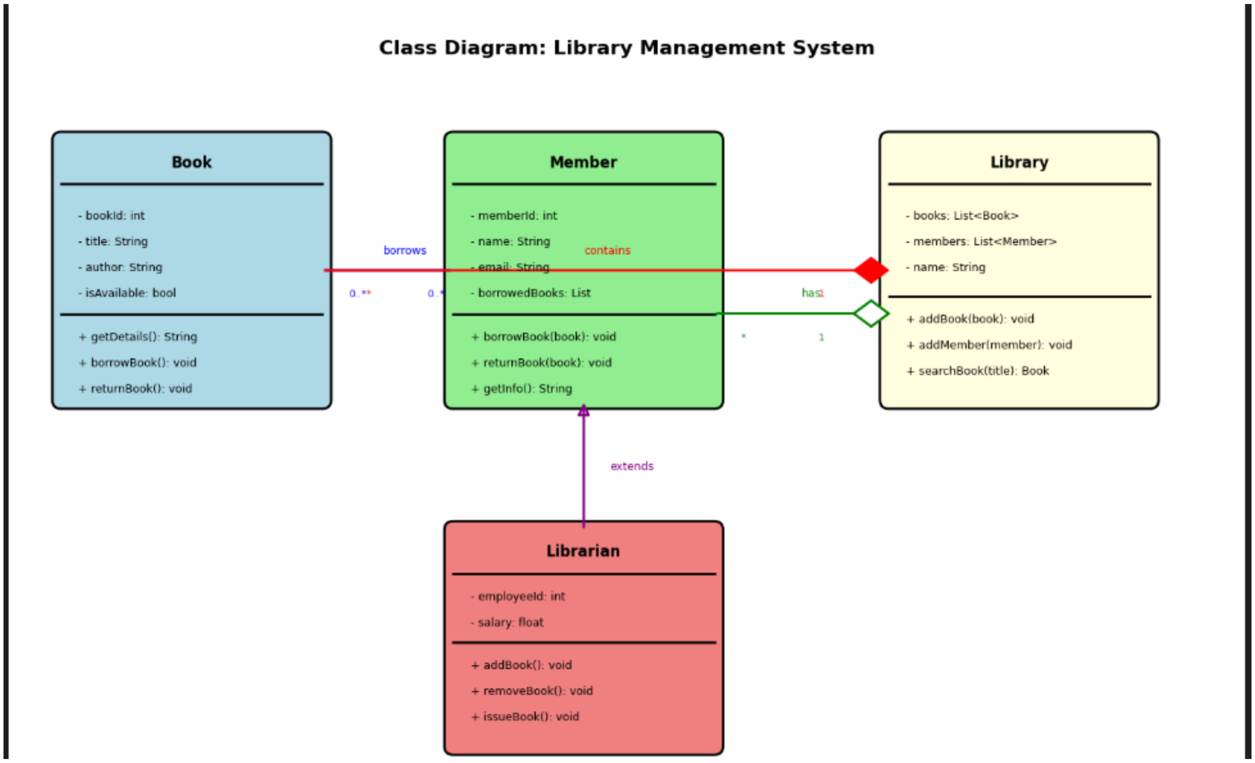
```
# ============================================
# FINAL SUMMARY
# ============================================
print("\n" + "=" * 70)
print("✅ EXPERIMENT COMPLETE - SUMMARY")
print("=" * 70)

print("\n📦 Deliverables:")
print("   ✓ Class Diagram (PNG)")
print("   ✓ Use Case Diagram (PNG)")
print("   ✓ Sequence Diagram (PNG)")
print("   ✓ PlantUML text files (.puml)")
print("   ✓ Summary report (TXT)")

print("\n🎯 UML Concepts Demonstrated:")
print("   ✓ Class relationships (inheritance, composition, aggregation)")
print("   ✓ User-system interactions (actors and use cases)")
print("   ✓ Object communication over time (sequence flow)")
print("   ✓ System structure and behavior modeling")
```

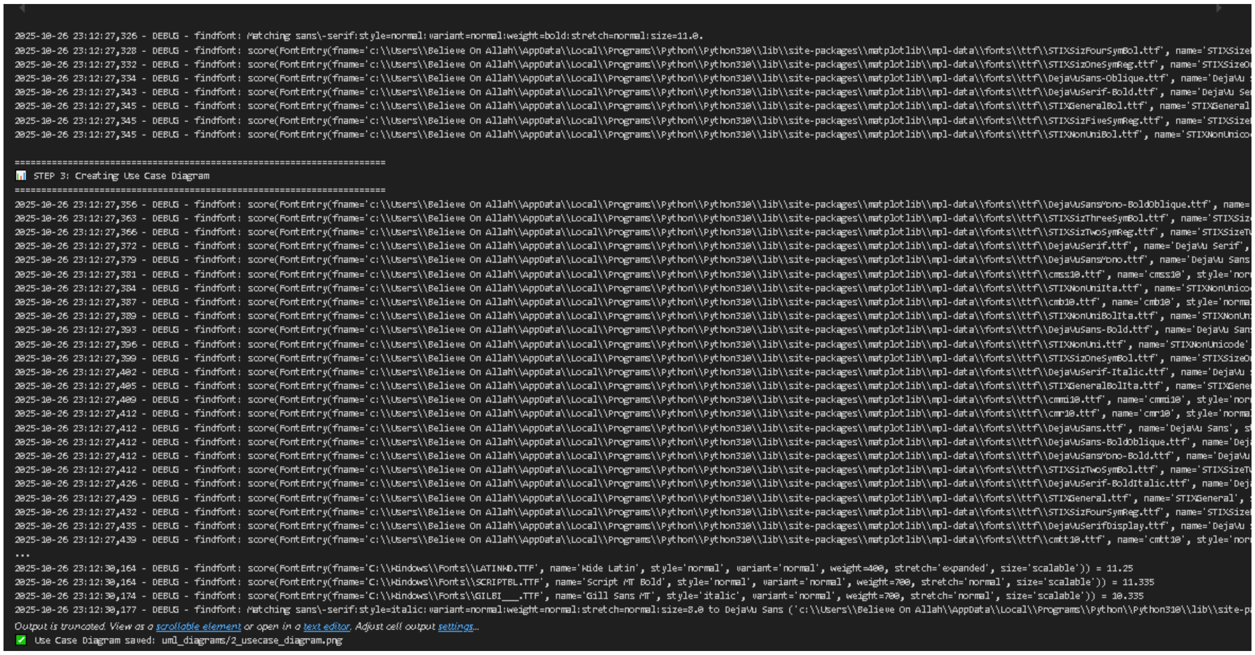


```
2025-10-26 23:12:17,307 - DEBUG - findfont: Matching sans\-serif:style=normal:variant=normal:weight=bold:stretch=normal:size=16.0.
2025-10-26 23:12:17,307 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\STIXSizFourSymBol.ttf', name='STIXSize
2025-10-26 23:12:17,307 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\STIXSizOneSymReg.ttf', name='STIXSizeO
2025-10-26 23:12:17,307 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\DejaVuSans-Oblique.ttf', name='DejaVu
2025-10-26 23:12:17,323 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\DejaVuSerif-Bold.ttf', name='DejaVu Se
2025-10-26 23:12:17,323 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\STIXGeneralBol.ttf', name='STIXGeneral
2025-10-26 23:12:17,323 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\STIXSizFiveSymReg.ttf', name='STIXSize
2025-10-26 23:12:17,323 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\STIXNonUniBol.ttf', name='STIXNonUnico
2025-10-26 23:12:17,323 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\DejaVuSansMono-BoldOblique.ttf', name=
2025-10-26 23:12:17,341 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\STIXSizThreeSymBol.ttf', name='STIXSiz
2025-10-26 23:12:17,343 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\STIXSizTwoSymReg.ttf', name='STIXSizeT
2025-10-26 23:12:17,346 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\DejaVuSerif.ttf', name='DejaVu Serif',
2025-10-26 23:12:17,349 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\DejaVuSansMono.ttf', name='DejaVu Sans
