Software Requirements Specification

for

OptiCode

Version 1.0.3 approved

Prepared by

A.Abhinav

B.Ramya Sree

M.Gowthami

S.Siddartha

M.Hindu Priya

KMIT

1st Dec 24

Copyright © 1999 by Karl E. Wiegers. Permission is granted to use, modify, and distribute this document.

Software Requirements Specification for OptiCode Page ii

# Table of Contents

[Table of Contents 2](#_Toc6145)

[Revision History 3](#_Toc6146)

[1. Introduction 1](#_Toc6147)

[1.1 Purpose 1](#_Toc6148)

[1.2 Document Conventions 1](#_Toc6149)

[1.3 Intended Audience and Reading Suggestions 1](#_Toc6150)

[1.4 Product Scope 1](#_Toc6151)

[1.5 References 1](#_Toc6152)

[2. Overall Description 1](#_Toc6153)

[2.1 Product Perspective 1](#_Toc6154)

[2.2 Product Functions 1](#_Toc6155)

[2.3 User Classes and Characteristics 2](#_Toc6156)

[2.4 Operating Environment 2](#_Toc6157)

[2.5 Design and Implementation Constraints 2](#_Toc6158)

[2.6 User Documentation 2](#_Toc6159)

[2.7 Assumptions and Dependencies 2](#_Toc6160)

[3. External Interface Requirements 2](#_Toc6161)

[3.1 User Interfaces 2](#_Toc6162)

[3.2 Hardware Interfaces 2](#_Toc6163)

[3.3 Software Interfaces 3](#_Toc6164)

[3.4 Communications Interfaces 3](#_Toc6165)

[4. System Features 3](#_Toc6166)

[4.1 System Feature 1 3](#_Toc6167)

[4.2 System Feature 2 (and so on) 3](#_Toc6168)

[5. Other Nonfunctional Requirements 4](#_Toc6169)

[5.1 Performance Requirements 4](#_Toc6170)

[5.2 Safety Requirements 4](#_Toc6171)

[5.3 Security Requirements 4](#_Toc6172)

[5.4 Software Quality Attributes 4](#_Toc6173)

[5.5 Business Rules 5](#_Toc6174)

[6. Other Requirements 5](#_Toc6175)

[Appendix A: Glossary 5](#_Toc6176)

[Appendix B: Analysis Models 6](#_Toc6177)

[Appendix C: To Be Determined List 6](#_Toc6178)

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Date | Reason For Changes | Version |
| Abhinav | 3rd Dec 24 | Updated the description and requirements | 1.0.1 |
| Abhinav | 16th Dec 24 | Updated the Features and Non-functional requirements | 1.0.2 |
| Abhinav | 20th Dec 24 | Made some minor changes in System Features | 1.0.3 |

# Introduction

## Purpose

The purpose of this SRS is to provide detailed specifications for the AI-Driven Code Review and Refactoring Assistant. The system aims to enhance developers' productivity and code quality by providing automated suggestions for code review and refactoring.

## Document Conventions

### Mandatory requirements are marked as **[MUST]**.

### Optional requirements are marked as **[MAY]**.

## Intended Audience and Reading Suggestions

### **Developers**: To understand system functionalities and to improve the implementation.

### **QA Engineers**: To create relevant test cases.

### **Project Managers**: For project oversight.

## Product Scope

The AI-Driven Assistant will provide real-time code review feedback, suggest improvements, and offer refactoring recommendations. The Assistant will integrate seamlessly into an existing development environment, enhancing code quality and consistency.

## References

### [Platform API Documentation]

### [Coding Standards Documentation]

### [Research papers on AI-driven code analysis]

# Overall Description

## Product Perspective

This Product will providing real-time insights and suggestions to the developers and it will leverage machine learning algorithms to understand code patterns and make recommendations.

## Product Functions

### Accept code from IDEs or pipelines.

### Analysis code and suggest improvements.

### Provide refactoring tips and quality reports.

### Integrate with tools like JDoodle and Gemini.

## User Classes and Characteristics

### **Developers**: Primary users who will receive actionable suggestions.

### **QA Engineers**: Secondary users who may use the tool for compliance checks.

### **Project Managers**: Use it for code quality insights.

## Operating Environment

The system will operate across Windows, macOS, and Linux environments in compatible IDEs.

