API Issues We Need to Address (with Line Numbers)

1. Getting Cached Data Quickly
    - **The loop in pirail_web.py that processes cached sensor data starts at line 380.**
    - **Problem:** Right now, the loop pulls in data object by object, which could seriously slow things down when dealing with big datasets, especially if we ever want to scale this.
    - **Why It's an Issue:** The performance could get bad really fast. If the API can't handle big data efficiently, users might experience major delays, and the server could get overwhelmed. We need to optimize how we process and serve this data to keep things smooth.

2. Setting Up Real-Time Streaming
    - **The do_GET() method in web_server.py starts at line 22.**
    - **Problem:** This method is responsible for handling incoming GET requests, including streaming data to clients in real-time. But if a client disconnects, it throws a `BrokenPipeError`. The current setup just logs the error and keeps going, but we need a more graceful way to handle these situations.
    - **Why It's an Issue:** If we don't handle streaming well, the server could slow down or crash, especially when lots of clients are connected or if there are frequent disconnects. We need to make sure the server stays stable, even with heavy traffic.

3. Optimizing Data Filtering
    - **The thin_acc_z function in pirail_web.py is from line 89.**
    - **Problem:** This function filters and thins out sensor data to reduce noise, but it could be too slow when processing large datasets. If the calculations aren't efficient, we could end up with a bottleneck.
    - **Why It's an Issue:** Slow data processing means the API might not be able to keep up with real-time data demands, which would be a problem for any

applications relying on fast, accurate updates. We need to streamline these data operations to make everything run faster.

4. Improving Error Handling
    - **The web_server() function in util.py starts at line 14.**
    - **Problem:** The error handling in this function is pretty basic. It logs warnings for unexpected errors and keeps running, but this isn't enough to protect the server from more serious problems.
    - **Why It's an Issue:** If something major goes wrong (like a system-level failure or a really bad request), the server might not recover gracefully. This could lead to crashes or data loss. We need to add more strong error handling to make sure the server stays reliable.