CodeSentinel AI - 智能代码效能与测试平台

版本号： V1.0.0
状态： 规划中
日期： 2026-01-07

一、项目概述（Overview）

CodeSentinel AI 是一个基于大语言模型（LLM）的**智能研发效能与测试平台**，致力于解决软件研发过程中“**需求-代码-测试割裂**”的问题，通过 AI 实现从需求理解、代码审查、测试生成到问题修复的**全流程自动化闭环**。

平台目标不是替代开发者，而是作为一个 **24 小时在线的 AI 架构师 + 测试专家 + 代码审计员**，持续提升研发质量与交付效率。

二、项目背景与目标（Background & Goals）

2.1 行业背景

在传统软件研发流程中，普遍存在以下问题：

- **文档与代码脱节**：需求文档更新后，代码实现未同步调整
- **回归测试成本高**：每次改动都需要全量回归，效率低下
- **代码审查质量不稳定**：依赖人工经验，容易遗漏隐性风险

这些问题在复杂业务系统和多人协作场景下被进一步放大。

2.2 核心目标

CodeSentinel AI 的核心目标包括：

1. **一致性校验**：自动验证需求文档与代码实现的一致性
2. **精准回归**：基于代码变更的真实影响范围生成测试策略
3. **自动化闭环**：实现“生成 → 执行 → 分析 → 修复”的自动循环
4. **量化提效**：
5. 回归测试时间降低 ⩾ 60%
6. Code Review 效率提升 ⩾ 40%

三、用户角色与使用场景（User Personas）

角色	典型痛点	平台价值
开发人员（Dev）	改 Bug 引入新 Bug；单测编写成本高	自动影响分析 + 单测自动生成
测试人员（QA）	用例频繁失效；定位问题困难	用例自动更新 + 根因分析

角色	典型痛点	平台价值
产品经理（PM）	难以确认需求是否完全落地	文档-代码可视化映射

四、功能需求（Functional Requirements）

> **说明：** CodeSentinel AI 不仅是一个“测试与代码工具”，更是一个 **面向产品、研发、测试三方的「需求质量与实现一致性平台」**。以下功能覆盖从需求本身质量评估，到代码实现、再到测试与修复的全链路。

模块一：智能文档理解、质量评估与映射（Docu Intelligence & Mapping）

优先级：P0（最高）

4.1.1 功能目标

帮助产品经理与研发团队 **发现需求文档本身的不足**，并建立需求与代码、测试之间的可追溯关系。

建立需求文档与代码实现之间的**语义级映射关系**，形成可追溯的需求矩阵（Traceability Matrix）。

4.1.2 输入支持

- Markdown / Word / PDF 需求文档
- Swagger / OpenAPI 接口文档
- Git 仓库代码（多语言）

4.1.3 核心能力

- 文档语义切片与向量化存储（RAG）
- 代码结构解析（类 / 方法 / 调用关系）
- 需求 ↔ 代码 映射矩阵生成

4.1.4 关键功能点

A. 需求质量智能评估（面向 PM）

1. **需求完整性检测**
2. 是否缺少异常场景说明
3. 是否缺少边界条件（如空值、极端值）
4. 是否存在“只描述结果、不描述规则”的模糊需求
5. **需求歧义与风险提示**
6. 识别模糊词（如：尽量、合理、快速、视情况）
7. 标注可能引发多种实现方式的段落
8. **需求可测试性分析**

9. 判断需求是否可被验证
10. 提示“该需求当前无法设计明确测试用例”的原因
11. **需求结构评分（PRD Score）**
12. 清晰度
13. 完整度
14. 可实现性
15. 可测试性

B. 需求-代码映射（面向 Dev / QA）

1. **需求覆盖率分析**：
2. 点击文档段落，高亮对应代码实现
3. 支持反向追溯（代码 → 需求）
4. **遗漏检测**：
5. 识别“文档存在但代码缺失”的功能点
6. **变更预警**：
7. 文档修改时，自动通知相关代码维护者
8. **需求覆盖率分析**：
9. 从文档定位到对应代码实现
10. 支持反向追溯（代码 → 需求）
11. **遗漏检测**：
12. 识别“文档存在但代码缺失”的功能点
13. **变更预警**：
14. 文档变更自动通知相关代码负责人

