学号：171860023 姓名：符冰瑶

# Smart IDE需求文档

**1.业务需求**

* 实现一款智能的Smart IDE，除了传统的代码编写、分析、编译、调试、执行等IDE 的基本功能，还要有让人眼前一亮的创新的功能，打败市场上的同类产品。
* 与我公司其他产品能有效形成互动。
* 增加开发人员的生产力，同时也将各种开发工作做更密切的整合，来提高生产力。

**2.功能需求**

* **实时共享**：实时协作开发共享代码。

1. 会话发起和邀请：每个协作项目包含一个发起人和一个或多个协作者。发起人创建一个协作会话，然后邀请其他协作者进入会话。
2. 访问控制：IDE对项目中的文件进行访问控制。发起人作为协作项目的管理员，具有所有文件的读写权限，并能够设置所有文件的访问权限（只读／可读写／不可访问），或者增加新的管理员。
3. 共享编辑：协作者加入会话后，能够实时看到别人的编辑，并且可以对自己具有写权限的文件进行编辑。
4. 共享调试：开发者完成某一文件的编辑后，可以发起共享调试会话，进入会话的每个开发者可以进行自己的调试，同时能够实时查看其他开发者的调试情况。
5. 共享终端：开发者可以以只读／可读写的方式与别人共享一个终端。
6. 提供会话框：开发者可以互相交流。
7. 文件本地导出／上传：开发者可以将会话中的文件导出到本地，也可以也可以将本地文件上传到会话中。

* **IDE中的Git管理和仓库创建、克隆**

1. 登录：IDE与GitHub连接，程序员可在IDE中登录GitHub.com和Github账户。
2. 同步功能：IDE与Github连接后，可以将本地项目发布到Github仓库。
3. 克隆仓库功能：将连接的Github的仓库克隆到IDE中。
4. 创建仓库功能：在IDE中创建Github仓库并且将其克隆到IDE中。
5. 在IDE中查看、提交pull request。
6. 版本控制功能：可查看参考、作者、测试、提交历史记录和其他重要信息。支持版本回退。

* **基于人工智能的智能预测**

1. 当程序员编写代码时，IDE能够当前的代码上下文和模式，利用人工智能技术，从网上的源代码或者程序员自己的代码学习、预测程序员此时最可能需要的 API，提供一个API参考列表。
2. 利用人工智能学习用户的行为，如：输入习惯、代码风格、偏好、常犯错误等。根据学习的结果，在用户写代码时，可能发生错误的地方自动提示，按照用户习惯调整推进、自动填充空格、换行等，同时保护用户隐私。

* **智能搜索**

提供一个搜索框，用户输入问题/关键字，IDE能在网上/自己的数据库中搜索相关信息，利用人工智能辅助给出推荐的算法/函数/源代码/示例。

* **变量/函数定位**

在代码中双击变量/函数或者在搜索框搜索函数/变量时，立即跳到定义点；按下“→”键，跳到声明点；用阴影标记项目中所有的引用点标记，按下⬇，跳到下一个引用点。

* **监测、可视化显示栈**

1. 代码调试时，监测每一个函数的栈从函数调用到函数返回的变化情况。
2. 进行可视化处理，以图的形式显示。

* **智能测试**

智能单元测试组件

1. 自动生成测试用例：打开单元测试组件，选择要测试的单元，IDE利用Al辅助自动生成测试用例。程序员可以根据自己的需求修改、增、删测试用例。
2. 点击测试，组件就能根据测试用例自动进行测试。
3. 测试结束后，提供结果和错误分析报告，包括错误类型、错误代码定位以及Al辅助的原因推断和debug建议。

* **实时错误预防：**能够立即了解写代码时所做的每个更改的影响，并在用户键入代码时对其进行测试。在错误发生时立即提示/修改错误，并查看现有测试是否覆盖了新更改。

1. 实时对代码中的语句，生成执行该语句的测试输入；实时针对不同的方法，生成测试用例，并进行实时的单元测试。
2. 在代码编辑器中，在每行代码的最左端，显示它是否由单元测试覆盖/覆盖这些代码的测试是否通过。程序员可以点击查看每行代码的测试信息（如：未通过的测试名和用例、未通过的原因、人工智能给出的修改建议）。

* **在线Q&A交流平台**
* **快速、安全地重构**

在代码中的多个位置进行类似的编辑时（对某个变量或函数进行重命名、移动、删除、提取方法、嵌入变量、向上/向下推动成员、 改变签名等），只需几下点击操作，即可执行全项目范围内的改变，并且可以安全地撤销操作。

* **代码可视化和逆视化**

1. 静态可视化：
   * 1. IDE能够分解出代码中的各语法成分，分析它们间的逻辑关系和代码的语义、结构。
     2. IDE定义各种可视化结构对应的的标准视化形式。可视化结构包括类、源文件、复杂数据结构、指针及其指向等。视化形式包括流程图/流程树、数据结构图、类图、反映指针的指向情况的图等。
     3. IDE能够根据代码生成相应的标准视化形式。
2. 动态可视化：
   * 1. 在调试/测试/执行过程前，IDE对代码中的所有变量进行可视化，生成该程序的程序状态（代码中所有变量及其值的集合）结构框架。
     2. 在调试/测试/执行过程中，IDE记录下每一个（或者指定位置的）程序状态（向框架中填值）。
     3. 最终生成一个可用类似PPT放映的形式播放的状态转移动画，程序员每点击一次鼠标，就播放下一个状态。例如：动态播放插入排序的过程。
3. 逆视化

由程序员输入一个符合IDE标准定义的视化形式，IDE根据输入的视化形式，生成相应的可视化结构的代码。

**3.非功能需求**

* 软件的安装、环境配置尽量简便、智能，减少用户的手动工作量。
* 优秀的图形化界面。在界面和操作上支持用户自定义设置。
* 良好的系统运行速度、可靠性、容量、可用性、可使用性。
* 良好的编程接口:可以方便的安装卸载开发所需要的工具和组件包等工作负载。需要支持尽量多种编程语言、在不同平台不同终端的开发。
* 较小的系统运行内存。
* 目标机器配置要求尽可能低。

**4. 约束与限制**

* 软件产品完成的时间以及开发费用问题待定。
* 软件安全问题，软件盈利模式为订阅收取费用，故防盗版破解要做好。
* 对软件体量进行增减，分化出三个版本:轻量、基本功能(免费版)，面向小型开发团队的版本以及可以满足任何开发规模的版本。
* 我公司只有C++程序员，为了产品交付方便，项目必须用C++完成。
* 支持在windows和macos上运行。