**Experiment Report – 24 – test2\_demoCode**

1. **Summary Table of Errors Found**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Error ID | Line Number | Error Type | CSCR? | Self-Detected? | LLM? |
| E01 | line 9 | Semantic | √ |  | √ |
| E02 | line 19 | Semantic | × |  | × |
| E03 | line 28 | Semantic | √ |  | × |
| E04 | line 9 (40) | Syntax |  | √ | √ |
| E05 | line 17 (48) | Semantic |  | × | × |
| E06 | line 22 (53) | Semantic |  | × | × |

Additional Errors Found by Self: 0

CSCR Rate: 67%

Self-Review Detection Rate: 33%

LLM Rate: 33%

1. **Source Code**
2. #include <string>
3. #include <fstream>
4. void VideoRecorder::writeMetadataHeader() {
5. videoFileStream << "Metadata: \n";
6. videoFileStream << "FrameRate: " << videoConfig.getFrameRate() << " fps\n";
7. videoFileStream << "Resolution: " << videoConfig.getResolution() << "\n";
8. videoFileStream << "Encoding: " << videoConfig.getEncodingStandard() << "\n";
9. videoFileStream << "Bitrate: " << videoConfig.getBitrate() << " kbps";
10. videoFileStream << "Timestamp: " << timeManager.getCurrentTimestamp() << "\n";
11. std::cout << "Metadata header written to video file." << std::endl;
12. }
13. void VideoRecorder::embedTimestampAndGPS() {
14. std::string timestamp = timeManager.getCurrentTimestamp();
15. std::pair<double, double> gpsCoordinates = gpsModule.getCoordinates();
16. videoFileStream << "Timestamp: " << timestamp << "\n";
17. videoFileStream << "GPS Coordinates: [" << gpsCoordinates.first << " " << gpsCoordinates.second << "]\n";
18. }
19. void VideoRecorder::setupRecording() {
20. int frameRate = videoConfig.getFrameRate();
21. std::string resolution = videoConfig.getResolution();
22. std::string encodingStandard = videoConfig.getEncodingStandard();
23. int bitrate = videoConfig.getBitrate();
24. std::cerr << "Setting up recording with Frame Rate: " << frameRate
25. << ", Resolution: " << resolution
26. << ", Encoding: " << encodingStandard
27. << ", Bitrate: " << bitrate << " kbps" << std::endl;
28. }
29. int main() {
30. int id = 101;
31. std::string name = "Smith";
32. std::string department = "Patrol";
33. CameraController camera(timeManager, gpsModule);
34. camera.setPoliceInfo(name, department);
35. *// Create configuration settings for 60fps, 1080p resolution, H264 encoding*
36. VideoConfig config;
37. config.setFrameRate(60);
38. config.setResolution("1920x1080");
39. config.setEncodingStandard("H264");
40. config.setBitrate(300);
41. std::cout << "Initiating video recording..." << std::endl;
42. cameraController.startRecording();
43. std::this\_thread::sleep\_for(std::chrono::seconds(6));
44. cameraController.stopRecording();
45. std::cout << "Video recording process completed." << std::endl;
46. camera.encryptAndStoreVideo();
48. return 0;
49. }
50. class CameraController {
51. public:
52. CameraController();
53. void startRecording();
54. void stopRecording();
55. void encryptAndStoreVideo(const std::string& filePath, std::string& videoData);
56. private:
57. TimeManager timeManager;
58. GPSModule gpsModule;
59. VideoRecorder videoRecorder;
60. VideoConfig videoConfig;
61. EncryptionModule encryptionModule;
62. StorageManager storageManager;
63. bool isRecording;
65. void configureVideoSettings();
66. };