**Lab Exercise 4- Developing Features Faster with GitHub Copilot Chat**

**Objective:**

Use **GitHub Copilot Chat** to generate, debug, refactor, and optimize code efficiently.

**Prerequisites:**

* **GitHub Copilot Chat** enabled in **VS Code**.
* Basic knowledge of programming (Python/JavaScript).

**Exercise Steps**

**1. Using Copilot Chat for Code Generation**

**Task: Create a Function from a Prompt**

1. Open a Python file.
2. In **Copilot Chat**, ask:

Write a function to check if a string is a palindrome.

1. Accept the generated function and test it.

**Expected Outcome:**

Copilot will generate a function that correctly checks for palindromes.

**2. Automating Boilerplate Code**

**Task: Generate a Flask API**

1. In **Copilot Chat**, ask:

Generate a basic Flask API with a single GET endpoint.

1. Copy and run the generated code.

**Expected Outcome:**

Copilot will generate a simple API with a working endpoint.

**3. Debugging with Copilot Chat**

**Task: Fix Buggy Code**

1. Copy and paste this buggy code into **Copilot Chat**:

def divide\_numbers(a, b):

return a / b

print(divide\_numbers(10, 0)) # This will cause an error

1. Ask Copilot:

Why is this code not working?

1. Accept the suggested fix.

**Expected Outcome:**

Copilot will explain the **ZeroDivisionError** and suggest handling it using try-except:

def divide\_numbers(a, b):

try:

return a / b

except ZeroDivisionError:

return "Error: Division by zero is not allowed"

**4. Refactoring Code for Efficiency**

**Task: Optimize a Function**

1. Provide Copilot Chat with this inefficient function:

def find\_max(numbers):

max\_num = numbers[0]

for num in numbers:

if num > max\_num:

max\_num = num

return max\_num

1. Ask:

How can I make this function more efficient?

1. Apply Copilot’s optimized version.

**Expected Outcome:**

Copilot will suggest using Python’s built-in max() function:

def find\_max(numbers):

return max(numbers)

**5. Writing Unit Tests with Copilot Chat**

**Task: Generate Unit Tests**

1. Provide this function to Copilot Chat:

def add(a, b):

return a + b

1. Ask:

Write unit tests for this function using pytest.

1. Copy the generated test cases and run them.

**Expected Outcome:**

Copilot will generate structured test cases like this:

import pytest

from mymodule import add

def test\_add():

assert add(2, 3) == 5

assert add(-1, 1) == 0

assert add(0, 0) == 0

**6. Generating Documentation & Comments**

**Task: Add Docstrings to a Function**

1. Paste this function into Copilot Chat:

def multiply(a, b):

return a \* b

1. Ask:

Add docstrings to this function.

1. Copy the improved function.

**Expected Outcome:**

Copilot will generate proper documentation:

def multiply(a, b):

"""

Multiply two numbers.

Args:

a (int or float): The first number.

b (int or float): The second number.

Returns:

int or float: The product of a and b.

"""

return a \* b

**Conclusion**

* **Generate code faster** with natural language prompts.
* **Automate repetitive tasks** like setting up APIs and writing tests.
* **Use Copilot Chat for debugging** and fixing errors efficiently.
* **Refactor and document code** with minimal effort.

By following this exercise, you'll **speed up feature development** and improve coding efficiency with **GitHub Copilot Chat**!