2025-10-26 23:12:17,352 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\cmss10.ttf', name='cmss10', style='nor
2025-10-26 23:12:17,354 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\STIXNonUniIta.ttf', name='STIXNonUnico
2025-10-26 23:12:17,360 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\cmb10.ttf', name='cmb10', style='norma
2025-10-26 23:12:17,362 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\STIXNonUniBolIta.ttf', name='STIXNonUn
2025-10-26 23:12:17,364 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\DejaVuSans-Bold.ttf', name='DejaVu San
2025-10-26 23:12:17,366 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\STIXNonUni.ttf', name='STIXNonUnicode',
2025-10-26 23:12:17,369 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\STIXSizOneSymBol.ttf', name='STIXSizeO
2025-10-26 23:12:17,371 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\DejaVuSerif-Italic.ttf', name='DejaVu
2025-10-26 23:12:17,376 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\STIXGeneralBolIta.ttf', name='STIXGene
2025-10-26 23:12:17,379 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\cmmi10.ttf', name='cmmi10', style='nor
======================================================================
🧪 EXPERIMENT 9: UML DIAGRAMS - COMPLETE DEMONSTRATION
======================================================================

======================================================================
📁 STEP 1: Setting Up Output Directory
======================================================================
✅ Created directory: uml_diagrams

======================================================================
📊 STEP 2: Creating Class Diagram
======================================================================
2025-10-26 23:12:17,381 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\cmr10.ttf', name='cmr10', style='norma
2025-10-26 23:12:17,385 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\DejaVuSans.ttf', name='DejaVu Sans', s
2025-10-26 23:12:17,386 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\DejaVuSans-BoldOblique.ttf', name='Dej
2025-10-26 23:12:17,386 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\DejaVuSansMono-Bold.ttf', name='DejaVu
