需求分析规格说明文档

项目组

2025 年 3 月 31 日

目录

1 文档概述 **4**

1.1 文档目的 4

1.2 项目背景 4

1.3 读者对象 4

1.4 参考文献 4

2 系统概述 **4**

2.1 系统目标 4

2.2 系统范围 5

2.3 系统架构概述 5

3 用户需求 **6**

3.1 用户角色定义 6

3.2 用户场景 6

4 功能需求 **6**

4.1 UI 组功能需求 6

4.2 System 组功能需求 7

4.3 Data 组功能需求 11

4.4 Analysis 组功能需求 11

5 非功能需求 **12**

5.1 性能需求 12

5.2 可靠性需求 12

5.3 安全需求 12

5.4 可维护性需求 13

6 用例分析 **13**
6.1 系统组用例总览 . . . 13
6.2 系统组用例分析 . . . 14
6.2.1 构建部署流程 . . . 14
6.2.2 集成第三方服务 . . . 15
6.2.3 协同开发数据库 . . . 16
6.2.4 API 设计与实现 . . . 17
6.2.5 性能优化 . . . 18
6.2.6 安全保障 . . . 19
6.2.7 监控与日志系统 . . . 20
6.3 其他关键用例 . . . 20

7 数据需求 **21**
7.1 数据实体 . . . 21
7.2 数据字典 . . . 21
7.3 数据流 . . . 21

8 接口需求 **22**
8.1 System 组与其他组接口概述 . . . 22
8.2 System 组与 UI 组接口 . . . 23
8.2.1 API 接口规范 . . . 23
8.2.2 核心 API 接口 . . . 23
8.2.3 前后端交互流程 . . . 24
8.3 System 组与 Data 组接口 . . . 24
8.3.1 数据库访问接口 . . . 24
8.3.2 主要数据交互场景 . . . 25
8.3.3 职责分工 . . . 25
8.4 System 组与 Analysis 组接口 . . . 25
8.4.1 数据分析 API . . . 25
8.4.2 主要交互场景 . . . 26
8.4.3 数据格式规范 . . . 26
8.5 跨组协作接口 . . . 26
8.5.1 通知与事件系统 . . . 26
8.5.2 集成测试接口 . . . 26
8.6 接口文档与版本管理 . . . 27
8.6.1 接口文档规范 . . . 27
8.6.2 接口版本管理 . . . 27
8.6.3 接口变更管理 . . . 27

9 约束与假设 **27**
9.1 技术约束 27
9.2 业务约束 28
9.3 假设条件 28

10 验收标准 **28**
10.1 功能验收标准 28
10.2 非功能验收标准 28

11 附录 **29**
11.1 术语表 29
11.2 用例关系说明 29
11.3 修订历史 29

1 文档概述

1.1 文档目的

本文档是项目的需求分析规格说明文档，旨在详细描述系统的功能需求、非功能需求、用例分析和各模块交互关系，作为后续设计和开发工作的基础依据。它明确定义了用户需求，并将这些需求转化为可实现的技术规范。同时，通过用例图描述 System 组在项目中的职责和与其他组（UI 组、Data 组和 Analysis 组）的交互关系，为项目各方提供清晰的职责界定和协作指南。

1.2 项目背景

本项目旨在开发一个综合性系统，由四个主要组成部分协同工作：系统组（System）、用户界面组（UI）、数据组（Data）和分析组（Analysis）。这四个组各司其职，共同构建一个完整的解决方案，以满足用户的需求和业务目标。系统组作为项目的技术基础设施提供者，是项目技术架构的设计者和实现者，负责系统核心功能的开发。

1.3 读者对象

本文档适用于以下读者：

- 项目管理人员：了解项目整体需求和范围
- 开发团队成员：理解具体技术需求和实现细节
- 测试人员：制定测试计划和用例
- 用户代表：确认系统是否满足业务需求

1.4 参考文献

1. Claude 3.7
2. QwQ

2 系统概述

2.1 系统目标

本系统旨在 [此处描述系统的主要目标和要解决的问题]。通过整合用户界面、系统架构、数据处理和分析功能，为用户提供全面的解决方案，提高工作效率和决策质量。

2.2 系统范围

本系统包括但不限于以下功能范围：

- 用户界面展示和交互
- 系统核心架构和服务
- 数据采集、存储和管理
- 数据分析和可视化

2.3 系统架构概述

系统由四个主要组件构成，各组件职责如下：

- **系统组（System）**：负责核心架构设计、API 实现、第三方服务集成、系统部署、性能优化、安全保障及监控系统
- **用户界面组（UI）**：负责用户界面设计和实现、用户体验优化、前端与后端 API 交互
- **数据组（Data）**：负责数据模型设计、数据库实现、数据处理流程、数据质量保障
- **分析组（Analysis）**：负责数据分析算法、报表生成、数据可视化、决策支持模型

