数据分析智能体多端开发概要设计说明书

1. 引言

1.1 编写目的

本概要设计说明书是《数据分析智能体多端开发》项目从需求分析到详细设计的关键衔接文档，旨在明确系统的总体架构、模块划分、数据模型及核心接口设计。其核心目标是为开发团队提供统一的设计基准——开发人员可依据此文档进行模块开发与接口对接，测试人员可据此制定测试方案，同时也为课程指导教师提供项目设计合理性的评估依据。文档覆盖系统设计的核心要素，确保 8 周内完成 MVP 版本开发，满足高校学生基础数据分析场景的需求。

1.2 项目背景与范围

项目背景：当前高校学生在课程作业、学科竞赛及学生工作中面临数据分析工具操作复杂（如 SPSS、Python 库学习成本高）、跨端协作不便（数据与分析结果难以在手机/电脑间同步）、商业工具成本高等问题。结合开源大模型（Phi-3-mini）的自然语言交互能力与跨端开发技术，开发一款轻量化、易操作的数据分析智能体成为解决上述痛点的有效方案。

项目范围：本项目面向本校学生，开发支持网页端、移动端 APP（Android/iOS）、桌面软件（Windows/macOS）的多端应用，核心功能包括：自然语言驱动的数据分析需求解析、多工具代码生成（Python/SPSS/R）、基础数据可视化（折线图/柱状图/饼图/散点图）、多端数据同步。技术栈限定为 Flutter（移动端/网页端）+Electron（桌面端）+Flask（后端）+MySQL（数据库）+Phi-3-mini（大模型），暂不支持复杂机器学习建模、动态反爬爬虫及多用户协作功能，8 周内完成 MVP 开发与交付。

1.3 术语与缩略语

术语/缩写	含义说明
MVP	最小可行产品，指包含核心功能、可满足用户基本需求的产品版本
跨端框架	Flutter（用于移动端/网页端 UI 开发）、Electron（用于桌面端封装网页端代码）
Phi-3-mini	微软开源的小型语言模型，用于自然语言需求解析与代码生成
双模式输	系统支持“仅生成可复用代码”和“生成代码+直接输出分析结论”两

出	种模式
RESTful API	基于 REST 架构风格的应用程序接口，用于前后端数据交互

1.4 参考资料

1.《数据分析智能体多端开发需求规格说明书》v1.2（2025 年 12 月）

2.Flutter 官方文档：https://docs.flutter.dev/

3.Electron 开发指南：https://www.electronjs.org/docs

4.Phi-3-mini 部署手册（微软开源社区）

5.MySQL 8.0 数据库设计规范

6.《软件工程：实践者的研究方法》（Roger S. Pressman 著）

2. 总体设计概述

2.1 设计目标与原则

2.1.1 设计目标

• 功能完整性：覆盖“数据输入-需求解析-代码生成-可视化-同步”核心流程，满足用户基础数据分析需求；

• 架构合理性：采用分层解耦设计，确保模块间低耦合、高内聚，便于团队分工开发（如前端开发、后端接口开发、AI 模块开发）；

• 性能适配性：在学生常用硬件环境下（如笔记本电脑、中端手机），保证接口响应时间≤3 秒，代码生成时间≤10 秒；

• 可扩展性：预留功能扩展接口（如新增图表类型、支持更多分析工具），便于后续迭代；

• 开发可行性：技术选型匹配学生开发能力，确保 8 周内可完成 MVP 版本开发与测试。

2.1.2 设计原则

• 用户中心原则：架构设计优先适配学生使用习惯，简化操作流程（如自然语言交互替代复杂参数配置）；

• 分层架构原则：严格区分表示层（UI）、业务逻辑层（Service）、数据访问层（DAO）、AI 能力层，避免模块职责混淆；

• 开源优先原则：核心组件均采用开源技术（如 Phi-3-mini、Flutter），降低开发成本与版权风险；

• 安全性原则：设计代码执行沙箱、用户身份认证、数据加密存储等机制，保

障系统与用户数据安全；

• 一致性原则：多端 UI 风格、数据格式、接口协议保持一致，提升用户体验。

2.2 系统运行环境

环境类型	具体配置要求
后端服务环境	操作系统：Ubuntu 22.04 LTS；CPU：≥2 核；内存：≥8GB；数据库：MySQL 8.0；Python：3.9+
开发环境	开发工具：VS Code（配 Flutter/Electron 插件）、PyCharm；版本控制：Git；接口测试：Postman；数据库管理：Navicat
客户端环境	网页端：Chrome 95+/Edge 95+/Safari 15+；移动端：Android 10.0+/iOS 14.0+；桌面端：Windows 10+/macOS 11+

2.3 技术路线与架构风格

系统采用“四层架构+跨端混合模式”设计，整体架构如图 1 所示（文字描述）：

```
数据分析智能体多端系统架构
├– 表示层（UI 层）：负责用户交互与界面展示
|  ├– 移动端 APP（Flutter 开发，共享 UI 组件库）
|  ├– 网页端（Flutter Web 编译，适配主流浏览器）
|  └– 桌面端（Electron 封装网页端，支持 Windows/macOS）
├– 业务逻辑层（Service 层）：核心业务处理中心
|  ├– 用户管理、数据输入、智能分析、可视化、多端同步、辅助功能模块
|  └– 统一接收客户端请求，分发至对应模块处理
├– AI 能力层：提供大模型与代码处理能力
|  ├– Phi-3-mini 模型本地化部署（量化优化）
|  ├– Prompt 模板库（需求解析、代码生成模板）
|  └– 代码执行沙箱（隔离执行环境，安全校验）
└– 数据存储层：负责数据持久化与缓存
   ├– MySQL 数据库（用户数据、分析记录、同步信息）
   └– 本地缓存（SQLite，移动端/桌面端离线数据）
```

