Asking ai to refine and find problems

Question 1. Refine your logic or Word Code:

Prompt :

- Hey co-pilot. I have an assessment for intro to information technology. It's asking me - In this part of the assignment, you will explore how Artificial Intelligence (AI) can assist in solving problems, refining logic, and enhancing your assignment's documentation. You are encouraged to use an AI agent such as Microsoft Copilot to support your work. Use Copilot to assist with at least two or more of the following: i picked number 1 - Refine your logic or Word Code: Ask Copilot to review your implementation step and suggest improvements or identify potential issues.

Outcome:

Thanks for sharing your Word Code, Liam! You've laid out a solid logic sequence for managing a train crossing system—great job. Let's refine it a bit to improve clarity, reduce redundancy, and ensure smooth operation under all conditions. Here's my feedback and suggestions:

🔍 Review & Refinement Suggestions

✅ Strengths:

- Clear structure: Steps are logically ordered and easy to follow.
- Safety-first approach: You've prioritized emergencies and sensor faults, which is excellent.
- Realistic timing: Including wait times (e.g., 5s, 1min) shows practical thinking.

🛠️ Suggested Improvements:

1. Step 2 – Monitoring Inputs Consider breaking this into sub-steps or using a loop structure to clarify that monitoring is continuous and parallel:
    - Suggestion: *"Continuously monitor all inputs using a loop or event-driven system to detect changes in real time."*
2. Step 3 – Emergency Handling You could specify what kind of warning is activated (lights, bells, alerts to operator):
    - Suggestion: *"If sensor fault or emergency → activate fail-safe: lower gates, turn on warning lights/bells, and notify operator."*
3. Step 6 – Car Stuck on Tracks This is a critical step. You might want to add a retry mechanism or escalation protocol:
    - Suggestion: *"If car detected on track → maintain gates down, warnings on, alert operator, and initiate emergency train stop protocol."*
4. Step 7 – Rechecking Train Sensor Instead of waiting 5s and rechecking, consider a loop with a timeout:

  - Suggestion: *“While the train is approaching, continue monitoring every 5s for up to 1 minute.”*
5. Step 8 & 9 – Train Passing Logic You could combine these into a conditional wait:
   - Suggestion: *“Wait until the train has passed (max 1 min), then confirm track clearance via crossing sensor.”*
6. Step 10 – Gate Reset Consider adding a check to ensure no vehicle is still on the crossing before raising gates:
   - Suggestion: *“If track and crossing are clear → raise gates, wait 5s, then deactivate warnings.”*

Question 2. Improve documentation: Ask Copilot to help you write a professional README.md file or summarize your project for presentation.

Prompt : for the second question the assessment is asking me to ask you to - Improve documentation: Ask Copilot to help you write a professional README.md file or summarize your project for presentation.

Outcome:

# 🚦 Smart Train Crossing System

## 📌 Project Overview

This project presents a safe, reliable, and cost-effective railway crossing system designed to protect road users and train passengers. It uses state-machine logic to manage gate operations, warning signals, and emergency protocols based on real-time sensor inputs.

## 🎯 Objectives

- Detect train approach and track occupancy

- Lower gates and activate warnings when necessary

- Identify vehicles stuck on tracks and alert operators

- Ensure gates only raise when the crossing is clear

- Default to fail-safe mode during faults or emergencies

## 🧠 Logic Approach

Chosen Method: **State-Machine Logic**

The system transitions between defined states such as:

- Idle

- Warning

- Lowering

- Closed

- Emergency

- Raising

Each state has clear rules for transitioning based on sensor inputs and elapsed time, ensuring robust handling of faults and edge cases.

## 🔍 Inputs & Outputs

Inputs:

- Train approach sensor

- Vehicle presence sensor

- Gate position sensor

- Emergency stop signal

Outputs:

- Gate motor control

- Warning lights and bells

- Operator alert system

## ⚙️ Word Code Summary

1. Start system

2. Continuously monitor all inputs

3. If emergency or fault → activate fail-safe

4. If train approaching → lower gates, activate warnings

5. If car stuck → keep gates down, alert operator

6. If clear → wait, confirm train passage

7. Once train has passed → raise gates, deactivate warnings

8. Repeat cycle

## 🧪 Alternatives Considered

- **Rule-Based Logic**: Simple and intuitive, but less reliable under complex conditions

- **State-Machine Logic**: More robust, scalable, and fault-tolerant

## ✅ Final Decision

State-Machine Logic was chosen for its reliability, clear state transitions, and superior handling of emergencies. While slightly more complex, it prioritizes human safety over cost.

## 👤 Author

Liam – Intro to Information Technology

Documentation and logic refined with Microsoft Copilot 🤖

Reflection:

To refine my train crossing monitoring system, I used Microsoft Copilot to assist with logic review and documentation. I began by asking Copilot to review my Word Code: *“Can you check my logic and suggest improvements?”* and provided it with my complete word code. Copilot responded with targeted suggestions, such as implementing loop structures for continuous monitoring, clarifying emergency protocols, and optimizing wait conditions using conditional logic.

These insights helped me improve the clarity and fault tolerance of my system. For example, Copilot recommended adding a retry mechanism for obstruction detection and refining the train passage logic to avoid unnecessary delays. It also highlighted the importance of explicitly defining system states and transitions, which reinforced my decision to use state-machine logic over rule-based alternatives.

In the second part of the task, I asked Copilot to help me write a professional README.md and a project summary. The result was a clear, structured document that captured my system’s purpose, logic, and design decisions. Copilot’s support enhanced both the technical robustness and presentation quality of my final solution, deepening my understanding of how AI can assist with problem solving and documentation.