各组件之间通过定义良好的接口进行交互，如下图所示：

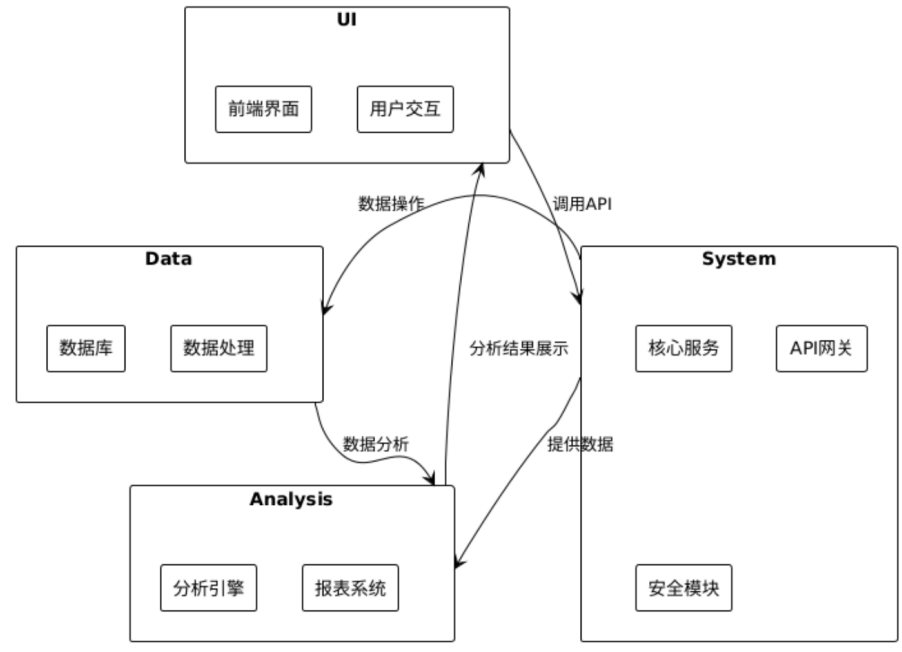

图 1: 系统架构概述图

3 用户需求

3.1 用户角色定义

系统识别了以下主要用户角色：

1. **系统管理员**：负责系统管理和维护
2. **普通用户**：使用系统的主要功能
3. **分析师**：进行数据分析和报表生成
4. **数据管理员**：管理系统数据和权限

3.2 用户场景

以下是主要用户场景的概述：

1. 用户登录与认证
2. 数据录入与编辑
3. 数据查询与检索
4. 数据分析与报表生成
5. 系统管理与配置
6. 安全审计与监控

4 功能需求

4.1 UI 组功能需求

1. **用户界面设计**
   - 设计符合用户体验原则的界面布局
   - 实现响应式设计，适配不同设备
   - 提供统一的视觉风格和组件库
2. **用户交互实现**
   - 实现用户登录、注册、密码重置等基础功能
   - 提供个性化设置和主题定制

