**Concise Justification Statement:**

This research leverages **SonarQube** as the baseline tool due to its open-source nature, comprehensive static analysis capabilities, and widespread industry adoption, making it ideal for benchmarking and enhancement. **Java projects**, particularly those using **Spring Boot and Apache frameworks**, are chosen for their mature SonarQube support, diverse and realistic codebases, and manageable complexity, providing an optimal environment to develop and evaluate AI-driven, feedback-enhanced code-smell prioritization and automated ticketing.

**Justification for Tool Selection – Cycle 1**

For the purpose of this research, **SonarQube** has been selected as the baseline tool for code-smell detection and prioritization for several compelling reasons:

1. **Open Source and Extensible** – SonarQube Community Edition is fully open-source, providing complete access to its source code. This allows direct modification and integration of novel approaches, such as feedback-driven prioritization and AI/ML enhancements, without licensing restrictions.
2. **State-of-the-Art and Industry Standard** – SonarQube is widely adopted in the software industry and recognized for its comprehensive static analysis capabilities. Its detection engine, prioritization logic, and metrics like code health and defect density reflect current best practices, making it an ideal benchmark for evaluating improvements.
3. **Mature Ecosystem and Documentation** – The tool has extensive documentation, plugins, and community support, facilitating experimentation and deeper understanding of its strengths and limitations. This ensures a practical and reproducible research environment.

**Justification for Programming Language Selection**

For Cycle 1, the research focuses on **Java projects** built with **Spring Boot and Apache frameworks** due to the following reasons:

1. **Mature SonarQube Support** – Java has the most comprehensive set of detection rules in SonarQube, covering code smells, bugs, and vulnerabilities. This ensures a rich dataset for analysis, prioritization, and ML model training.
2. **Diverse and Realistic Codebases** – Open-source projects such as Spring Boot, Apache Commons Lang, and JUnit 5 offer varying sizes, modularity, and complexity. This diversity provides realistic scenarios for testing detection engines and prioritization logic under conditions similar to professional software development.
3. **Reproducibility and Extendibility** – These projects are widely used and actively maintained, ensuring that results are reproducible and that improvements to detection or prioritization methods can be validated across multiple real-world codebases.
4. **Optimal Balance Between Complexity and Manageability** – Focusing on Java + Spring Boot / Apache projects provides a manageable scope for Cycle 1, allowing meaningful experiments with ML-based enhancements while keeping development and data labeling efforts feasible within the research timeline.