1. **Approach: Clearly explain how you tackled the problem and the reasoning behind your choices:-**

I accessed the Docker container and located the relevant .cpp files, which I opened in VS Code to debug, correct, and save. The corrected code was then integrated into the Core Flight System (cFS), with outputs linked to the provided telemetry Python script to ensure accurate real-time data handling. Controlled faults were introduced into the system to simulate anomalies, and the simulation was run to demonstrate the system breaching the predefined threshold of 1 km within the allowed time of 360 seconds. A video recording of the demonstration was captured to validate system behavior under fault conditions and provide evidence of successful integration and fault response**.**

1. **Resources used:**

 ChatGPT

 Perplexity

1. **If the target output was achieved:**

**◦ Provide the code/circuit files. –** given in a different doc file (corrected source file)

**◦ Document the exact commands or steps to reproduce.** – given is different doc file (CubeSat Onboard Computer Integration Report)

**◦ Attach clear simulation screenshots/plots. –** video provided in form

1. **Failures & fixes: Briefly describe any issues encountered and how you diagnosed and resolved them:**

Initially, I struggled to locate the .cpp files, not realizing they were inside the Docker container. Once accessed, the next challenge was integrating the corrected code with cFS, which required guidance from AI platforms to successfully complete. The final stage involved running the simulator and recording the system behavior, which initially ran for over an hour without achieving the intended fault demonstration. With assistance from Harshavardhan K, I realized that fault injection had been overlooked. After updating the telemetry Python script to introduce controlled errors, I reran the simulation, recorded the corrected video, and submitted the final results.