南开大学 Nankai University

南　　开　　大　　学

计 算 机 学 院

Codeopt 需求规格说明书

徐老师班: 李佳豪 (2111252) 、钟雨哲 (2112103) 、艾明旭 (2111033)

李老师班: 马显钱 (2111917)

团队序号 ：跨班第 1 组

指导教师 ：徐思涵、李起成

2024 年 4 月 7 日

目录

1 引言 **1**
1.1 目的 . . . 1
1.2 背景 . . . 1
1.3 术语和缩略词 . . . 1
1.4 参考资料 . . . 1

2 项目概述 **2**
2.1 概述 . . . 2
2.1.1 项目来源和背景 . . . 2
2.1.2 项目目标 . . . 2
2.2 系统功能概述 . . . 2
2.3 用户分类和特定 . . . 3
2.4 假定和约束 . . . 3
2.5 数据源 . . . 3

3 功能需求 **3**
3.1 功能总览 . . . 3
3.1.1 系统功能组成 . . . 4
3.1.2 功能编号和优先级 . . . 4
3.2 功能描述 . . . 5

4 数据需求 **8**
4.1 训练数据 . . . 8
4.2 数据库描述 . . . 8
4.3 模型数据 . . . 8

5 性能需求 **9**
5.1 精度要求 . . . 9
5.2 性能要求 . . . 9

6 软硬件及外部系统接口需求 **9**
6.1 vscode 插件 . . . 9
6.2 GUI 界面 . . . 9
6.3 运行环境 . . . 9

7 可靠性与可用性需求 **9**

1 引言

1.1 目的

本次软件工程作业主题在智能运维，于是我们选定主题–开源社区的智能运维，项目名称暂定 Codeopt。我们希望在团队成员构建项目期间，提升不同小组成员代码质量，以提升最后工程项目的质量。

1.2 背景

项目名称：CODEOPT

背景：在智能运维选题基础上，结合如今智能编程助手广泛流传的情况，设计一款“面向”开源社区的软件，即更适用于开源社区的代码优化器。

1.3 术语和缩略词

Codeopt：即 Code optimizer，与软件优化代码质量的目的相应。

CVE：Common Vulnerabilities & Exposures，通用漏洞与披露。

prompt：Prompt 通常是一种短文本字符串，用于指导语言模型生成响应。Prompt 提供上下文和任务相关信息，以帮助模型更好地理解要求，并生成正确的输出。

1.4 参考资料

[1]. https://netman.aiops.org/wp-content/uploads/2023/03/2.-mt008.pdf

[2]. Qodana 使用手册:https://www.jetbrains.com/help/qodana/about-qodana.html

[3]. 漏洞扫描:https://blog.csdn.net/2301_76161259/article/details/130054686

[4]. 云效产品文档：https://help.aliyun.com/document_detail/2526762.html

[5]. SonarQube 文档 https://www.sonarsource.com/blog/

2 项目概述

2.1 概述

2.1.1 项目来源和背景

项目来源：自选项目。

项目背景：提交到开源社区的代码质量参差不齐，我们希望构建一个软件，提升 commit 到社区的代码质量，同时保证便捷性与安全性。如今网络上已存在多种智能编程助手，可以借鉴其思路，同时设计适应于开源社区的创新部分，如规整代码提升可读性、安全性等。

2.1.2 项目目标

(I) 软件整体目标

提升团队成员代码质量，自动检测代码中的错误、安全漏洞、重复项和缺陷，并给出相应的修复建议，在提交代码前，团队成员可以提交并且修复代码问题，以减少性能问题和安全问题。

(II) 提供一种自动进行代码质量检查的 pipeline 机制

模型对待提交代码进行静态检查、智能扫描，在必要时进行一些选择性的测试，最后给出建议报表供开发人员审阅。同时，提供实时的语法检查功能，以语句为单位，阶段性检查存在的语法错误。嵌入智能 AI 助手，辅助提供建议方案与修正方案，并解答用户存在的困惑。