2025-10-26 23:12:17,396 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\STIXSizTwoSymBol.ttf', name='STIXSizeT
2025-10-26 23:12:17,396 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\DejaVuSerif-BoldItalic.ttf', name='Dej
2025-10-26 23:12:17,396 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\STIXGeneral.ttf', name='STIXGeneral',
2025-10-26 23:12:17,411 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\STIXSizFourSymReg.ttf', name='STIXSize
2025-10-26 23:12:17,411 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\DejaVuSerifDisplay.ttf', name='DejaVu
2025-10-26 23:12:17,411 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\cmtt10.ttf', name='cmtt10', style='nor
```

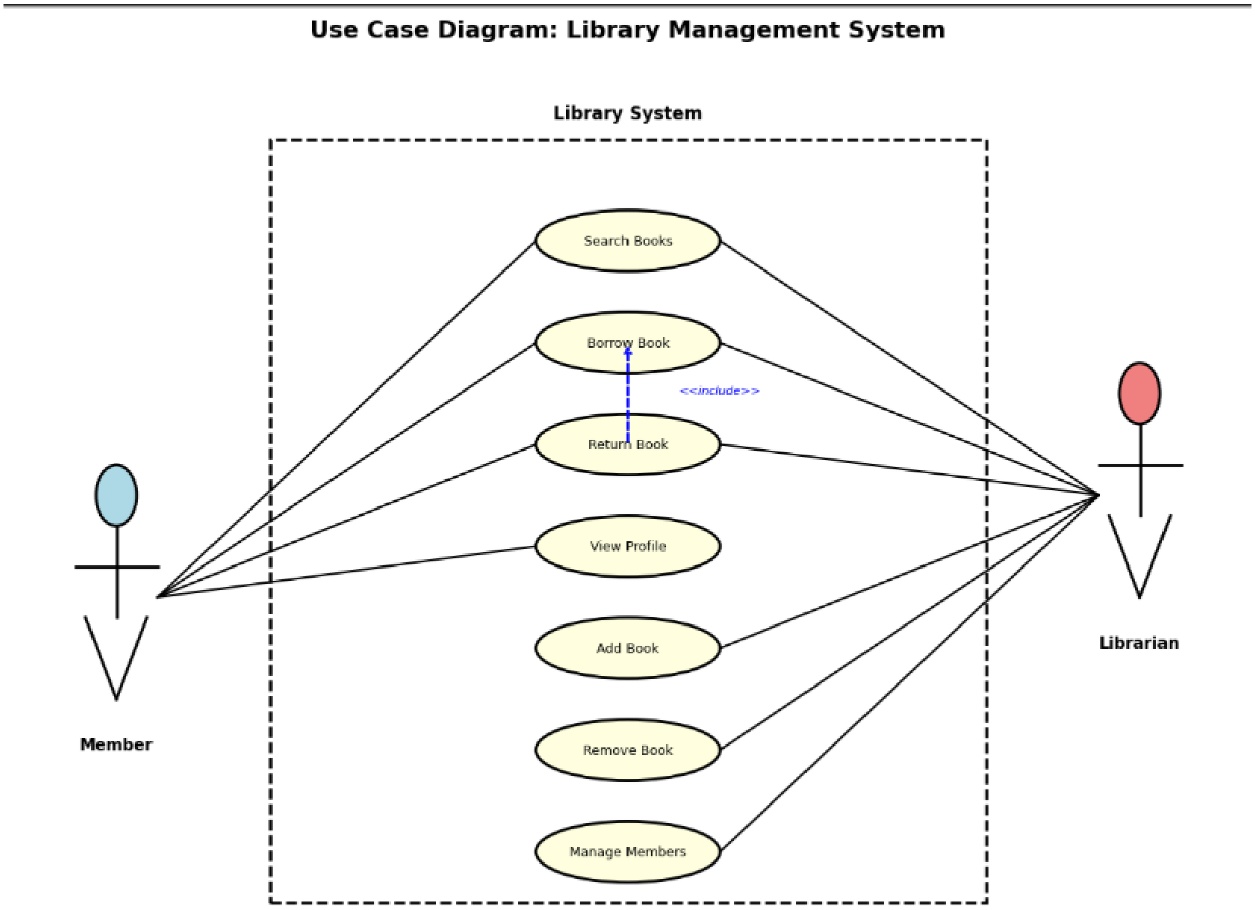


```
======================================================================
📊 STEP 2: Creating Class Diagram
======================================================================
2025-10-26 23:12:17,381 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\cmr10.ttf', name='cmr10', style='n
2025-10-26 23:12:17,385 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\DejaVuSans.ttf', name='DejaVu Sans
2025-10-26 23:12:17,386 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\DejaVuSans-BoldOblique.ttf', name=
2025-10-26 23:12:17,386 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\DejaVuSansMono-Bold.ttf', name='De
2025-10-26 23:12:17,396 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\STIXSizTwoSymBol.ttf', name='STIXS
2025-10-26 23:12:17,396 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\DejaVuSerif-BoldItalic.ttf', name=
2025-10-26 23:12:17,396 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\STIXGeneral.ttf', name='STIXGenera
2025-10-26 23:12:17,411 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\STIXSizFourSymReg.ttf', name='STIX
2025-10-26 23:12:17,411 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\DejaVuSerifDisplay.ttf', name='Deja
2025-10-26 23:12:17,411 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\cmtt10.ttf', name='cmtt10', style=
2025-10-26 23:12:17,411 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\STIXSizThreeSymReg.ttf', name='STIX
2025-10-26 23:12:17,411 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\STIXGeneralItalic.ttf', name='STIX
2025-10-26 23:12:17,432 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\cmsy10.ttf', name='cmsy10', style=
