**Software Requirements**

**Specification**

**For**

**AI Code Reviewer**

**Version 1.0 approved**

**Prepared by**

**TIRTHORAJ BHATTACHARYA**

**TANMESH SINGH**

**INDIAN INSTITUTE OF INFORMATION TECHNOLOGY ALLAHABAD**

**04/04/2025**

**Table of Contents**

**Table of Contents ............................................................................................ ii**

1. **Introduction .......................................................................................... 3**
   1. Purpose ............................................................................................. 3
   2. Document Conventions .................................................................... 3
   3. Intended Audience and Reading Suggestion .................................... 3
   4. Product Scope ................................................................................... 4
   5. References ......................................................................................... 4
2. **Overall Description …………………………………………………… 4**

2.1 Product Perspective ………………………………………………… 4

2.2 Product Functions …………………………………………………... 5

2.3 User Classes and Characteristics …………………………………… 5

2.4 Operating Environment ………………………………………...…... 8

2.5 Design and Implementation ………………………………………… 8

2.6 User Documentation ………………………………………………… 8

2.7 Assumptions and Dependencies ……………………………………... 9

1. **External Interface Requirements ………………………………….... 10**
   1. User Interfaces ………………………………………………….…. 10
   2. Hardware Interfaces …………………………………………….…. 10
   3. Software Interfaces ………………………………………………… 11
   4. Communication Interfaces …………………………………………. 11

**1. Introduction**

**1.1 Purpose**

This Software Requirements Specification (SRS) document describes the functional and non-functional requirements for the AI-Powered Code Reviewer system, version 1.0. This document covers the Minimum Viable Product (MVP) phase of the system, which will integrate with GitHub to analyze JavaScript/TypeScript code changes against company standards using AI to detect issues that traditional static analyzers might miss.

**1.2 Document Convention**

The following conventions have been used in this document:

* Requirements are organized by functional areas and are numbered hierarchically.
* "Must" indicates a mandatory requirement.
* "Should" indicates a desirable requirement.
* "May" indicates an optional requirement.
* TBD (To Be Determined) is used for details that will be defined in future versions of this document.

**1.3 Intended Audience and Reading Suggestions**

This document is intended for:

* **Development Team:** To understand what needs to be built
* **QA Team:** To develop test plans and test cases
* **Project Manager:** To plan the implementation and allocate resources
* **Stakeholder:** To ensure the system meets business requirements

**1.4 Product Scope**

The AI-Powered Code Reviewer system is designed to serve as a preliminary checkpoint before developers submit code for formal peer review. It aims to:

* Automate initial code quality checks to ease senior developers' workload.
* Help junior developers receive immediate feedback on their code changes.
* Improve overall code quality and consistency.
* Reduce the number of review iterations required before code acceptance.
* Enforce company coding standards consistently.

The system will integrate with GitHub to analyze pull requests, focusing initially on JavaScript/Type Script code analysis, and will provide feedback directly within GitHub's interface.

**1.5 Reference**

* GitHub API Product Perspective Documentation: <https://docs.github.com/en/rest>
* ESLint Documentation: <https://eslint.org/docs/latest/>
* Hugging Face Models Documentation: <https://huggingface.co/docs/transformers/index>
* JavaScript/TypeScript Code Style Guide: [\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*]
* Node.js Documentation: https://nodejs.org/en/docs/

**2. Overall Description**

**2.1 Product Perspective**

The AI-Powered Code Reviewer is a new, standalone system that integrates with existing GitHub workflows and tooling. It operates as a service that:

1. Listens for GitHub webhook events when pull requests are created or updated
2. Fetches the code changes from GitHub
3. Processes the code through static analysis and AI-based pattern recognition
4. Generates feedback based on the analysis
5. Posts the feedback as comments in the pull request

**2.2 Product Functions**

The major functions of the AI-Powered Code Reviewer include:

* GitHub webhook integration for pull request analysis
* Static code analysis using ESLint with custom rules
* Basic pattern detection against reference code examples
* AI-assisted code quality classification using pre-trained models
* Pull request commenting with inline suggestions and feedback
* Simple web dashboard for configuration and visualization of analysis metrics
* User authentication and authorization for dashboard access

**2.3 User classes and characteristics**

1. **Junior Developers (Primary Users)**

* Frequent use; will receive feedback on their code submissions
* May have limited experience with code reviews
* Need clear, actionable feedback with examples
* Technical proficiency: Moderate to high

**2.** **Senior Developers/Code Reviewers (Secondary Users)**

* Less frequent use; will configure rules and review system effectiveness
* Experienced in code review processes
* Need ability to customize rules and provide feedback on AI suggestions
* Technical proficiency: High

**3. System Administrators (Tertiary Users)**

* Infrequent use; will manage system configuration and monitor health
* Need access to logs, performance metrics, and configuration settings
* Technical proficiency: High



Fig-1: Use Case Diagram for AI Code Reviewer

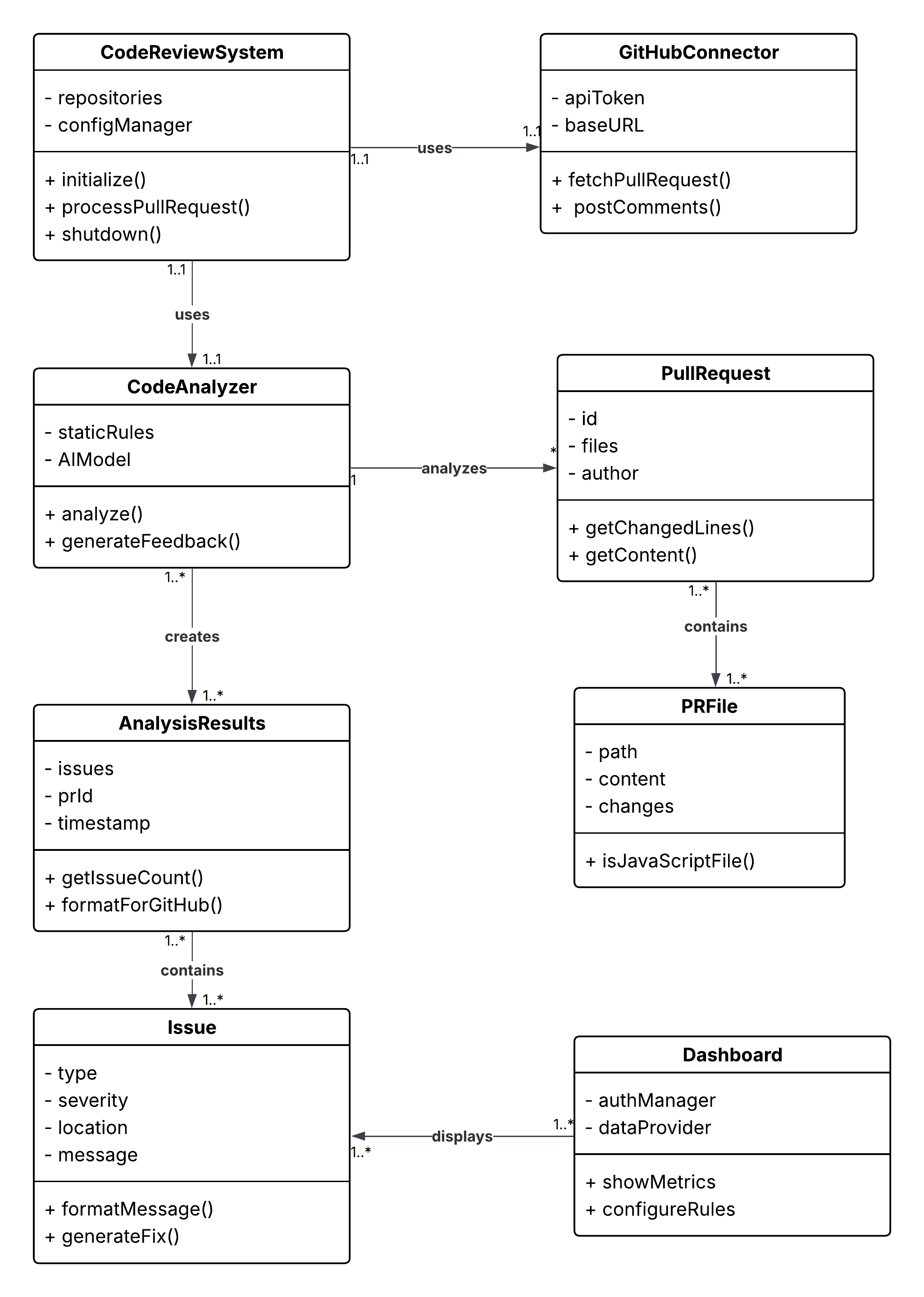


Fig-2: Class Diagram for AI Code Reviewer

**2.4 Operating Environment**

The AI-Powered Code Reviewer will operate in the following environment:

* Deployed as Docker containers on Linux-based servers.
* Backend implemented as Node.js tech-stack.
* Frontend dashboard implemented with React.
* Integrated with GitHub via webhooks and API.
* Minimum Node.js version 18.x required.
* Support for modern web browsers (Chrome, Firefox, Safari, Edge).
* Internet connectivity required for GitHub integration.

**2.5 Design and Implementation Constraints**

The system is subject to the following constraints:

* Must use free and open-source tools to minimize costs.
* Initial focus on JavaScript / TypeScript analysis only.
* Will use pre-trained models rather than custom ML training for the MVP.
* Must comply with GitHub API rate limits.