(III) 最后，提供 CVE 漏洞扫描，调用漏洞扫描库寻找潜在的安全性问题并为用户提供可选的解决方案。

2.2 系统功能概述

功能主要可以分为四部分：实时检查、整体扫描、问答助手以及漏洞披露。

(I) 实时检查：

软件可以根据手动的分隔记号（如换行符）或者识别结构（函数体、结构体等）将代码分块。当用户在代码块内工作时，软件一方面会对尚未完成的语句进行猜测、提供可能性的补充，用户可以选择是否接受这些补充语句；另一方面，每当用户完成语句，软件都会对其进行实时的语法检查，并提示存在的错误。

(II) 整体扫描

区别于实时检查，整体扫描在更高的维度上（代码块乃至全局）对代码进行检查。整体扫描可以检查冗余（定义了而始终未被使用的变量），对代码块进行试探性的测试（如指派参数给函数体并运行），规整代码的格式以美化布局以及提供一些优化方案。

(III) 漏洞披露

用户可以从给定的漏洞脚本中选择合适的方案来进行 CVE 漏洞扫描，以期从软件提供的解决方案来完善代码的安全性能。

2.3 用户分类和特定

1. 使用软件检测代码漏洞，重复项和错误的开发者: 需要快速的检测出来代码是否具有可渗透性，及时排除高危漏洞。检测并删除代码当中的重复项。对代码的错误和缺陷提出一定的建议。
2. 使用软件对代码进行深度渗透扫描检测的企业: 需要一些软件检测相应的代码是否具有安全性，相应的漏洞以及如何进行修复
3. 辅助开发的编程者: 需要相应的代码辅助工具协助进行代码的编写和修正。

2.4 假定和约束

1. 本项目不需要考虑开发者对程序的高级优化，算法上的优化我们不进行实现，只进行简单的代码格式优化。
2. 本项目不需要考虑开发者需要的高级漏洞检测，默认个人版只有简单的代码检查，企业版才有高级的渗透检测。
3. 给定项目的代码都是我们软件可以识别的代码，并不是无法识别的未知代码。
4. 项目当中默认可以有权限调用内核态调试，以便进行渗透检测的时候可以检测系统内核。
5. 软件有权限可以申请硬件的调用，包括可以调用操作系统当中的 pipeline 机制等等，有权限进行相关的操作，以便进行打印和其他的接口功能。

2.5 数据源

- 实时检测收集的数据
- 训练数据集
- 使用者认为是安全和正确的代码作为数据
- 网络和开源社区当中默认合理的代码

3 功能需求

3.1 功能总览

- 模型 *prompt* 选择:

  根据用户不同的开发需求，提供不同模型、不同的自然语言处理模型应对不同的输入文本和问题，模型 *prompt* 应当清晰、具体，并针对所需的任务或问题进行设计。

- 实时代码规范提示

  向 IDE 嵌入插件，用户在 coding 过程中将实时收到提示，这些提示包括代码风格、代码质量、代码安全隐患等等，同时以较清晰的形式将结果展现给用户。

- 生成检测报告

  用户将代码倒入到软件，软件进行检测后以报告的形式呈现给用户，用户可以导出报告进行分享，也可以查看历史检查记录进行对比。

- 本地部署：用户可以通过下载 docker 文件进行模型的本地部署，并且提供 GUI 界面使用 codeopt 提供的功能。
- github 远程审计：用户可以在 *github* 通过配置配置文件，访问我们的在线平台在线对仓库内代码进行审查。