- 实现多语言支持

3. **前端数据展示**

- 实现数据表格、图表等展示组件
- 提供数据筛选、排序和分页功能
- 支持数据导出和打印

4.2 System 组功能需求

System 组在项目中的主要职责是作为技术基础设施提供者，负责系统核心功能的开发。以下是 System 组的具体功能需求：

1. **构建部署流程**

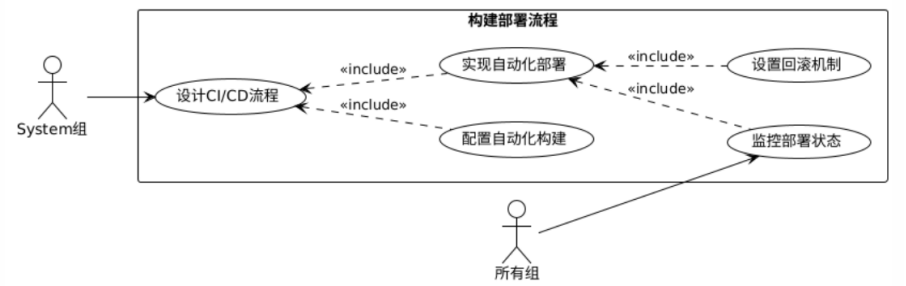

图 2: 构建部署流程用例图

- 设计并实现完整的 CI/CD 流程
- 配置自动化构建和测试环境
- 提供部署状态监控
- 建立部署失败的回滚机制

2. **集成第三方服务**

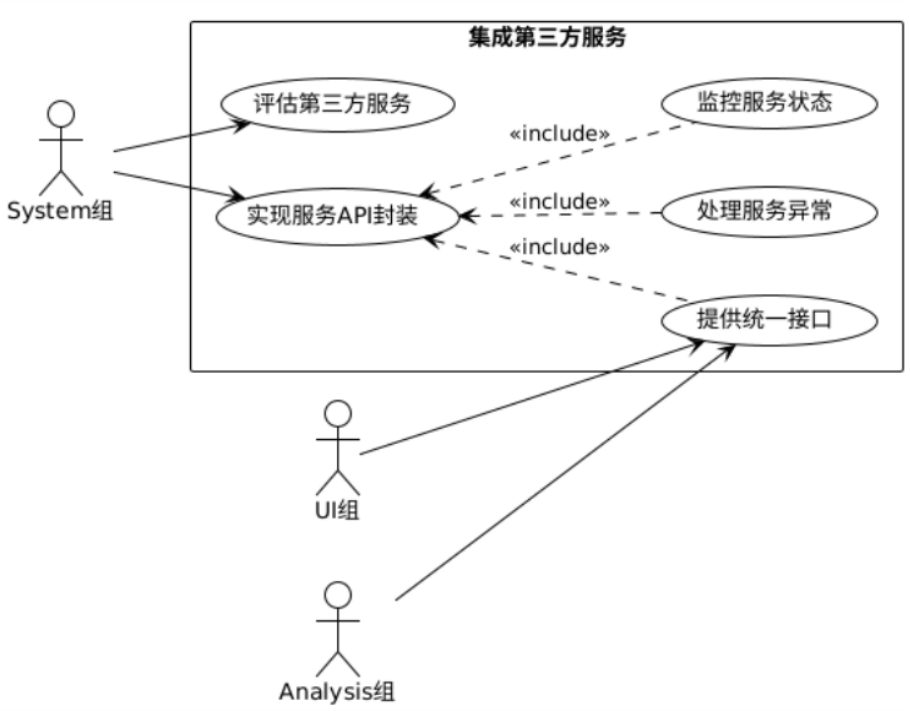

图 3: 集成第三方服务用例图

- 评估和选择适合项目需求的第三方服务
- 实现对第三方 API 的封装
- 提供统一的接口规范
- 处理第三方服务异常情况
- 监控第三方服务的可用性和性能

3. **协同开发数据库**

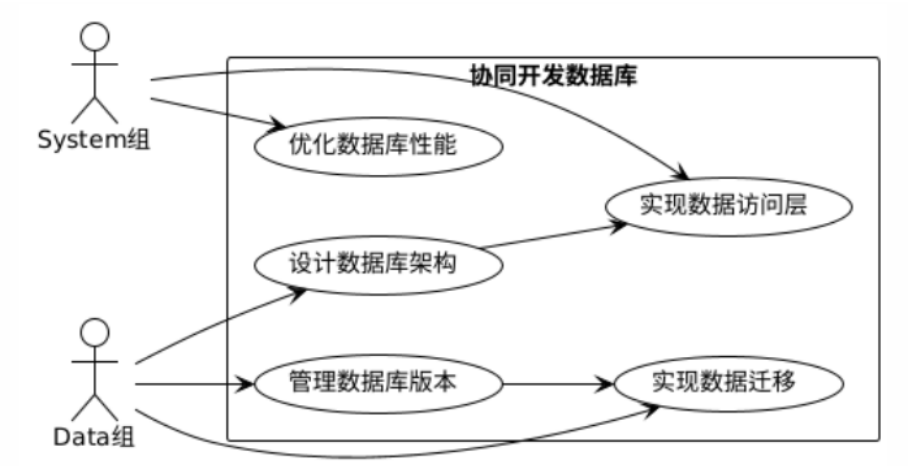

图 4: 协同开发数据库用例图

- 与 Data 组协作实现数据访问层

- 优化数据库性能
- 实现数据缓存机制

4. **API 设计与实现**

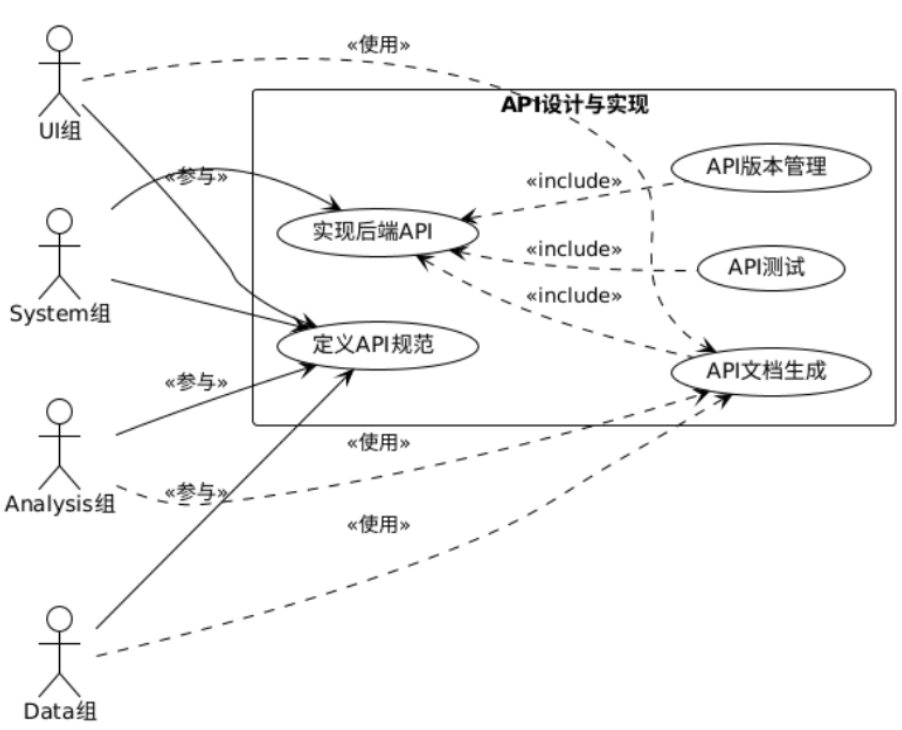

图 5: API 设计与实现用例图

- 制定 API 设计规范和标准
- 实现所有后端 API 功能
- 生成完整的 API 文档
- 进行 API 单元测试和集成测试
- 管理 API 版本和兼容性

