**Cycle 1 Detailed Plan (Java + Spring Boot / Apache)**

**1️⃣ Goals**

* Establish baseline with SonarQube on selected Java projects.
* Identify initial weaknesses: false positives in the **detection engine**, static **prioritization logic**.
* Build **first ML/AI prototype** for filtering false positives and adjusting prioritization.

**2️⃣ Step-by-Step Plan**

**Step 1: Environment Setup**

1. Install **SonarQube Community Edition** (latest version).
2. Install **SonarQube Scanner** for Java projects.
3. Clone selected repositories locally:
   * Spring Boot, Apache Commons Lang, JUnit 5.
4. Ensure you can **run full scans and export results** (CSV/JSON).

**Step 2: Baseline Analysis**

1. Run **SonarQube scans** on all 3 projects.
2. Collect **all issues**: code smells, bugs, vulnerabilities.
3. Focus on common **false positive-prone smells**:
   * Long methods
   * Large classes
   * Duplicate code
   * Complex conditionals
4. Manually **annotate a small subset** (200–500 samples) to mark **true vs false positives**.

📌 Deliverable: **Baseline dataset** with annotations + SonarQube issue metadata.

**Step 3: Initial Weakness Targeting**

* **Detection Engine**: static rules → false positives.
* **Prioritization Logic**: severity static, not context-aware.

**ML Intervention Plan**:

1. **Detection Engine** → classification model for false positives.
2. **Prioritization** → simple weighting adjustment (e.g., recent commits, file churn).

**Step 4: Dataset Preparation**

1. Export SonarQube results (JSON/CSV).
2. Label subset for ML training: true/false positives.
3. Collect metadata for ranking:
   * File churn
   * Commit count
   * Last modified date

**Step 5: ML Prototype**

* **Detection Engine (False Positive Filter)**:
  + Input: rule type, method/class size, complexity metrics
  + Output: true/false positive
  + Algorithm: Random Forest / XGBoost (simple, interpretable)
* **Prioritization Adjustment**:
  + Weight Sonar severity by file churn or commit frequency
  + Produce a ranked list of smells

**Step 6: Evaluation**

* Metrics:
  + **False positive reduction**: Precision, Recall
  + **Ranking improvement**: Spearman correlation with “ground truth” (if available)
* Compare **baseline SonarQube → ML-enhanced results**.

📌 Deliverable: **Prototype v0.1** (Detection + naive prioritization) + evaluation report.

**Step 7: Documentation**

* Environment setup & config steps
* Dataset preparation & labeling methodology
* ML model selection, training, and evaluation
* Results, metrics, and observations
* Notes for **Cycle 2 iteration planning**

**3️⃣ Tools / Frameworks**

* **Language:** Java (Spring Boot + Apache projects)
* **ML / Data Analysis:** Python (pandas, scikit-learn, XGBoost)
* **Version Control:** Git (for commit frequency/churn metrics)
* **Visualization:** Jupyter Notebook

**4️⃣ Expected Output**

1. **Baseline SonarQube report** on selected Java projects
2. **Annotated false positives dataset** (~200–500 issues)
3. **Prototype ML filter + prioritization tweak**
4. **Evaluation report** vs baseline
5. **Cycle 1 documentation** for thesis chapter