3.1.1 系统功能组成

- 代码扫描和解析系统：负责扫描代码、解析代码结构，提取代码内容，并对代码进行过滤和数据编码，以便后续处理和分析。包括了代码扫描、语法分析、词法分析等功能。
- 质量检测和问题识别系统：接收来自代码扫描和解析系统的数据作为输入，通过模型检测等技术识别代码中的各种问题和质量缺陷，如潜在的 bug、性能问题、安全漏洞等。
- 界面和交互系统：提供给用户的界面，如 IDE 内插件界面和 GUI 界面，用户可以通过这些界面与系统进行交互。
- 存储和管理系统：负责存储和管理系统的核心配置信息、登录口令等重要数据，也可能存储一些需要持久化的代码扫描结果和质量检测报告等数据。
- 网络系统负责处理系统与外部系统或服务之间的通信和交互，可能包括与远程服务器的数据交换、与用户的网络通信等功能。

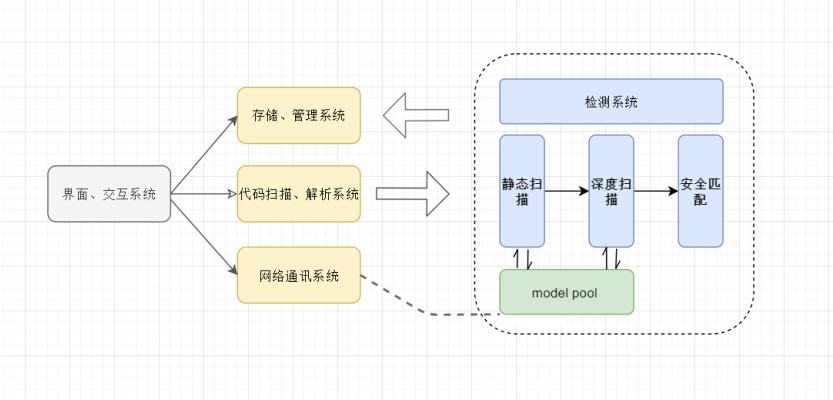

图 1: 系统组成

3.1.2 功能编号和优先级

表 1: 功能编号和优先级

功能编号	名称	优先级
FN001	质量检测和问题识别	高
FN002	代码扫描和解析	中
FN003	界面和交互系统	中
FN004	网络通讯功能	低
FN005	存储和管理系统	低
FN006	数据可视化	低

3.2 功能描述

- **模型 *prompt* 选择**:

  为了应对不同的开发场景，例如小组完成学校作业时的练习场景、小团体开发 WEB 应用的场景、团队完成大型项目的场景，在使用我们的模型之前，需要给它一些 *prompt*。选择适当的模型 *prompt* 对于获取准确、有用的模型输出至关重要。通过设计清晰、具体的模型 *prompt*，可以帮助模型更好地理解任务要求，并生成符合预期的输出。常见的模型 prompt 选择策略有如下:

  * 清晰明了的问题描述：在模型 prompt 中清晰地描述问题或任务的要求，以确保模型能够正确理解并提供相关的答案或输出。
  * 示例引导：提供相关示例或示例输入，以帮助模型更好地理解所需任务的上下文和要求。这有助于模型更准确地生成所需的输出。
  * 限制条件：在模型 prompt 中设定适当的限制条件，以确保模型生成的输出符合预期并且满足特定的要求。这可以包括词汇选择、输出长度、内容约束等。
  * 提示调整：根据模型的性能和输出结果进行模型 prompt 的调整。如果模型的输出不符合预期，可以尝试调整模型 prompt 以改善结果。
  * 上下文引入：在模型 prompt 中提供适当的上下文信息，以帮助模型更好地理解问题并生成相关的输出。这可以包括相关背景知识、先前对话内容等。

  具体 codeopt 而言，我们可以给出如下 *prompt*:

  ▶ 项目目标
  ▶ 编程语言、开发框架
  ▶ 开发规范：如函数命名方式、文件组织方式
  ▶ 输入的不同文本需要的自然语言处理模型。
  ▶ 旧有的处理相关问题的思路

  选择的模型还能适当的限制条件以保证输出格式符合模型的预期，包括词汇选择，输出长度，内容范围等等，都可以根据问题的不同类型和模型的假设以及用户的权限等等规定。

- **实时代码规范提示**:

  1. 向 IDE 嵌入插件，用户在 coding 过程中将实时收到提醒。
  2. 支持代码风格检查（缩进、命名规范、注释等）、代码质量检查（未定义变量、未处理异常等）、性能检查（循环次数、不必要的对象创建等）、安全检查（安全漏洞、CVE 特征匹配等）。
  3. 在 IDE 中以直观的方式显示代码规范问题，并提供相应的修复建议。

  实时代码检测的业务流程如图2所示:

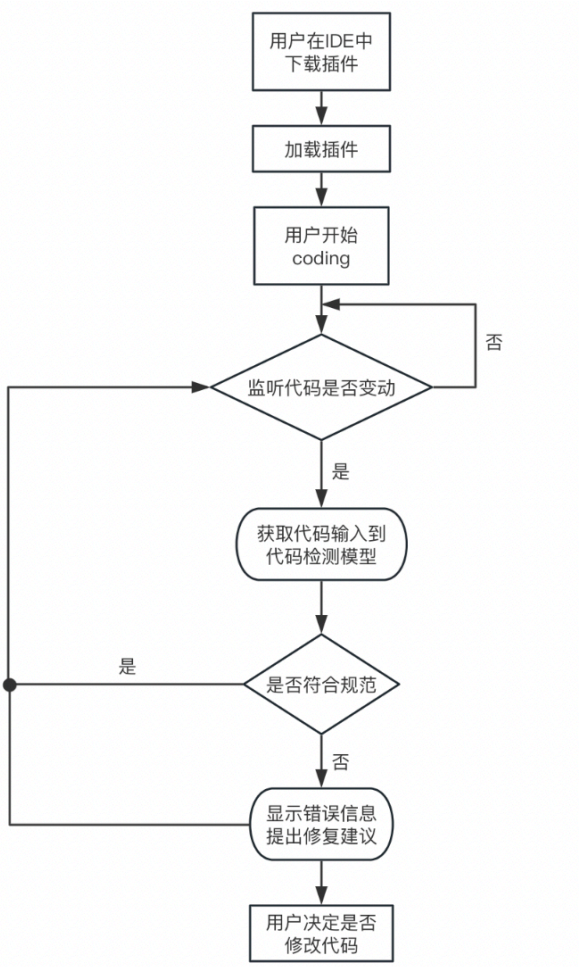

图 2: 实时代码检测流程图

用户首先在 IDE 中下载插件，再加载好插件后就可以开始 coding。插件会实时监听代码是否发生变动，如果发生，则获取代码输入到代码检测模型中，模型检测代码是否符合相关已定义的规范，如果出现不符合代码规范的地方，以用户友好的方式显示代码规范问题，提出修复建议。用户此时可以决定是否修改代码，插件则会继续监听。