5. **性能优化**

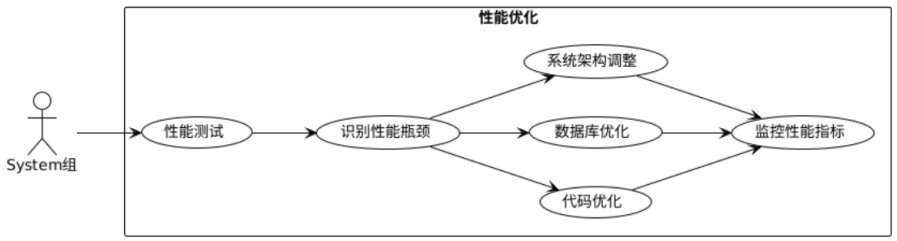

图 6: 性能优化用例图

- 设计并执行性能测试方案
- 分析并识别系统性能瓶颈
- 优化代码执行效率
- 与 Data 组协作进行数据库性能优化
- 必要时进行系统架构调整
- 持续监控系统性能指标

6. **安全保障**

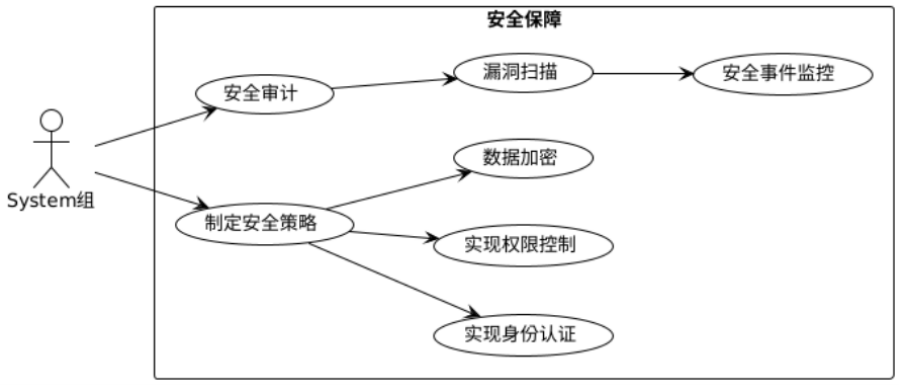

图 7: 安全保障用例图

- 制定系统安全策略和标准
- 实现身份认证和授权机制
- 实现细粒度的权限控制
- 敏感数据加密存储和传输
- 防止常见安全威胁（SQL 注入、XSS 等）
- 定期进行安全审计和漏洞扫描
- 实施安全事件监控和响应机制

7. **监控与日志系统**

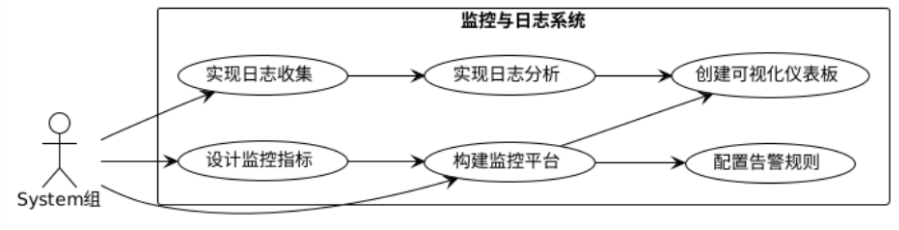

图 8: 监控与日志系统用例图

- 设计系统关键监控指标
- 实现全面的日志收集机制
- 构建统一的监控平台
- 配置合理的告警规则
- 实现日志聚合和分析功能
- 创建直观的可视化仪表板

4.3 Data 组功能需求

1. **数据库设计**
   - 设计数据库模型和表结构
   - 定义数据约束和关系
   - 实现数据版本控制
2. **数据处理流程**
   - 实现数据 ETL（提取、转换、加载）流程
   - 提供数据批量导入和导出功能
   - 实现数据清洗和验证机制
3. **数据质量管理**
   - 实现数据质量检查规则
   - 监控数据完整性和一致性
   - 提供数据修复工具

4.4 Analysis 组功能需求

1. **数据分析算法**
   - 实现统计分析功能
   - 提供预测分析模型
   - 支持自定义分析规则
2. **报表生成**
   - 设计标准报表模板
   - 支持自定义报表

- 实现报表定时生成和分发

3. **数据可视化**
   - 提供多种图表和可视化组件
   - 支持交互式数据探索
   - 实现数据钻取和聚合分析

5 非功能需求

5.1 性能需求

1. **响应时间**
   - 页面加载时间不超过 3 秒
   - API 接口响应时间不超过 1 秒
   - 报表生成时间不超过 5 秒
2. **并发处理**
   - 系统支持至少 100 个并发用户
   - 数据库能同时处理 50 个并发事务

5.2 可靠性需求

1. **数据备份与恢复**
   - 每日进行数据备份
   - 数据恢复时间不超过 4 小时
2. **错误处理**
   - 系统提供明确的错误提示
   - 关键操作提供回滚机制

