AI Task Report

Task 1: AI-Powered Code Completion

Tool: GitHub Copilot

Task: Write a Python function to sort a list of dictionaries by a specific key, compare the AI-suggested code with a manual implementation, and document which version is more efficient.

Manual Implementation:

```
def sort_by_key(data, key):
    return sorted(data, key=lambda x: x[key])
```

AI-Suggested Implementation (GitHub Copilot Style):

```
def sort_by_key(data, key):
    return sorted(data, key=lambda item: item.get(key, 0))
```

Analysis (200 Words):

Both implementations effectively sort a list of dictionaries by a specified key. However, the AI-generated version introduces a defensive programming approach by using item.get(key, 0), which prevents runtime errors when the key is missing in some dictionaries. This makes the AI-generated code more robust for real-world scenarios where data may be inconsistent or partially structured.

In terms of performance, both implementations use Python's built-in sorted() function, which operates in O(n log n) time, making their algorithmic efficiency identical. The difference lies in safety, not speed.

The manual version is slightly cleaner and assumes well-structured input. It's ideal for controlled environments, such as educational exercises or well-validated datasets. The AI-suggested version, on the other hand, anticipates edge cases - a hallmark of production-ready code. This highlights one advantage of using tools like GitHub Copilot: they often provide safe, generalized templates with minimal input, accelerating development and reducing bugs.

Overall, while both versions achieve the same result, the AI's code offers better fault tolerance, making it more efficient in terms of robustness rather than raw performance.

Task 2: Automated Testing with AI

Framework: Testim.io

Task: Automate a test case for a login page (valid/invalid credentials), run the test and capture results, explain how AI improves test coverage compared to manual testing.

Test Cases

Valid Login Test:

1. Navigate to login page (https://example.com/login)
2. Enter 'testuser@example.com' in the email field
3. Enter 'Test1234!' in the password field
4. Click 'Login' button
5. Assert that Dashboard or 'Welcome' appears

Invalid Login Test:

1. Navigate to login page
2. Enter 'wrong@example.com' in the email field

3. Enter 'WrongPass!' in the password field

4. Click 'Login' button

5. Assert that 'Invalid credentials' error message appears

Test Results:

[Insert Screenshot of Test Result Here]

Summary (150 Words):

AI-enhanced test automation tools like Testim.io significantly improve test coverage and efficiency compared to manual testing. With features like smart locators and auto-healing scripts, these tools adapt to UI changes without needing manual updates. For instance, when testing a login page, AI can identify form fields and buttons based on context rather than static IDs, allowing tests to remain valid even if element positions or styles change. Additionally, AI tools can generate test suggestions based on user flows, covering edge cases developers may overlook. This reduces human error, increases accuracy, and saves time - especially in continuous integration environments. Compared to traditional scripting or manual QA, AI-driven testing provides faster execution, better error detection, and broader UI resilience.