Software Requirements Document

# Phase 1: Planning & Design

## Goals:

• Define use cases.  
• Establish the target audience (developers, QA, DevOps, etc.).  
• Choose technologies and platforms.

## High-Level Requirements:

• Define core features (e.g., bug detection, auto-fixing, PR creation).  
• Identify supported languages (e.g., Python, JavaScript, etc.).  
• Draft UX flow (CLI tool, GitHub App, Web UI?).  
• Define integration method (GitHub App vs. GitHub Actions).  
• Define user permissions and access scopes (e.g., repo read/write, PRs).  
• Choose AI/ML engine or static analysis tools.

# Phase 2: GitHub Integration Setup

## Goals:

• Enable the tool to access GitHub repositories and interact with them (e.g., scanning code, creating PRs).

## High-Level Requirements:

• Authenticate via GitHub OAuth or Personal Access Token.  
• Access repository files and commit history via GitHub API.  
• Detect pull request events or commits via webhooks.  
• Allow tool installation via GitHub Marketplace or manual method.  
• Define user settings (e.g., auto-repair preferences, branch targets).

# Phase 3: Code Analysis & Repair Engine

## Goals:

• Analyze code for issues and suggest/code fixes.  
• Build the core logic of the assistant.

## High-Level Requirements:

• Integrate with static analysis tools (e.g., ESLint, Pylint, SonarQube).  
• (Optional) Integrate an AI model (e.g., fine-tuned LLM) to:  
 - Understand the bug context.  
 - Generate human-like code repairs.  
 - Comment in PRs with reasoning.  
• Handle multiple bug types: syntax, logic, runtime, style.  
• Add support for multi-file analysis and dependency graphs.

# Phase 4: Automated PR Workflow & Collaboration

## Goals:

• Create a collaborative flow where the assistant proposes changes via GitHub.

## High-Level Requirements:

• Automatically create a new branch for fixes.  
• Generate a Pull Request with:  
 - Description of the issue.  
 - Summary of the fix.  
 - Test results.  
• Allow developer feedback or manual override.  
• Notify reviewers/testers via PR tagging or commenting.  
• Integrate with GitHub Actions to run tests on PR.

# Phase 5: Testing, Deployment & Monitoring

## Goals:

• Validate the tool and release it for use in real repositories.

## High-Level Requirements:

• Perform unit tests, integration tests, and regression tests.  
• Support staging vs. production environments.  
• Deploy via:  
 - GitHub App  
 - GitHub Action  
 - CLI tool  
 - Web dashboard (optional)  
• Log errors and usage metrics.  
• Add a feedback loop for users to rate suggested fixes.  
• (Optional) Roll out auto-fix mode for trusted repositories.