5.3 安全需求

1. **身份认证与授权**
   - 实现多因素认证
   - 基于角色的访问控制

- 定期密码更新策略

2. **数据安全**

- 敏感数据加密存储
- 通信加密（HTTPS）

5.4 可维护性需求

1. **模块化设计**

- 系统采用松耦合、高内聚的设计原则
- 支持模块独立更新和部署

2. **文档完备**

- 提供详细的系统架构文档
- 编写完整的 API 文档
- 维护代码注释和关键算法说明

3. **可测试性**

- 支持自动化单元测试和集成测试
- 提供测试环境和测试数据

6 用例分析

6.1 系统组用例总览

以下用例图展示了 System 组的主要职责和与其他组的交互关系：

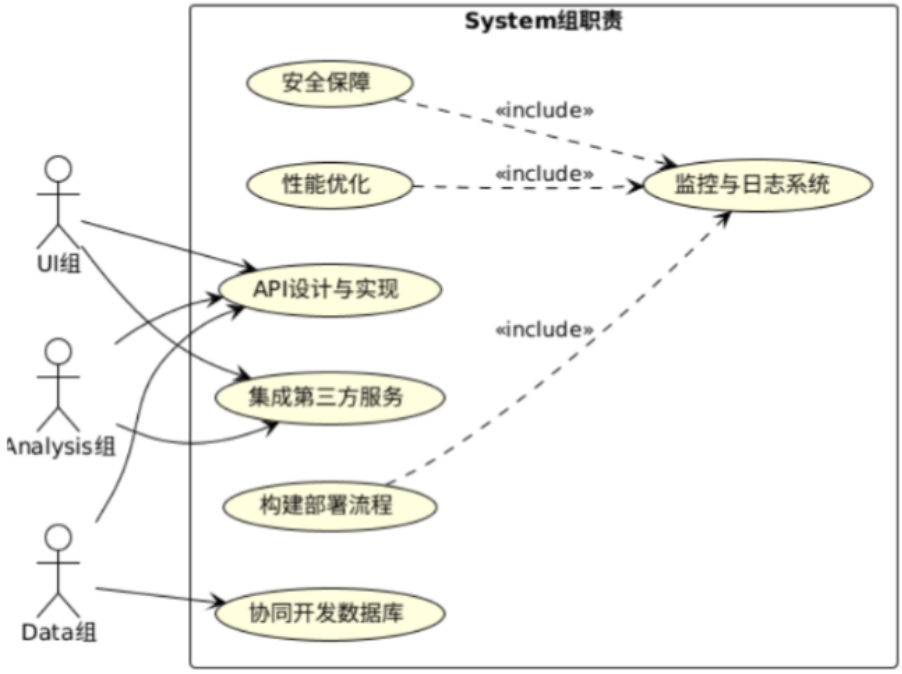

图 9: 系统组用例总览图

6.2 系统组用例分析

6.2.1 构建部署流程

用例名称：构建部署流程
行为主体：System 组
前置条件：

- 项目代码仓库已建立
- 部署环境已准备就绪

后置条件：

- 自动化 CI/CD 流程可正常运行
- 各组可使用部署流程部署代码

基本流程：

1. System 组设计 CI/CD 流程
2. 配置自动化构建环境
3. 实现自动化部署机制
4. 设置部署监控和回滚机制

5. 各组使用部署流程部署代码

替代流程：

1. 部署失败时，自动触发回滚机制
2. 环境不可用时，提供手动部署选项

交付物：

- CI/CD 配置文件
- 部署脚本
- 部署文档

6.2.2 集成第三方服务

用例名称：集成第三方服务
行为主体：System 组
前置条件：

- 已确定需要集成的第三方服务
- 已获取相关 API 文档和访问权限

后置条件：

- 第三方服务成功集成到系统中
- 提供统一的接口供其他组使用

基本流程：

1. 评估和选择适合项目需求的第三方服务
2. 实现对第三方 API 的封装
3. 提供统一的接口规范
4. 处理第三方服务异常情况
5. 监控第三方服务的可用性和性能

替代流程：

1. 第三方服务不可用时，提供降级策略

2. 接口变更时，提供兼容性处理

交付物：

- 第三方服务集成文档
- API 封装代码
- 服务配置指南

6.2.3 协同开发数据库

用例名称：协同开发数据库
行为主体：System 组和 Data 组
前置条件：

- 系统需求已明确
- 数据需求已收集

后置条件：

- 数据库架构设计完成
- 数据访问层实现并优化
- 数据版本管理机制建立

基本流程：

1. Data 组设计数据库架构
2. System 组实现数据访问层
3. Data 组管理数据库版本
4. System 组优化数据库性能
5. Data 组实现数据迁移方案

替代流程：

1. 架构设计有问题时，System 组提出修改建议
2. 性能问题严重时，共同重新设计数据模型

职责分工：

- System 组：实现数据访问层和数据库性能优化
- Data 组：负责数据库架构设计、版本管理和数据迁移

交付物：

- 数据库设计文档
- 数据库版本控制脚本
- 数据访问层代码

6.2.4 API 设计与实现

用例名称：API 设计与实现
行为主体：System 组
前置条件：

- 系统功能需求已明确
- 数据模型已定义

后置条件：

- API 接口设计完成并实现
- API 文档生成并发布
- API 测试通过

基本流程：

1. 制定 API 设计规范和标准
2. 实现所有后端 API 功能
3. 生成完整的 API 文档
4. 进行 API 单元测试和集成测试
5. 管理 API 版本和兼容性

