一、用例

1. 用户需求管理

- **用例名**：编辑用户需求
- **参与者**：需求分析师、普通用户
- **描述**：用户通过界面编辑、分类和设置需求优先级，支持文本输入、需求关联和版本管理。
- **子用例**：
    - 创建需求文档
    - 修改需求内容
    - 标记需求状态（待评审/已确认/已冻结）

---

2. 架构与数据库设计生成

- **用例名称**：生成系统架构
- **参与者**：架构师、大模型（自动触发）
- **描述**：根据需求文档生成可执行的架构代码（如微服务分层、API 定义）和数据库 Schema。
- **输入**：需求文档（自然语言或结构化数据）
  输出：架构代码框架（如 Spring Boot 项目骨架）、数据库 ER 图及 DDL 脚本。

---

3. 模块代码生成

- **用例名称**：生成模块代码
- **参与者**：程序员、大模型
- **描述**：基于架构设计生成具体模块代码（如 Controller/Service/Dao 层），支持手动调整。
- **扩展流程**：
    - 用户手动修改生成后的代码
    - 代码规范性检查（如 Lint 工具集成）

---

4. 测试用例生成与执行

- **用例名称**：生成测试用例
  参与者：测试人员、大模型
- **描述**：根据模块代码自动生成单元测试、接口测试用例（如 JUnit/Postman 脚本）。
- **用例名称**：执行自动化测试
- **参与者**：测试人员、CI/CD 工具
- **描述**：部署代码到测试环境，运行测试并生成报告（覆盖率、通过率）。

---

5. 部署与监控

- **用例名称**：部署到生产环境
- **参与者**：运维人员、自动化部署工具
- **描述**：通过容器化（Docker/K8s）或云服务（AWS/Azure）部署系统，监控运行状态。

---

6. 协作与配置管理

- **用例名称**：管理项目配置
- **参与者**：配置管理员
- **描述**：设置代码规范（如 Git 分支策略）、依赖版本、环境变量等。
- **子用例**：
    - 版本控制（Git 提交/回滚）
    - 代码评审流程管理

---

7. 文档生成与导出

- **用例名称**：导出项目文档
- **参与**：所有角色
  描述：一键生成需求文档、设计文档、测试报告等，支持 PDF/Markdown 格式。

---

8. 权限与团队管理

- **用例名称**：管理用户权限
- **参与者**：项目经理
- **描述**：分配角色权限（如开发只读/编辑权限）、管理团队成员。

---

9. 异常处理用例

- **用例名称**：处理生成错误
- **参与者**：系统、用户
- **描述**：当大模型生成代码或文档不符合预期时，提示用户手动干预或重新生成。
    - **示例场景**：
        - 生成代码存在语法错误
        - 需求描述模糊导致架构设计冲突

二、系统界面原型

一、原型介绍

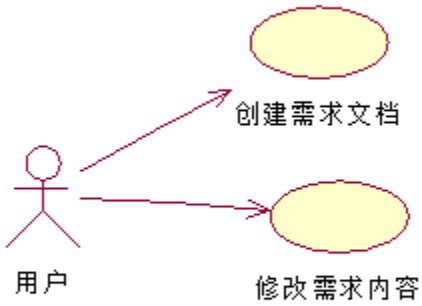
1.多个 AI Agent，扮演不同角色（比如：产品经理、架构师、前端工程师、后端工程师、测试工程师等），在平台内交互协作。

