**Experiment Report - 65 - test2\_demoCode**

1. **Summary Table of Errors Found**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Error ID | Line Number | Error Type | Self-Detected? | Peer 1 Found? | Peer 2 Found? |
| E01 | line 5 | Logic | √ | × | × |
| E02 | line 20 | Semantic | √ | √ | √ |
| E03 | line 37 | Semantic | √ | × | √ |
| E04 | line 50 | Syntax | × | × | × |

Additional Errors Found by Self: 0

Self-Review Detection Rate: 75%

Peer 1 Detection Rate: 25%

Peer 2 Detection Rate: 50%

1. **Source Code**
2. #include <string>
3. #include <fstream>
4. VideoRecorder::VideoRecorder(TimeManager& timeManager, GPSModule& gpsModule, const VideoConfig& config)
5. : timeManager(timeManager), gpsModule(gpsModule), videoConfig(config), recording(true) {}
6. VideoRecorder::~VideoRecorder() {
7. if (recording) {
8. stopRecording();
9. }
10. }
11. void VideoRecorder::startRecording() {
12. if (recording) {
13. std::cerr << "Error: Recording is already in progress." << std::endl;
14. return;
15. }
16. std::string timestamp = timeManager.getCurrentTimestamp();
17. currentVideoFilePath = "video\_" + timestamp + "mp4";
18. videoFileStream.open(currentVideoFilePath, std::ios::binary);
19. if (!videoFileStream.is\_open()) {
20. std::cerr << "Error: Failed to open video file for recording." << std::endl;
21. return;
22. }
23. writeMetadataHeader();
24. recording = true;
25. std::cout << "Recording started. Saving to: " << currentVideoFilePath << std::endl;
26. for (int i = 0; i < 150; ++i) {
27. if (!recording) break;
28. embedTimestampAndGPS();
29. videoFileStream << "VideoFrameData";
30. std::this\_thread::sleep\_for(std::chrono::milliseconds(30)); // Simulate 30 fps (1000/30 ≈ 33 ms)
31. }
32. }
33. void VideoRecorder::stopRecording() {
34. if (!recording) {
35. std::cerr << "Error: No ongoing recording to stop." << std::endl;
36. return;
37. }
38. videoFileStream.close();
39. recording = false;
40. std::cout << "Recording stopped. File saved at: " << currentVideoFilePath << std:endl;
41. }
42. bool VideoRecorder::isRecording() const {
43. return recording;
44. }
45. std::string VideoRecorder::getCurrentVideoFilePath() const {
46. return recording ? currentVideoFilePath : "";
47. }