替代流程：

1. 需求变更时，更新 API 设计并通知相关方
2. 发现安全漏洞时，及时修复并更新

交付物：

- API 设计文档
- API 实现代码
- API 测试用例
- Swagger/OpenAPI 文档

6.2.5 性能优化

用例名称：性能优化
行为主体：System 组
前置条件：

- 系统基本功能已实现
- 已有初步的性能数据收集

后置条件：

- 系统性能达到预期目标
- 性能监控机制建立

基本流程：

1. 设计并执行性能测试方案
2. 分析并识别系统性能瓶颈
3. 优化代码实现提高执行效率
4. 协同 Data 组进行数据库性能优化
5. 必要时进行系统架构调整
6. 持续监控系统性能指标

替代流程：

1. 性能问题无法通过优化解决时，考虑扩展硬件资源
2. 发现关键路径性能瓶颈时，进行重点优化

交付物：

- 性能分析报告
- 优化实施方案
- 性能基准测试结果

6.2.6 安全保障

用例名称：安全保障
行为主体：System 组
前置条件：

- 系统基本功能已实现
- 已识别系统安全需求

后置条件：

- 系统安全机制建立并生效
- 安全审计和监控机制运行

基本流程：

1. 制定系统安全策略和标准
2. 实现用户身份认证机制
3. 实现细粒度的权限控制
4. 敏感数据加密存储和传输
5. 定期进行安全审计和漏洞扫描
6. 实施安全事件监控和响应机制

替代流程：

1. 发现安全漏洞时，快速响应并修复
2. 遭受攻击时，启动应急响应机制

交付物：

- 安全策略文档
- 安全审计报告
- 安全组件与实践指南

6.2.7 监控与日志系统

用例名称：监控与日志系统
行为主体：System 组
前置条件：

- 系统基本功能已实现
- 已确定监控需求和指标

后置条件：

- 监控与日志系统建立并运行
- 异常告警机制生效

基本流程：

1. 设计系统关键监控指标
2. 实现全面的日志收集机制
3. 构建统一的监控平台
4. 配置合理的告警规则
5. 实现日志聚合和分析功能
6. 创建直观的可视化仪表板

替代流程：

1. 日志量过大时，实施日志分级和采样策略
2. 误报过多时，调整告警阈值和规则

交付物：

- 监控系统配置
- 日志分析平台
- 告警规则配置
- 操作手册

6.3 其他关键用例

本节按需补充其他关键用例的详细说明。

7 数据需求

7.1 数据实体

系统涉及的主要数据实体包括：

1. **用户**：存储系统用户信息
2. **角色**：定义用户角色和权限
3. **业务数据**：[根据具体项目补充]
4. **配置数据**：系统配置和参数
5. **日志数据**：系统操作和事件日志

7.2 数据字典

以下是系统主要数据实体的字段定义：

实体	字段	类型	说明
用户	id	整数	用户唯一标识
	username	字符串	用户名
	password	字符串	加密密码
	email	字符串	用户邮箱
	status	枚举	用户状态 (活跃/禁用)
角色	id	整数	角色唯一标识
	name	字符串	角色名称
	permissions	字符串数组	权限列表