2025-10-26 23:12:17,434 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\cmex10.ttf', name='cmex10', style=
2025-10-26 23:12:17,437 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\DejaVuSansDisplay.ttf', name='Deja
2025-10-26 23:12:17,438 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\DejaVuSansMono-Oblique.ttf', name=
2025-10-26 23:12:17,446 - DEBUG - findfont: score(FontEntry(fname='C:\\Windows\\Fonts\\BELLI.TTF', name='Bell MT', style='italic', variant='normal', weight=400, stretch='normal', size='scalable')) = 11.335
2025-10-26 23:12:17,446 - DEBUG - findfont: score(FontEntry(fname='C:\\Windows\\Fonts\\NirmalaB.ttf', name='Nirmala UI', style='normal', variant='normal', weight=700, stretch='normal', size='scalable')) = 10.05
2025-10-26 23:12:17,446 - DEBUG - findfont: score(FontEntry(fname='C:\\Windows\\Fonts\\GARABD.TTF', name='Garamond', style='normal', variant='normal', weight=700, stretch='normal', size='scalable')) = 10.05
2025-10-26 23:12:17,446 - DEBUG - findfont: score(FontEntry(fname='C:\\Windows\\Fonts\\BOD_BLAI.TTF', name='Bodoni MT', style='italic', variant='normal', weight=900, stretch='normal', size='scalable')) = 11.24
2025-10-26 23:12:17,446 - DEBUG - findfont: score(FontEntry(fname='C:\\Windows\\Fonts\\calibrili.ttf', name='Calibri', style='italic', variant='normal', weight=300, stretch='normal', size='scalable')) = 11.43
2025-10-26 23:12:17,446 - DEBUG - findfont: score(FontEntry(fname='C:\\Windows\\Fonts\\GARAIT.TTF', name='Garamond', style='italic', variant='normal', weight=400, stretch='normal', size='scalable')) = 11.335
2025-10-26 23:12:17,462 - DEBUG - findfont: score(FontEntry(fname='C:\\Windows\\Fonts\\timesi.ttf', name='Times New Roman', style='italic', variant='normal', weight=400, stretch='normal', size='scalable')) = 11.335
2025-10-26 23:12:17,465 - DEBUG - findfont: score(FontEntry(fname='C:\\Windows\\Fonts\\COPRGTB.TTF', name='Copperplate Gothic Bold', style='normal', variant='normal', weight=400, stretch='normal', size='scalable')) = 10.335
2025-10-26 23:12:17,468 - DEBUG - findfont: score(FontEntry(fname='C:\\Windows\\Fonts\\BRLNSR.TTF', name='Berlin Sans FB', style='normal', variant='normal', weight=400, stretch='normal', size='scalable')) = 10.335
...