- **生成检测报告**

step 1 代码导入：允许用户将其代码导入到软件（网页或插件）中，支持多种编程语言和文件类型。

step 2 代码检测：对导入的代码进行分析，检测其中可能存在的错误和代码质量问题。

step 3 检测规则：提供一系列的检测规则，例如错误的命名约定、未使用的变量、未处理的异常等，用户可以选择启用或禁用这些规则。

step 4 检测报告生成：将检测结果以报告的形式呈现给用户，包括代码中发现的问题列表、问题的详细描述、问题所在的文件和行号等信息。

step 5 报告导出：允许用户将检测报告导出为常见的文件格式，如 PDF、HTML、CSV 等，方便用户与其他人共享或存档。

step 6 历史记录：记录每次代码检测的结果和报告，方便用户进行对比和追踪，同时提供统计信息和趋势分析。

如图3所示

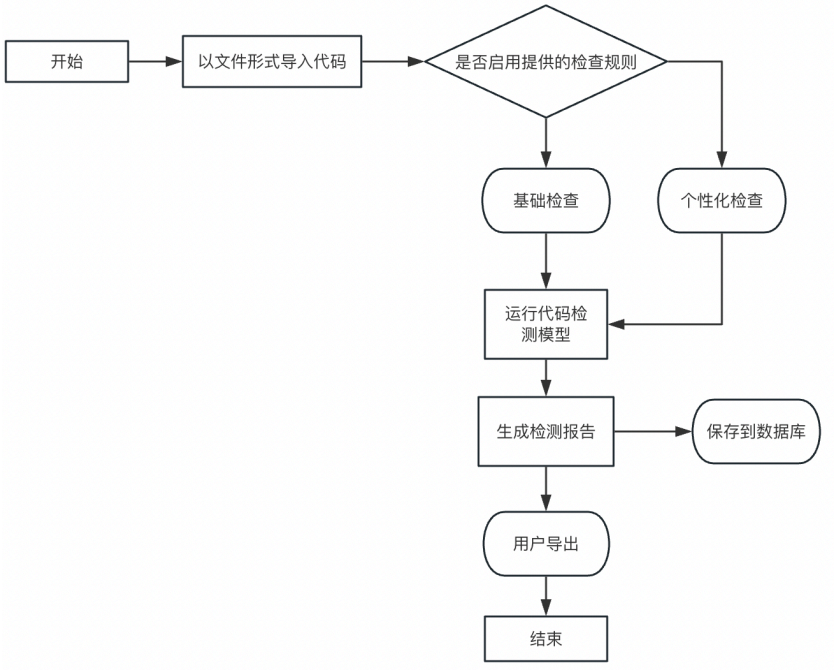

图 3: 检测报告生成流程图

- **本地部署**

  您可以通过 Docker 部署我们的应用，并且通过访问 GUI 界面以完成相关代码审查任务

```
docker pull Nkufond/codeopt

docker run -d -p 3000:3000
    -e model=chatgpt3.5 tuple
    -e CODE= 页面访问密码
    Nkufond/codeopt
```

  同时也可以指定 *proxy*

```
docker run -d -p 3000:3000
    -e model=chatgpt3.5 tuple
    -e CODE= 页面访问密码
    –net=host
    -e PROXY_URL=http://127.0.0.1:7890
    Nkufond/codeopt
```

  如果你需要指定其他环境变量，请自行在上述命令中增加 -e 来指定。

- **github 远程审计**

  用户在待检查的 github 仓库的根目录上传配置文件（用于验证个人身份、上以及待检查代码目录），之后在 codeopt 在线平台进行代码审查。

  yaml 配置示例：

```
name: Codeopt Scan
user: FondH/1234567@qq.com
scan-list:
    scac-dirs:*
    scac-level:High
```

4 数据需求

4.1 训练数据

微调现有的大型模型通常需要大量的训练数据，包括 *Github*、*GitLab* 的代码库、开源项目的 Issues 和 Pull Requests、一些代码质量评估工具（如 SonarQube、CodeClimate 等）提供了关于代码质量和可维护性的度量指标以及 Stack Overflow 等网站上的问题和答案可以提供有关代码中常见错误的信息。

4.2 数据库描述

User-information: 需要存储小组内每个人的个人信息
project-preferences: 项目信息，包括项目名字、prompt 等等
model-list: 模型信息，包括所有可用模型的名字、路径等
history: 操作历史