模块二：深度代码审查与影响分析（Smart Code Review）

优先级：P0（最高）

4.2.1 功能目标

提供接近**资深架构师水平**的自动化 Code Review 能力。

4.2.2 核心能力

1. **影响范围分析（Impact Analysis）**
2. 基于调用图、依赖关系分析变更影响
3. 示例：> “该函数变更影响 3 个 API 接口、2 个前端页面”

4. **业务逻辑审查**

5. 精度丢失
6. 边界条件缺失

7. 并发 / 死锁风险

8. **安全扫描**

9. SQL 注入
10. 硬编码密钥
11. 不安全反序列化

模块三：验证、测试与规则执行引擎（Validation & Test Engine）

优先级：P0（最高）

4.3.1 功能目标

不仅生成测试代码，更将 **需求规则转化为可执行的验证逻辑**，确保“做的对”而不仅是“跑得通”。

实现测试用例的**自动生成、自动执行与自动反馈**。

4.3.2 功能点

1. **需求驱动验证（Requirement-driven Validation）**
2. 将 PRD 中的规则转化为断言（Assertions）

3. 示例：
   > “订单金额大于 1000 需二次校验” → 自动生成校验逻辑

4. **单元测试生成**

5. 支持 Python / Java / JavaScript

6. 覆盖正常路径 + 异常路径

7. **接口测试生成**

8. 基于 Swagger 自动生成 API 测试

9. 自动处理 Token / 鉴权依赖

10. **测试执行与规则校验沙箱**

11. Docker 隔离执行

12. 同时输出：

    - 测试结果
    - 需求规则是否被满足

13. **单元测试生成**

14. 支持 Python / Java / JavaScript

15. 覆盖正常路径 + 异常路径

16. **接口测试生成**

17. 基于 Swagger 自动生成 API 测试

18. 自动处理 Token / 鉴权依赖

19. **测试执行沙箱**

20. Docker 隔离执行
21. 一键生成测试报告

模块四：自愈与自动修复（Auto-Fix & Healing）

优先级：P1（高）

4.4.1 功能目标

在测试失败后，系统具备**自主修复能力**。

4.4.2 功能点

1. **测试脚本自愈**

2. 前端 DOM / ID 变更自动适配

3. **简单 Bug 自动修复**

4. 空指针、判空缺失等
5. 输出 Git Patch 供人工审核

模块五：产品、研发与质量洞察仪表盘（Product & Engineering Insights）

优先级：P1（高）

4.5.1 功能点

1. **需求健康度视图（面向 PM）**
2. 需求质量评分趋势

3. 高风险 / 高歧义需求列表

4. **实现一致性视图（面向 Dev）**

5. 已实现 / 未实现需求
6. 高变更频率需求模块
7. **根因分析（RCA，面向 QA / Dev）**
8. AI 自动总结失败根因
9. 示例：Redis 连接超时、配置错误
10. **项目质量综合评分**
11. 测试覆盖率
12. 文档一致性
13. 技术债务趋势
14. **根因分析（RCA）**
15. AI 自动总结失败根因
16. 示例：Redis 连接超时、配置错误
17. **项目质量评分**
18. 测试覆盖率
19. 文档一致性
20. 技术债务趋势

五、非功能性需求（Non-Functional Requirements）

1. **性能要求**
2. Code Review ⩽ 30 秒
3. 测试生成 ⩽ 1 分钟
4. **数据安全与隐私**
5. 支持私有化部署
6. 公有模型调用前自动脱敏
7. **准确性与可控性**
8. 语法错误率 < 5%
9. 支持误报反馈与模型调优

六、技术架构说明（Technical Architecture）

6.1 架构分层

- **用户交互层**：Web Dashboard / 文档上传
- **智能体编排层**：任务调度与多 Agent 协作
- **数据与知识层**：向量数据库 + 代码索引
- **执行环境层**：Git + Docker + 测试运行器

6.2 架构特点

- Agent 化设计，能力可插拔
- 支持多模型接入（私有 / 公有）
- 天然支持 CI/CD 集成

七、里程碑规划（Milestones）

阶段	目标
Phase 1	文档-代码映射 + 基础 Code Review
Phase 2	自动测试生成与执行
Phase 3	自愈修复 + 智能仪表盘

八、总结

CodeSentinel AI 的核心价值在于：

> **让“需求 → 代码 → 测试 → 修复”形成一个可验证、可追溯、可自愈的智能闭环。**

它不仅是一个工具，更是一个持续进化的 AI 研发协作者。