* Must complete analysis within 5 minutes of PR submission to provide timely feedback.
* Must be containerized for easy deployment.
* Must work within GitHub Actions environment constraints.

**2.6 User Documentation**

The following user documentation will be provided:

* Installation and setup guide for system administrators.
* User manual for developers explaining the feedback system.
* Configuration guide for senior developers to customize rules.
* Dashboard usage guide for visualizing and monitoring metrics.
* API documentation for potential integration with other systems.

**3. External Interface Requirements**

**3.1 User Interfaces**

**3.1.1 GitHub Pull Request Interface**

* The system will post comments directly to GitHub pull requests
* Comments will be formatted with Markdown and include:
  + - * + Issue description
        + Code snippet highlighting the problem
        + Suggested fix (when possible)
        + Severity level (Critical, Major, Minor, Suggestion)
        + Reference to relevant coding standards

**3.2 Hardware Interfaces**

The system does not directly interface with hardware. It will run on standard server infrastructure with:

* Minimum 4GB RAM
* - 2 CPU cores
* - 20GB storage
* - Network connectivity

**3.3 Software Interfaces**

**3.3.1 GitHub API**

* REST API v3
* Authentication via OAuth tokens
* Methods used:
  + - * + Pull Request events webhook
        + - Get PR content
        + - Post PR comments

**3.3.2 ESLint**

* Version 8.x or higher
* - Custom rule configurations
* - JavaScript and TypeScript parsers

**3.3.3 Hugging Face Models API**

* HTTPS for all external communications
* - WebSockets for real-time dashboard updates
* - JSON for data interchange format
* - GitHub webhooks for event notifications
* - RESTful API endpoints for system integration

**3.3.4 Database**

* MongoDB for configuration and historical data storage
* - Connection via standard MongoDB driver
* - Minimum version 4.4

**3.4 Communications Interface**

* - WebSockets for real-time dashboard updates
* - JSON for data interchange format
* - GitHub webhooks for event notifications
* - RESTful API endpoints for system integration