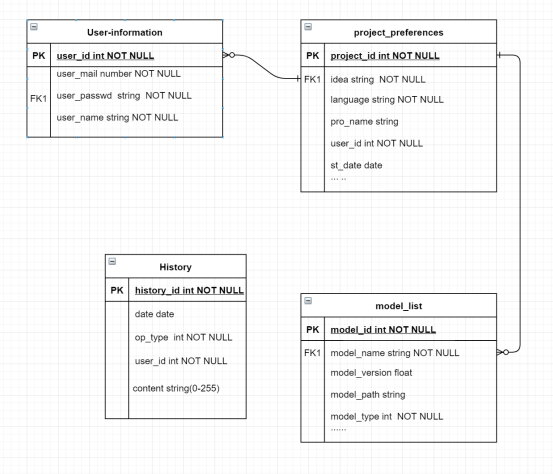

图 4: 数据库描述

4.3 模型数据

1. 应用中使用大模型时，包括本地存格式包括本地文件系统、网络文件系统（如 NFS）、云存储服务（如 AWS S3、Google Cloud Storage）等，本地以 TensorFlow 的 SavedModel 格式、PyTorch 的 TorchScript 存取，远程则通过调用其他大模型提供的 API 微调后使用。
2. ：版本控制：若有多个模型版本，需要考虑如何管理这些版本，包括命名规范、版本控制系统等。确保能够轻松地切换和管理不同版本的模型。
3. 安全性：对于一些敏感或者重要的模型，需要考虑模型存储的安全性，包括对访问权限的控制、数据加密等措施。

5 性能需求

5.1 精度要求

表 2: 精度需求

类别	字段	精度要求
User	邮箱	邮箱格式
	密码	6-18 位数字或字符
model	密钥	根据所需模型要求
	名称	模型名称的字符

5.2 性能要求

用户在本地或远程使用时，都希望能够快速得到检测结果，以便及时解决问题；为了能够在短时间内完成代码检测和生成检测报告的操作。

此外支持多个用户同时进行代码检测和生成检测报告的操作。在开源社区中，可能有多个开发者**同时提交**代码，因此软件需要处理多个请求，并保持良好的性能。

因此无论本地部署、异或服务器部署均对机器有较高要求。

6 软硬件及外部系统接口需求

6.1 vscode 插件

作为和用户交互最频繁、直接的部分，设计一个优雅、美观、操作简便的插件是重要的。vscode 应更新到最新版本。

6.2 GUI 界面

用户界面是程序中用户能看见并与之交互作用的部分，设计一个好的用户界面是非常重要的，本设计将为用户提供美观，大方，直观，操作简单的用户界面。

6.3 运行环境

- **硬件系统**：移动终端硬件配置应遵循如下原则：具有高的可靠性，可用性和安全性。【描述系统中软件和硬件每一接口的特征。这种描述可能包括支持的硬件类型、软硬件之间的交流的数据和控制信息的性质以及使用的通信协议。】
- **操作系统**：Windows10+
- **Web 浏览器**：0+、Chrome、Opera、Safari、Firefox 及任何支持 HTML5 标准的浏览器。

7 可靠性与可用性需求

1. **可维护性**

考虑到开源社区的规模和发展，软件应具备良好的可扩展性，能够适应未来的增长。例如，能够处理更多的代码量、支持更多代码检测工具、适应更多的远程部署方式等。软件应具备完善的日志和监控功能，能够记录用户的操作和系统的运行情况。这样可以方便进行故障排查和性能优化，并及时发现和解决问题。

2. **用户友好性**

软件应具备一定的可定制性，能够根据用户的需求和偏好进行配置和调整。例如，用户可以选择使用不同的代码检测工具、调整检测报告的格式和内容等。软件应具备良好的用户界面和交互方式，方便用户进行操作和查看检测结果。用户应能够轻松地理解和使用软件的功能，减少学习成本和使用障碍。软件应具备简单易用的部署方式，能够快速在本地或远程环境中进行安装和配置。同时，软件的维护也应简单高效，能够方便地进行升级、修复漏洞和添加新功能。

3. **安全与可靠性**

软件应具备高可靠性，能够稳定地运行并正确地生成检测报告。在检测过程中，不应出现崩溃、错误或数据丢失等问题，以确保用户能够获得准确和可靠的检测结果。考虑到开源社区的代码可能涉及敏感信息，软件应具备一定的安全性，能够保护用户的代码和数据不被未经授权的人员访问或篡改。