注意：根据具体项目需求补充完整数据字典。

7.3 数据流

系统的主要数据流如下：

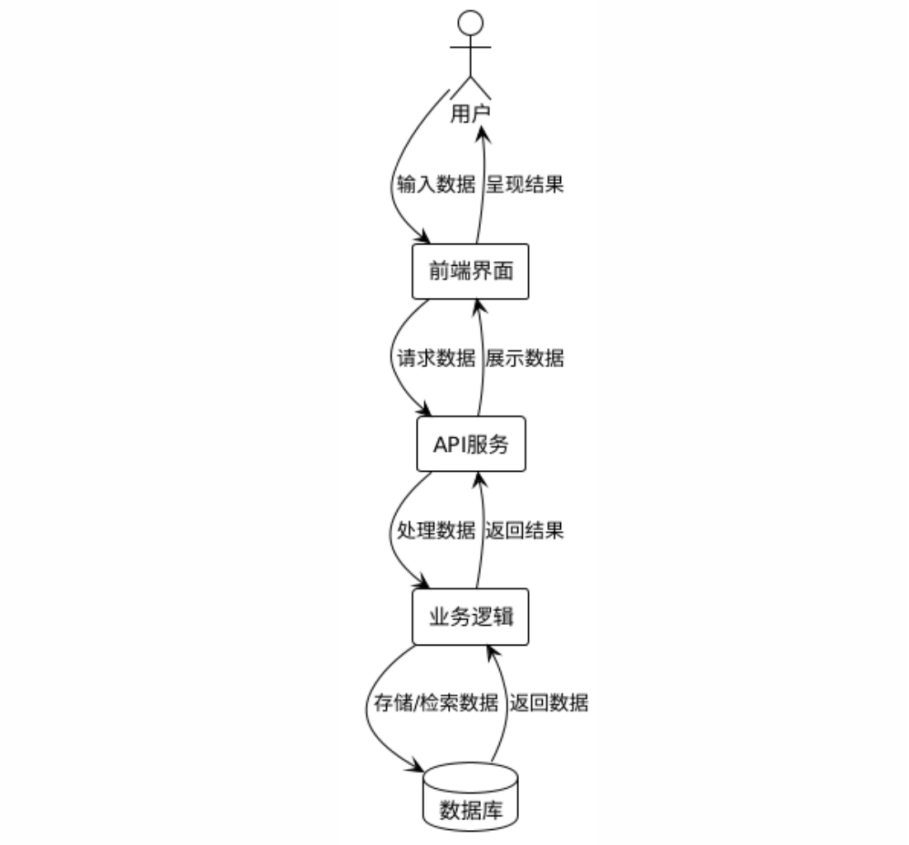

图 10: 数据流图

8 接口需求

8.1 System 组与其他组接口概述

System 组作为项目的技术基础设施提供者，需要与 UI 组、Data 组和 Analysis 组建立明确的接口，确保各组协作顺畅。以下图表展示了 System 组与其他组的主要接口关系：

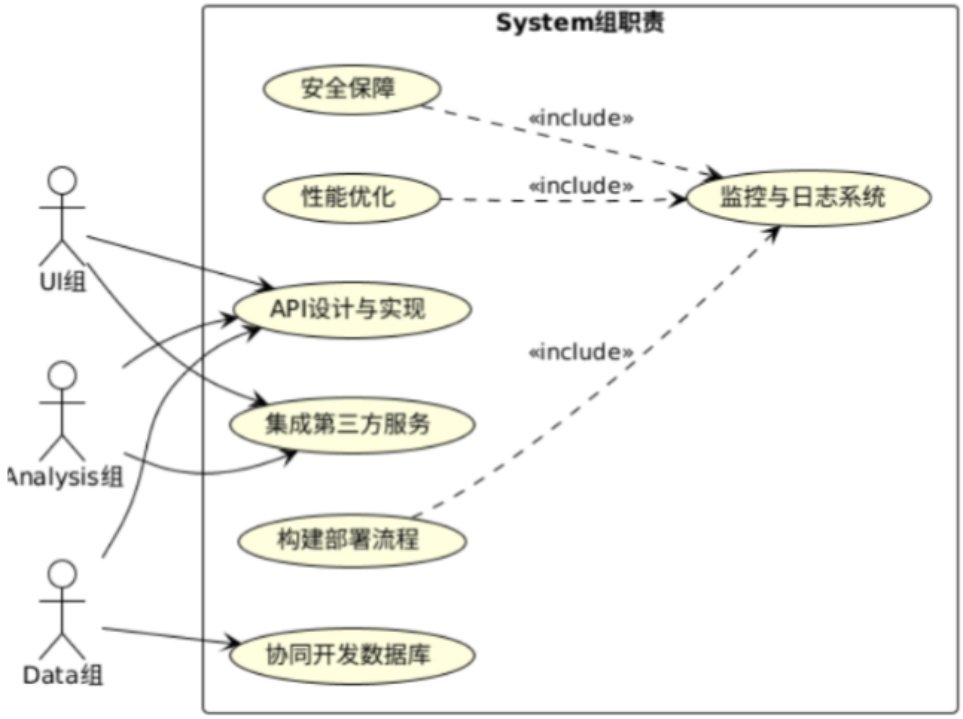

图 11: System 组与其他组的接口关系图

8.2 System 组与 UI 组接口

System 组为 UI 组提供后端服务支持，两组之间的接口主要体现在 API 层面。

8.2.1 API 接口规范

- **接口风格：**RESTful API
- **数据格式：**JSON
- **认证方式：**JWT (JSON Web Token)
- **状态码使用：**遵循 HTTP 标准状态码
- **版本控制：**URL 路径中包含版本号，如/api/v1/users
- **错误处理：**统一的错误响应格式
- **分页机制：**支持 limit/offset 和 cursor 分页

8.2.2 核心 API 接口

接口名称	请求方法	路径	说明
用户认证	POST	/api/v1/auth/login	UI 组调用此接口进行用户登录认证
获取用户信息	GET	/api/v1/users/profile	获取当前登录用户的详细信息
数据查询	GET	/api/v1/data	支持多种查询参数，返回分页数据
数据操作	POST/PUT/DELETE	/api/v1/data	创建、更新、删除数据
文件上传	POST	/api/v1/files	支持多文件上传，返回文件 URL
系统配置	GET	/api/v1/config	获取 UI 所需的系统配置参数

8.2.3 前后端交互流程

1. **用户认证流程**
   - UI 组收集用户凭证 (用户名/密码)
   - 调用认证 API 获取访问令牌
   - 后续请求携带令牌进行身份验证
2. **数据交互流程**
   - UI 组按照 API 规范构造请求
   - System 组处理请求并返回适当的响应
   - UI 组根据响应更新界面
3. **错误处理流程**
   - System 组返回标准错误码和详细错误信息
   - UI 组根据错误类型显示适当的错误提示

8.3 System 组与 Data 组接口

System 组与 Data 组在数据库开发和数据处理方面紧密协作，接口设计需要考虑数据访问效率和安全性。

8.3.1 数据库访问接口

- **数据访问模式：**Repository 模式
- **ORM 框架：**提供对象关系映射能力
- **事务支持：**支持分布式事务
- **缓存机制：**多级缓存策略
- **连接池管理：**优化数据库连接资源