2025-10-26 23:12:25,365 - DEBUG - findfont: score(FontEntry(fname='C:\\Windows\\Fonts\\LATINWD.TTF', name='Wide Latin', style='normal', variant='normal', weight=400, stretch='expanded', size='scalable')) = 10.535
2025-10-26 23:12:25,369 - DEBUG - findfont: score(FontEntry(fname='C:\\Windows\\Fonts\\SCRIPTBL.TTF', name='Script MT Bold', style='normal', variant='normal', weight=700, stretch='normal', size='scalable')) = 10.05
2025-10-26 23:12:25,371 - DEBUG - findfont: score(FontEntry(fname='C:\\Windows\\Fonts\\GILBI___.TTF', name='Gill Sans MT', style='italic', variant='normal', weight=700, stretch='normal', size='scalable')) = 11.05
2025-10-26 23:12:25,373 - DEBUG - findfont: Matching sans\-serif:style=normal:variant=normal:weight=bold:stretch=normal:size=10.0 to DejaVu Sans ('c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\sit
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
✅ Class Diagram saved: uml_diagrams/1_class_diagram.png
```

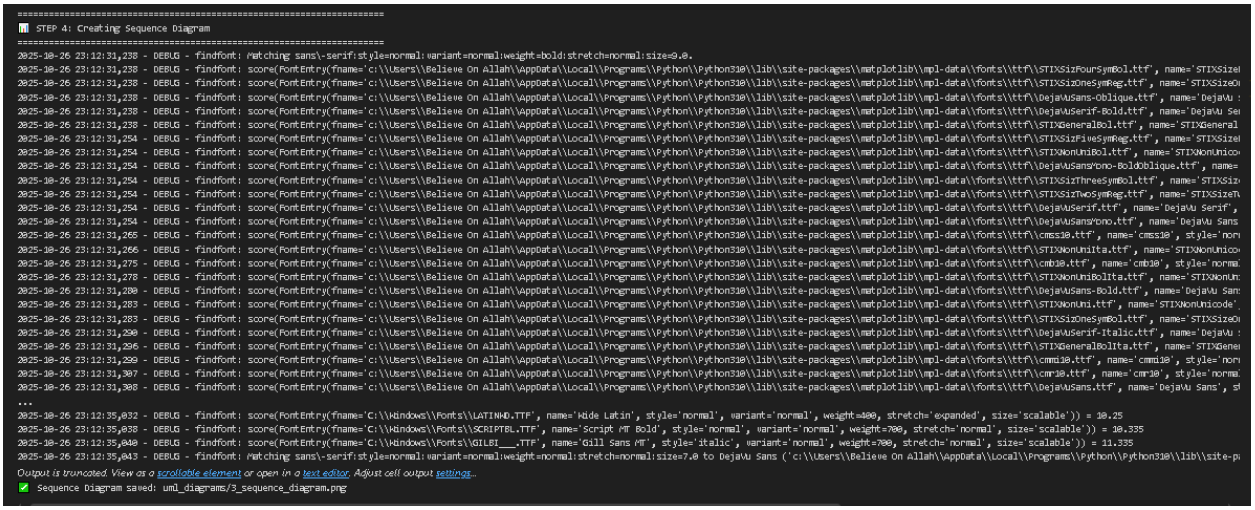




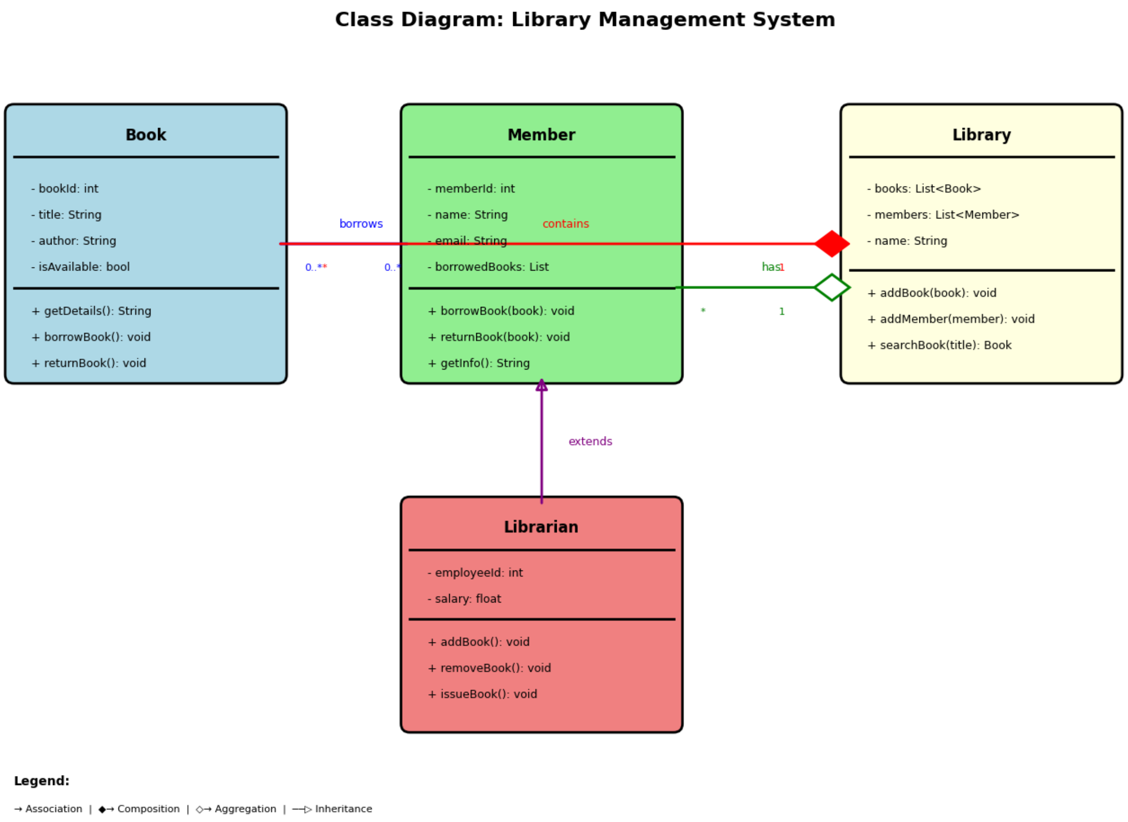
```
2025-10-26 23:12:27,326 - DEBUG - findfont: Matching sans\-serif:style=normal:variant=normal:weight=bold:stretch=normal:size=11.0.
