AI Task

Overview

Build an AI system that classifies PLC compilation errors and suggests fixes. OTee's customers develop PLC logic programs that go through a multi-stage build pipeline (XML validation → IEC code generation → compilation → C compilation). Engineers spend hours debugging cryptic errors that span multiple stages. Your system should parse error logs, classify errors by severity/stage/complexity, and suggest actionable fixes with code examples.

Code Requirements

1. Error Log Parser

- Parse multi-line, multi-stage error output
- Extract: error type, stage, line numbers, context
- Handle cascading errors (one XML issue → multiple downstream errors)
- See sample files for example error formats

2. Error Classifier

- **Severity**: blocking, warning, info
- **Stage (if applicable)**: xml_validation, code_generation, iec_compilation, c_compilation
- **Fix complexity**: trivial, moderate, complex
- Use LLM with appropriate prompting strategy

3. Fix Suggestion System

- Generate 1-3 actionable suggestions with code snippets (before/after)
- Explain why the error occurred (root cause)
- Assign confidence score (0.0-1.0) to each suggestion
- Handle cases with missing context gracefully

4. HTTP API

```
POST /classify      - Submit error log, get classification + fix suggestions
```

Response time target: < 3 seconds per request

5. Evaluation Framework

- Build synthetic error generator (create variations of the 2 base error types)
- Generate 20-30 test cases with ground truth labels
- Implement automated metrics:
  - Classification accuracy (severity, stage, complexity)
  - Suggestion quality score (can be manual labels for now)
- Output evaluation report showing performance across error types

Technical Requirements

- Python implementation with clear structure
- LLM integration (OpenAI, Anthropic, or open-source models)
- Basic error handling for malformed logs
- Simple test coverage for parser and classifier
- Requirements file with dependencies

Deliverables

- Complete working code with HTTP API

- README with setup, usage, and API documentation
- Brief explanation of architecture decisions (in README)
- Test suite for parser, classifier, and evaluation framework
- Evaluation results showing performance metrics

Estimated time

4-5 hours (within a 3 day period)

You can store your work in a private github repository and add `@radek-otee` to it.

Good luck!