**Experiment Report - 64 - test1\_demoCode**

1. **Summary Table of Errors Found**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Error ID | Line Number | Error Type | Self-Detected? | Peer 1 Found? | Peer 2 Found? |
| E01 | line 11 | Syntax | √ | √ | √ |
| E02 | line 28 | Semantic | × | × | √ |
| E03 | line 36 | Semantic | √ | × | × |
| E04 | line 47 | Semantic | √ | √ | × |

Additional Errors Found by Self: 0

Self-Review Detection Rate: 75%

Peer 1 Detection Rate: 50%

Peer 2 Detection Rate: 50%

1. **Source Code**
2. #include <string>
3. #include <fstream>
4. void CameraController::setPoliceInfo(int id, std::string name, std::string department){
5. PolicemanInfo info(id, name, department);
6. this->policemanInfo = info;
7. }
8. void CameraController::configureVideoSettings() {
9. videoConfig.setFrameRate(30fps); // 30 fps
10. videoConfig.setResolution("1920x1080"); // Full HD
11. videoConfig.setEncodingStandard("H264"); // H.264 encoding
12. videoConfig.setBitrate(2500); // 2500 kbps
13. }
14. void CameraController::startRecording() {
15. configureVideoSettings();
16. videoRecorder = std::make\_unique<VideoRecorder>(timeManager, gpsModule, videoConfig);
17. if (videoRecorder) {
18. std::cout << "Starting video recording..." << std::endl;
19. videoRecorder->startRecording();
20. std::cout << "Recording started successfully." << std::endl;
21. } else {
22. std::cout << "Error: VideoRecorder instance could not be created." << std::endl;
23. }
24. }
25. void CameraController::stopRecording() {
26. if (videoRecorder && videoRecorder->getCurrentVideoFilePath().empty() == false) {
27. std::cout << "Stopping video recording..." << std::endl;
28. videoRecorder->stopRecording();
29. std::cout << "Recording stopped. File saved: " << videoRecorder->getCurrentVideoFilePath() << std::endl;
30. } else {
31. std::cerr << "Error: No active recording to stop." << std::endl;
32. }
33. }
34. void CameraController::encryptAndStoreVideo(const std::string& filePath, std::string& videoData){
35. std::string encryptedData = this->encryptionModule.encrypt(videoData);
36. std::string encryptedFilePath = this->storageManager.generateEncryptedFileName(filePath);
37. this->storageManager.write(videoData);
38. }