2025-10-26 23:12:27,328 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\STIXSizFourSymBol.ttf', name='STIXSize
2025-10-26 23:12:27,332 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\STIXSizOneSymReg.ttf', name='STIXSizeO
2025-10-26 23:12:27,334 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\DejaVuSans-Oblique.ttf', name='DejaVu
2025-10-26 23:12:27,343 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\DejaVuSerif-Bold.ttf', name='DejaVu Se
2025-10-26 23:12:27,345 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\STIXGeneralBol.ttf', name='STIXGeneral
2025-10-26 23:12:27,345 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\STIXSizFiveSymReg.ttf', name='STIXSize
2025-10-26 23:12:27,345 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\STIXNonUniBol.ttf', name='STIXNonUnico

======================================================================
📊 STEP 3: Creating Use Case Diagram
======================================================================
2025-10-26 23:12:27,356 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\DejaVuSansMono-BoldOblique.ttf', name=
2025-10-26 23:12:27,363 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\STIXSizThreeSymBol.ttf', name='STIXSiz
2025-10-26 23:12:27,366 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\STIXSizTwoSymReg.ttf', name='STIXSizeTw
2025-10-26 23:12:27,372 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\DejaVuSerif.ttf', name='DejaVu Serif',
2025-10-26 23:12:27,379 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\DejaVuSansMono.ttf', name='DejaVu Sans
2025-10-26 23:12:27,381 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\cmss10.ttf', name='cmss10', style='nor
2025-10-26 23:12:27,384 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\STIXNonUniIta.ttf', name='STIXNonUnico
2025-10-26 23:12:27,387 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\cmb10.ttf', name='cmb10', style='norma
2025-10-26 23:12:27,389 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\STIXNonUniBolIta.ttf', name='STIXNonUn
2025-10-26 23:12:27,393 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\DejaVuSans-Bold.ttf', name='DejaVu Sans
2025-10-26 23:12:27,396 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\STIXNonUni.ttf', name='STIXNonUnicode'
2025-10-26 23:12:27,399 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\STIXSizOneSymBol.ttf', name='STIXSizeO
2025-10-26 23:12:27,402 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\DejaVuSerif-Italic.ttf', name='DejaVu
2025-10-26 23:12:27,405 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\STIXGeneralBolIta.ttf', name='STIXGener
2025-10-26 23:12:27,409 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\cmmi10.ttf', name='cmmi10', style='nor
2025-10-26 23:12:27,412 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\cmr10.ttf', name='cmr10', style='norma
2025-10-26 23:12:27,412 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\DejaVuSans.ttf', name='DejaVu Sans', s
2025-10-26 23:12:27,412 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\DejaVuSans-BoldOblique.ttf', name='Deja
2025-10-26 23:12:27,412 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\DejaVuSansMono-Bold.ttf', name='DejaVu
2025-10-26 23:12:27,412 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\STIXSizTwoSymBol.ttf', name='STIXSizeTw
2025-10-26 23:12:27,426 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\DejaVuSerif-BoldItalic.ttf', name='Deja
2025-10-26 23:12:27,429 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\STIXGeneral.ttf', name='STIXGeneral', 
2025-10-26 23:12:27,432 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\STIXSizFourSymReg.ttf', name='STIXSize
2025-10-26 23:12:27,435 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\DejaVuSerifDisplay.ttf', name='DejaVu 
2025-10-26 23:12:27,439 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\cmtt10.ttf', name='cmtt10', style='nor
...