## Design and Implementation Constraints

### Compliance with platform security policies.

### Integration constraints of the existing development environment.

## User Documentation

User manuals, installation guides, and online tutorials will be provided.

## Assumptions and Dependencies

### Assumes access to the code repository in read-only mode.

### The development platform must support external plugins or add-ons.

# External Interface Requirements

## User Interfaces

### **IDE Display**: Suggestions appear as inline comments or warnings.

### **Dashboard**: Provides a summary of code analysis and refactoring suggestions.

## Hardware Interfaces

NA

## Software Interfaces

### **IDE API**: For integration and feedback display.

## Communications Interfaces

### **HTTPS**: For secure communication with any cloud-hosted components.

# System Features

## Feature: Automated Code Review

### **Description**: Identifies bugs, inefficiencies, and coding standard violations.

### **Functional Requirements**:

### REQ-1: System **MUST** analyze code for errors in a file.

### REQ-2: System **MUST** provide warnings for potential bugs.

## Feature: Refactoring Suggestions

### **Description**: Suggests code improvements for readability and performance.

### **Functional Requirements**:

### REQ-1: System **MUST** recommend refactoring for complex or duplicated code.

### REQ-2: System **MUST** provide reason for each suggestion.

## Feature: Standards Enforcement

### **Description**: Ensures compliance with coding standards.

### **Functional Requirements**:

### REQ-1: System **MUST** flag non-compliant code sections.

### REQ-2: System **MUST** provide correction suggestions.

# Other Nonfunctional Requirements

## Performance Requirements

### **Requirement**: The system **MUST** handle at least 10 concurrent users with minimal latency.

## Safety Requirements

### **Requirement**: The system **MUST** maintain code integrity and prevent unintended modifications.

## Security Requirements

Key security measures include:

### Encryption of sensitive data such as code submissions, review feedback, and developer information.

### Secure communication with external APIs (JDoodle and Gemini) using encrypted API keys.

### Use of secure protocols (SSL/TLS) for all data transmission to protect against man-in-the-middle attacks.

## Software Quality Attributes

### **Reliability:** The system should provide accurate and consistent code suggestions with minimal errors.

### **Usability:** The platform should have an intuitive design that allows developers to easily submit code, view suggestions, and track progress.

### **Interoperability:** Ensure integration with existing platforms (IDE integrations, GitHub, etc.).

### **Scalability:** The platform should scale to handle increasing numbers of code submissions and reviews.

### **Performance:** Review and refactor suggestions should be delivered within a reasonable time, even for large codebases.

## Business Rules

### The system MUST comply with industry standards for code review practices and intellectual property protection.

### Developers must have access to review their own code but cannot access reviews from other developers unless explicitly shared.

### Only authorized users should be able to manage platform settings and access sensitive data (if applicable).

# Other Requirements

### **Database Requirements:** The system MUST use MongoDB for storing code submissions, review history, and developer data. The database should support fast read and write operations for handling multiple code submissions simultaneously.

### **Internationalization Requirements:** The system SHOULD support localization to cater to developers from different regions, including support for different languages in code comments, error messages, and suggestions (optional based on market demand).

### **Legal Requirements:** The system MUST comply with intellectual property laws regarding code ownership. It should include disclaimers about AI-generated suggestions and make it clear that code submission remains the intellectual property of the developer.

### **Reuse Objectives:** The codebase should be modular and reusable, allowing easy integration with other platforms and tools. The system should be built with reusable components for future scalability and maintenance.

### **Integration Requirements:** The system MUST integrate with external tools (like GitHub) to allow developers to submit code directly from their repositories.

# Appendix A: Glossary

**Code Refactoring:** The process of restructuring existing code to improve its design, readability, and performance without changing its external behavior.

**JDoodle API:** An API that allows the execution and compilation of code in multiple programming languages.

**Gemini API:** An AI-powered API used for providing code review suggestions and improvements.

# Appendix B: Analysis Models

**Data Flow Diagram (DFD):** A DFD showing how code is submitted, reviewed, and refactored within the system.

**Entity-Relationship Diagram (ERD):** A diagram representing how the collections are structured to store user data, code submissions, and review history.

# Appendix C: To Be Determined List

**TBD 1:** Define the exact data model for storing code submissions and reviews in the database.

**TBD 2:** Finalize the integration process for submitting code from GitHub repositories.

**TBD 3:** Determine if internationalization support will be implemented in the initial release or as a future feature.