**Lab test -2**

**Name:**Ramakrishna Vemula

**Ht.no:** 2403a54037(CSE-AIML\_15)

**Question 1: AI-Assisted Unit Test Generation**

**Task 1:**

Provide a Python function (e.g., calculate\_area(radius)) to the AI and ask it to generate unit

tests using unittest or pytest.

**Prompt**: Write a Python module with multiple functions (e.g., arithmetic operations) without comments or docstrings.

**Code**:

def add(a, b):

return a + b

def subtract(a, b):

return a - b

def multiply(a, b):

return a \* b

def divide(a, b):

if b == 0:

raise ValueError("Cannot divide by zero")

return a / b

**output:**

Addition: 15

Subtraction: 5

Multiplication: 50

Division: 2.0

**Obseravation:**

Each function in the module performs one arithmetic operation (add, subtract, multiply, divide).  
Division raises an error if the denominator is zero.

**Task-2:**

**Prompt:** generate detailed docstrings and inline comments following Google or NumPy documentation style.

**Code:**

from typing import Union

def add(a: Union[int, float], b: Union[int, float]) -> Union[int, float]:

"""Adds two numbers together.

Args:

a: The first number.

b: The second number.

Returns:

The sum of the two input numbers.

"""

return a + b

def subtract(a: Union[int, float], b: Union[int, float]) -> Union[int, float]:

"""Subtracts the second number from the first.

Args:

a: The number to be subtracted from.

b: The number to subtract.

Returns:

The difference between the two input numbers.

"""

return a - b

def multiply(a: Union[int, float], b: Union[int, float]) -> Union[int, float]:

"""Multiplies two numbers together.

Args:

a: The first number (multiplicand).

b: The second number (multiplier).

Returns:

The product of the two input numbers.

"""

return a \* b

def divide(a: Union[int, float], b: Union[int, float]) -> float:

"""Divides the first number by the second.

Args:

a: The numerator.

b: The denominator.

Returns:

The result of the division.

Raises:

ValueError: If the denominator `b` is zero.

"""

# Division by zero is mathematically undefined and will cause an error.

if b == 0:

raise ValueError("Cannot divide by zero")

return a / b

**Output:**

Addition: 15

Subtraction: 5

Multiplication: 50

Division: 2.0

**Observation:**

The program performs basic arithmetic operations with clear AI-generated docstrings and type hints for better readability

**Question 2: Project Documentation and Summarization**

**Task 1:**

Provide the AI with a small multi-file Python project (e.g., student management system).

Ask it to generate a README.md summarizing the project purpose, inputs, outputs, and

usage.

**Prompt**: with a small multi-file Python project (e.g., student management system). Ask it to generate a README.md summarizing the project purpose, inputs, outputs, and usage.

**Code:**

class StudentDatabase:

def \_\_init\_\_(self):

self.students = []

def add\_student(self, student):

self.students.append(student)

def remove\_student(self, student\_id):

self.students = [s for s in self.students if s.student\_id != student\_id]

def get\_all\_students(self):

return [s.get\_details() for s in self.students]

#main py:

from student import Student

from database import StudentDatabase

def main():

db = StudentDatabase()

s1 = Student(1, "Ravi", 20, "A")

s2 = Student(2, "Priya", 21, "B")

db.add\_student(s1)

db.add\_student(s2)

print("All Students:")

for details in db.get\_all\_students():

print(details)

db.remove\_student(1)

print("\nAfter Removing Student ID 1:")

for details in db.get\_all\_students():

print(details)

if \_\_name\_\_ == "\_\_main\_\_":

main()

**Output:**

All Students:

ID: 1, Name: Ravi, Age: 20, Grade: A

ID: 2, Name: Priya, Age: 21, Grade: B

After Removing Student ID 1:

ID: 2, Name: Priya, Age: 21, Grade: B  
**observation:**

The program successfully adds student records, displays them, and removes a student based on ID.  
It shows correct use of classes, objects, and list operations in Python.

**Task-2:**

**Prompt:**

**include license, dependencies, and setup instructions. Compare the AI’s first and improved documentation**

**Code:**

class Student:

def \_\_init\_\_(self, student\_id, name, age, grade):

self.student\_id = student\_id

self.name = name

self.age = age

self.grade = grade

def get\_details(self):

return f"ID: {self.student\_id}, Name: {self.name}, Age: {self.age}, Grade: {self.grade}"

from student import Student

from database import StudentDatabase

def main():

db = StudentDatabase()

s1 = Student(1, "Ravi", 20, "A")

s2 = Student(2, "Priya", 21, "B")

db.add\_student(s1)

db.add\_student(s2)

print("All Students:")

for details in db.get\_all\_students():

print(details)

db.remove\_student(1)

print("\nAfter Removing Student ID 1:")

for details in db.get\_all\_students():

print(details)

if \_\_name\_\_ == "\_\_main\_\_":

main()

**Output:**

All Students:

ID: 1, Name: Ravi, Age: 20, Grade: A

ID: 2, Name: Priya, Age: 21, Grade: B

After Removing Student ID 1:

ID: 2, Name: Priya, Age: 21, Grade: B

**Observation:**

The Student Management System demonstrates OOP and modular design effectively, and the improved documentation makes it easy to set up, use, and extend.