2025-10-26 23:12:30,164 - DEBUG - findfont: score(FontEntry(fname='C:\\Windows\\Fonts\\LATINWD.TTF', name='Wide Latin', style='normal', variant='normal', weight=400, stretch='expanded', size='scalable')) = 11.25
2025-10-26 23:12:30,164 - DEBUG - findfont: score(FontEntry(fname='C:\\Windows\\Fonts\\SCRIPTBL.TTF', name='Script MT Bold', style='normal', variant='normal', weight=700, stretch='normal', size='scalable')) = 11.335
2025-10-26 23:12:30,174 - DEBUG - findfont: score(FontEntry(fname='C:\\Windows\\Fonts\\GILBI___.TTF', name='Gill Sans MT', style='italic', variant='normal', weight=700, stretch='normal', size='scalable')) = 10.335
2025-10-26 23:12:30,177 - DEBUG - findfont: Matching sans\-serif:style=italic:variant=normal:weight=normal:stretch=normal:size=8.0 to DejaVu Sans ('c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-pa
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
✅ Use Case Diagram saved: uml_diagrams/2_usecase_diagram.png
```

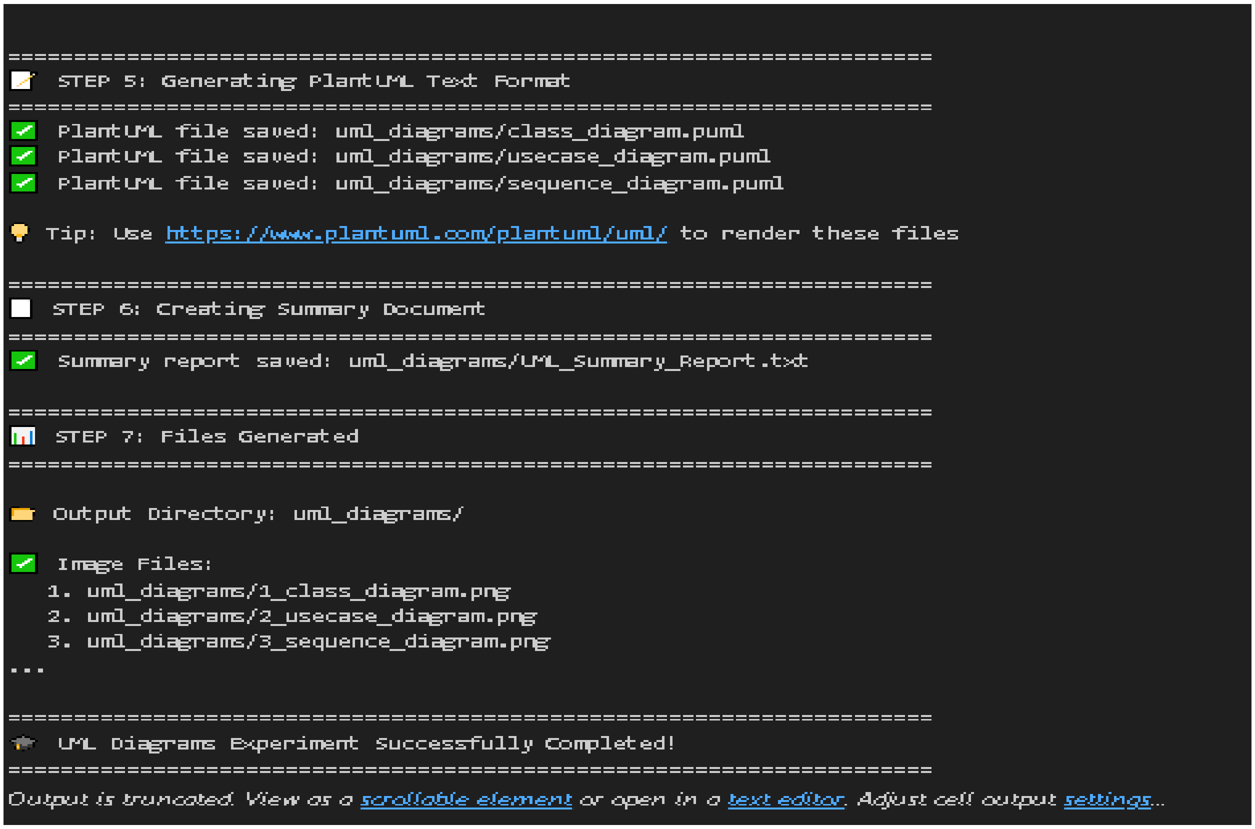


Use Case Diagram: Library Management System



```
==================================================================
📊 STEP 4: Creating Sequence Diagram
==================================================================
2025-10-26 23:12:31,238 - DEBUG - findfont: Matching sans\-serif:style=normal:variant=normal:weight=bold:stretch=normal:size=9.0.