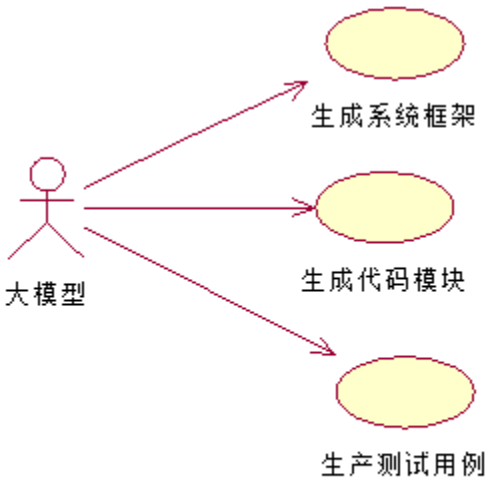
2.用户可以指挥、观察、干预这些 Agent 的讨论与开发过程。

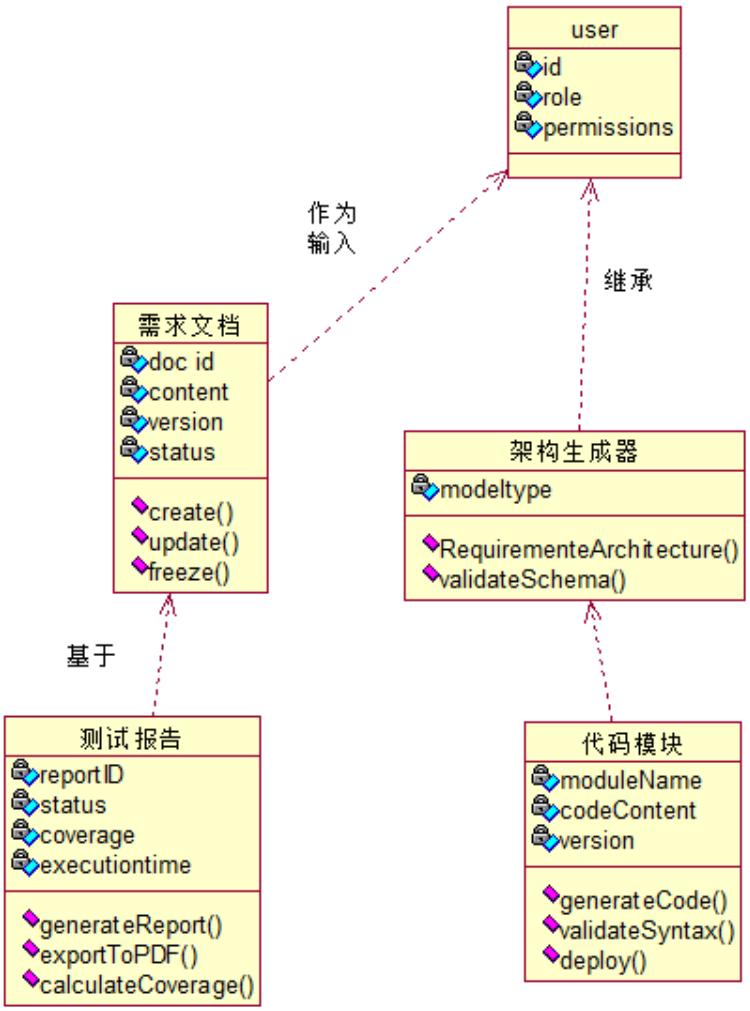
3.整个平台偏向软件工程流程管理，比如需求分析、设计、开发、测试、部署等阶段。

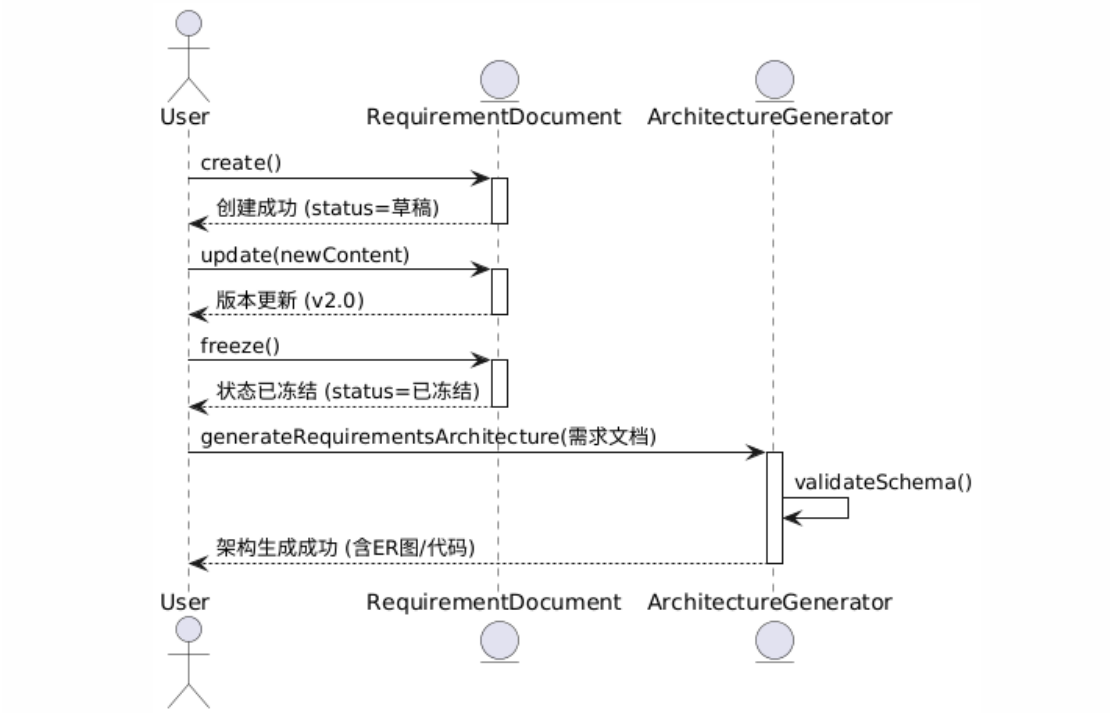
4.界面符合“多 Agent 交互”的特点

三、UML 建模


创建需求文档
用户
修改需求内容


生成系统框架
大模型
生成代码模块
生产测试用例


user
id
role
permissions
作为
输入
维承
需求文档
doc id
content
version
status
create()
update()
freeze()
架构生成器
modeltype
RequirementeArchitecture()
validateSchema()
基于
测试报告
reportID
status
coverage
executiontime
generateReport()
exportToPDF()
calculateCoverage()
代码模块
moduleName
codeContent
version
generateCode()
validateSyntax()
deploy()


User
RequirementDocument
ArchitectureGenerator
create()
创建成功 (status=草稿)
update(newContent)
版本更新 (v2.0)
freeze()
状态已冻结 (status=已冻结)
generateRequirementsArchitecture(需求文档)
validateSchema()
架构生成成功 (含ER图/代码)
User
RequirementDocument
ArchitectureGenerator