SpecLens – System Specification Document

1. Problem Definition

SpecLens-PML is a data-driven system designed to assist software engineers in assessing the operational risk of
formal specifications written in the PML (Program Modeling Language). The system analyzes PML units (functions and methods annotated with requires/ensures/invariants) and predicts the likelihood that a given specification is unsafe, incomplete, or fragile.

From a machine learning perspective, the problem is formulated as a binary classification task:
given a vector of structural features extracted from a PML unit, predict whether the unit belongs to the class SAFE (0) or RISKY (1).

Key Performance Indicators (KPIs):
- Classification accuracy on labeled datasets
- Recall on RISKY class (safety-oriented metric)
- End-to-end latency for file analysis (< 2s per file)
- Stability of predictions across model versions

2. Data Specification

Data originates from Python source files annotated in PML style. Each PML unit is parsed and transformed into a
feature vector including: number of parameters, requires, ensures, invariants, and lines of code.

The data pipeline performs:
- Static parsing of source files
- Feature extraction
- Label association (SAFE / RISKY)
- CSV dataset generation

Quality constraints:
- Each sample must correspond to exactly one PML unit
- Missing annotations are encoded explicitly
- Datasets are versioned (datasets_v1.csv, datasets_v2.csv, ...)

3. Functional Requirements

FR-01: The system shall parse PML-annotated Python files.
FR-02: The system shall extract structural features from each PML unit.
FR-03: The system shall classify each PML unit using a trained ML model.
FR-04: The system shall output a risk score in the range [0,1].
FR-05: The system shall map the risk score to an operational risk level:
LOW (score < 0.2), MEDIUM (0.2 ≤ score < 0.6), HIGH (score ≥ 0.6).
FR-06: The system shall persist trained models as versioned artifacts.
FR-07: The system shall support continuous retraining when performance degrades.

4. Non-Functional Requirements

NFR-01: The system shall analyze a file in less than 2 seconds.
NFR-02: The system shall be reproducible given the same configuration and dataset.

NFR-03: The system shall maintain traceability between datasets, models, and predictions.
NFR-04: The system shall allow configuration through external YAML files.
NFR-05: The system shall support rollback to previous model versions.

5. Architecture

The architecture follows an end-to-end MLOps pipeline:
- Data ingestion and dataset builder
- Training pipeline (train.py)
- Model registry (versioned artifacts)
- Inference layer (predict.py)
- Continuous Training engine (ct_trigger.py)
- Central configuration (config.yaml)

Predictions are not final decisions but decision-support signals for engineers.

6. Risk Analysis

Main risks:
- Concept drift due to evolving coding practices
- Class imbalance leading to poor recall on RISKY samples
- Overfitting on small datasets
- Misinterpretation of scores

Mitigations:
- Continuous Training loop
- Operational thresholds
- Human-in-the-loop validation
- Versioned rollback strategy