2025-10-26 23:12:31,238 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\STIXSizFourSymBol.ttf', name='STIXSize
2025-10-26 23:12:31,238 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\STIXSizOneSymReg.ttf', name='STIXSizeO
2025-10-26 23:12:31,238 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\DejaVuSans-Oblique.ttf', name='DejaVu :
2025-10-26 23:12:31,238 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\DejaVuSerif-Bold.ttf', name='DejaVu Se
2025-10-26 23:12:31,238 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\STIXGeneralBol.ttf', name='STIXGeneral
2025-10-26 23:12:31,254 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\STIXSizFiveSymReg.ttf', name='STIXSize
2025-10-26 23:12:31,254 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\STIXNonUniBol.ttf', name='STIXNonUnico
2025-10-26 23:12:31,254 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\DejaVuSansMono-BoldOblique.ttf', name=
2025-10-26 23:12:31,254 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\STIXSizThreeSymBol.ttf', name='STIXSiz
2025-10-26 23:12:31,254 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\STIXSizTwoSymReg.ttf', name='STIXSizeT
2025-10-26 23:12:31,254 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\DejaVuSerif.ttf', name='DejaVu Serif',
2025-10-26 23:12:31,254 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\DejaVuSansMono.ttf', name='DejaVu Sans
2025-10-26 23:12:31,265 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\cmss10.ttf', name='cmss10', style='nor
2025-10-26 23:12:31,266 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\STIXNonUniIta.ttf', name='STIXNonUnico
2025-10-26 23:12:31,275 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\cmb10.ttf', name='cmb10', style='norma
2025-10-26 23:12:31,278 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\STIXNonUniBolIta.ttf', name='STIXNonUn
2025-10-26 23:12:31,280 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\DejaVuSans-Bold.ttf', name='DejaVu San
2025-10-26 23:12:31,283 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\STIXNonUni.ttf', name='STIXNonUnicode'
2025-10-26 23:12:31,283 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\STIXSizOneSymBol.ttf', name='STIXSizeO
2025-10-26 23:12:31,290 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\DejaVuSerif-Italic.ttf', name='DejaVu :
2025-10-26 23:12:31,296 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\STIXGeneralBolIta.ttf', name='STIXGene
2025-10-26 23:12:31,299 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\cmmi10.ttf', name='cmmi10', style='nor
2025-10-26 23:12:31,307 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\cmr10.ttf', name='cmr10', style='norma
2025-10-26 23:12:31,308 - DEBUG - findfont: score(FontEntry(fname='c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\matplotlib\\mpl-data\\fonts\\ttf\\DejaVuSans.ttf', name='DejaVu Sans', s
...
2025-10-26 23:12:35,032 - DEBUG - findfont: score(FontEntry(fname='C:\\Windows\\Fonts\\LATINWD.TTF', name='Wide Latin', style='normal', variant='normal', weight=400, stretch='expanded', size='scalable')) = 10.25
2025-10-26 23:12:35,038 - DEBUG - findfont: score(FontEntry(fname='C:\\Windows\\Fonts\\SCRIPTBL.TTF', name='Script MT Bold', style='normal', variant='normal', weight=700, stretch='normal', size='scalable')) = 10.335
2025-10-26 23:12:35,040 - DEBUG - findfont: score(FontEntry(fname='C:\\Windows\\Fonts\\GILBI___.TTF', name='Gill Sans MT', style='italic', variant='normal', weight=700, stretch='normal', size='scalable')) = 11.335
2025-10-26 23:12:35,043 - DEBUG - findfont: Matching sans\-serif:style=normal:variant=normal:weight=normal:stretch=normal:size=7.0 to DejaVu Sans ('c:\\Users\\Believe On Allah\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-pa
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
✅ Sequence Diagram saved: uml_diagrams/3_sequence_diagram.png
```





```
======================================================================
📝 STEP 5: Generating PlantUML Text Format
======================================================================
✅ PlantUML file saved: uml_diagrams/class_diagram.puml
✅ PlantUML file saved: uml_diagrams/usecase_diagram.puml
✅ PlantUML file saved: uml_diagrams/sequence_diagram.puml

💡 Tip: Use https://www.plantuml.com/plantuml/uml/ to render these files

======================================================================
□ STEP 6: Creating Summary Document
======================================================================
✅ Summary report saved: uml_diagrams/UML_Summary_Report.txt

======================================================================
📊 STEP 7: Files Generated
======================================================================

📁 Output Directory: uml_diagrams/

✅ Image Files:
   1. uml_diagrams/1_class_diagram.png
   2. uml_diagrams/2_usecase_diagram.png
   3. uml_diagrams/3_sequence_diagram.png
...

======================================================================
🎓 UML Diagrams Experiment Successfully Completed!
======================================================================
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

9.2 Explainable:

UML (Unified Modeling Language) is a standardized visual modeling language used to design, document, and visualize software systems through diagrams. It provides a set of graphic notation techniques to create abstract models of systems, showing relationships between classes, user interactions, and system behavior. UML diagrams help developers plan system architecture, communicate design decisions, and understand complex systems before coding. Common UML diagrams include Class Diagrams (showing classes and relationships), Use Case Diagrams (showing user interactions), and Sequence Diagrams (showing object interactions over time).

9.3 Key Points

9.3.1. Class Diagram

- Shows system classes, attributes, methods, and relationships
- Displays inheritance, association, aggregation, and composition
- Blueprint for object-oriented programming structure

9.3.2. Use Case Diagram

- Represents user interactions with the system
- Shows actors (users), use cases (actions), and relationships
- Helps identify system requirements and functionalities

9.3.3. Sequence Diagram

- Illustrates how objects interact in time sequence
- Shows message flow between objects chronologically
- Useful for understanding complex operations and workflows