8.3.2 主要数据交互场景

交互场景	接口方法	说明
数据查询	findById(), findByCondition()	支持单条记录查询和条件查询，Data 组负责 SQL 优化
数据写入	save(), update(), delete()	提供统一的数据写入接口，System 组处理并发控制
批量操作	batchInsert(), batchUpdate()	高性能批量数据处理接口
数据迁移	migrateData()	版本升级时的数据迁移接口
数据验证	validate()	数据业务规则验证，确保数据一致性

8.3.3 职责分工

- **Data 组职责：**
  - 设计数据库结构（表、索引、约束）
  - 优化数据库查询性能
  - 编写数据库变更脚本
  - 管理数据库版本
- **System 组职责：**
  - 实现数据访问层代码
  - 处理数据业务逻辑
  - 确保数据访问安全
  - 实现数据缓存机制
  - 管理数据库连接资源

8.4 System 组与 Analysis 组接口

System 组为 Analysis 组提供数据服务和计算资源，支持 Analysis 组进行各类数据分析。

8.4.1 数据分析 API

- **数据获取 API：**提供结构化数据查询接口
- **数据流 API：**支持流式数据处理

- **计算任务 API**：支持异步分析任务提交和结果获取
- **结果存储 API**：提供分析结果持久化能力

8.4.2 主要交互场景

交互场景	接口方法	说明
原始数据获取	fetchRawData()	获取指定条件的原始数据，支持分页和筛选
提交分析任务	submitTask()	提交异步分析任务，返回任务 ID
获取任务状态	getTaskStatus()	查询分析任务的执行状态
获取分析结果	getTaskResult()	获取已完成任务的分析结果
注册数据回调	registerCallback()	分析结果生成后自动通知 Analysis 组

8.4.3 数据格式规范

- **输入数据格式**：JSON、CSV 或二进制格式
- **元数据描述**：提供数据结构和字段描述
- **输出结果格式**：统一的分析结果格式
- **错误信息格式**：详细的错误码和描述

8.5 跨组协作接口

某些功能需要多个组共同协作，System 组需要提供协调多组工作的接口。

8.5.1 通知与事件系统

- **事件发布接口**：System 组发布系统事件
- **事件订阅接口**：其他组订阅感兴趣的事件
- **消息队列**：确保事件异步处理不阻塞主流程
- **事件类型**：系统状态变更、任务完成、数据更新等

8.5.2 集成测试接口

- **测试环境配置**：提供独立测试环境接口
- **测试数据生成**：生成测试数据的接口
- **状态重置**：测试后恢复系统状态的接口
- **模拟接口**：模拟第三方服务的接口

8.6 接口文档与版本管理

8.6.1 接口文档规范

- **文档格式：**采用 OpenAPI(Swagger) 规范
- **必要字段：**接口 URL、请求方法、参数说明、响应格式、错误码
- **示例代码：**各语言的调用示例
- **更新记录：**接口变更历史

8.6.2 接口版本管理

- **版本命名：**主版本. 次版本. 修订版本（如 1.2.3）
- **兼容性原则：**次版本和修订版本必须向后兼容
- **废弃流程：**接口废弃前需要提前公告，保留过渡期

8.6.3 接口变更管理

- **变更评审：**重要接口变更需多组评审
- **变更通知：**接口变更前通知相关方
- **变更测试：**新接口必须通过自动化测试
- **回滚机制：**接口上线后发现问题可快速回滚

9 约束与假设

9.1 技术约束

1. 系统应使用现代 Web 技术开发
2. 数据库选择应考虑性能和可扩展性
3. 系统应支持主流浏览器
4. 开发应遵循安全编码规范

9.2 业务约束

1. 系统应符合相关行业法规和标准
2. 数据处理应遵循数据保护原则
3. 系统功能应与现有业务流程兼容

9.3 假设条件

1. 用户具备基本的计算机操作技能
2. 系统运行环境满足最低硬件要求
3. 网络连接稳定可靠
4. 用户数据可从现有系统迁移

10 验收标准

10.1 功能验收标准

系统验收应满足以下功能标准：

1. 用户可完成所有核心业务流程
2. 数据处理结果准确无误
3. 报表生成功能正常运行
4. 系统管理功能完备可用

10.2 非功能验收标准

系统还应满足以下非功能标准：

1. 系统响应时间满足性能需求
2. 安全测试无严重漏洞
3. 系统可靠性测试通过
4. 用户界面符合可用性原则

11 附录

11.1 术语表

术语	定义
API	应用程序编程接口 (Application Programming Interface)
UI	用户界面 (User Interface)
ETL	提取、转换、加载 (Extract, Transform, Load)
CI/CD	持续集成/持续部署 (Continuous Integration/Continuous Deployment)
RESTful	表现层状态转移 (Representational State Transfer) 架构风格的 API 设计原则

11.2 用例关系说明

- **包含关系 («include»)**：表示一个基础用例被包含在另一个用例中，是必须执行的部分
- **扩展关系 («extend»)**：表示一个用例在特定条件下扩展另一个用例的行为
- **使用关系**：表示角色使用某个用例提供的功能
- **参与关系**：表示角色参与某个用例的定义或讨论，但不是主要实施者

11.3 修订历史

版本	日期	修订人	修订内容
1.0	2025 年 3 月 31 日	蔡旭	文档初稿

表 6: 文档修订历史