架构交互流程：客户端通过 RESTful API 向业务逻辑层发起请求（如“上传数据并分析”），业务逻辑层调用数据输入模块解析数据，再将数据与用户需求传入 AI 能力层生成代码与结论，可视化模块基于分析结果生成图表，最后通过多端同步模块实现数据跨端共享，所有核心数据通过数据存储层持久化。

3. 系统总体结构设计

3.1 系统功能模块划分

基于需求规格说明书，系统业务逻辑层划分为 6 个核心模块，各模块功能边界清晰，通过接口实现通信。模块划分如图 2 所示（文字描述）：

```
业务逻辑层核心模块
├─ 用户管理模块（M01）：注册、登录、信息维护、权限校验
├─ 数据输入模块（M02）：文件上传解析（CSV/Excel）、联网检索、基础爬虫
├─ 智能分析模块（M03）：需求解析、Phi-3-mini 调用、代码生成、代码执行、结论整理
├─ 可视化模块（M04）：基础图表生成、图表导出（PNG）
├─ 多端同步模块（M05）：增量同步、冲突处理、同步状态管理
└─ 辅助功能模块（M06）：分析报告导出（Word）、历史记录查询、操作日志
```

3.2 核心模块详细说明

模块名称	模块编号	核心职责	关键输入/输出	约束条件
用户管理模块	M01	用户身份认证与信息管理	输入：注册信息、登录凭证；输出：token、用户信息	仅支持校园邮箱注册；密码 MD5 加密存储
数据输入模块	M02	多源数据采集与格式化	输入：上传文件、检索关键词、网页 URL；输出：标准化 JSON 数据	文件≤10MB；爬虫仅限静态网页；检索调用公开接口
智能分析模块	M03	自然语言需求转分析结果	输入：标准化数据、分析需求；输出：代码、结论、操作步骤	支持 Python/SPSS/R 代码；拒绝预测类复杂需求
可视化模块	M04	分析结果图表化展示与导出	输入：分析结果、图表类型；输出：图表数据、PNG 文件	支持折线/柱状/饼图/散点图；导出分辨率≥1080×720px
多端同步模块	M05	跨端数据一致性维护	输入：设备信息、同步时间戳；输出：同步结果、冲突提示	基于时间戳增量同步；冲突时用户手动选择策略

辅助功能模块	M06	分析成果归档与追溯	输入：分析记录 ID；输出：Word 报告、历史记录列表	报告含固定章节结构；日志保留最近 3 个月

3.3 模块间交互关系

核心模块间通过“请求-响应”模式交互，关键交互流程如下：

• **数据分析流程：**用户管理模块（登录验证）→数据输入模块（数据采集）→智能分析模块（代码生成与执行）→可视化模块（图表生成）→辅助功能模块（报告导出）；

• **多端同步流程：**客户端（触发同步）→多端同步模块（增量数据查询）→数据存储层（云端数据对比）→多端同步模块（冲突处理）→客户端（同步结果展示）；

• **AI 能力调用流程：**智能分析模块（需求格式化）→AI 能力层（Prompt 填充与模型调用）→智能分析模块（代码解析与结论整理）→客户端（结果返回）。

4. 数据设计

4.1 数据概念模型（ER 图）

系统核心数据实体包括：用户（User）、分析记录（AnalysisRecord）、原始数据（RawData）、生成代码（GeneratedCode）、可视化结果（VisualResult）、同步记录（SyncRecord）、设备信息（Device）。各实体间关系如下：

• 1 对 N 关系：1 个用户（User）可拥有多个分析记录（AnalysisRecord）、多个同步记录（SyncRecord）、多个绑定设备（Device）；

• 1 对 1 关系：1 个分析记录（AnalysisRecord）对应 1 份原始数据（RawData）、1 套生成代码（GeneratedCode）、1 份可视化结果（VisualResult）。

ER 图核心结构可概括为：User← AnalysisRecord → RawData/GeneratedCode/VisualResult；User ← SyncRecord/Device。

4.2 数据库表结构设计

系统采用 MySQL 8.0 作为核心数据库，共设计 7 张核心表，表结构如下（关键字段）：

4.2.1 users（用户表）

字段名	类型	约束	说明

id	INT	PK、AI	用户唯一标识
email	VARCHAR(100)	UNIQUE、NOT NULL	校园邮箱（登录账号）
username	VARCHAR(50)	NOT NULL	用户名
password	VARCHAR(32)	NOT NULL	MD5 加密后的密码
create_time	DATETIME	NOT NULL	注册时间

4.2.2 analysis_records（分析记录表）

字段名	类型	约束	说明
id	INT	PK、AI	分析记录唯一标识
user_id	INT	FK→users.id、NOT NULL	关联用户 ID
raw_data_id	INT	FK→raw_datas.id、NOT NULL	关联原始数据 ID
code_id	INT	FK→generated_codes.id、NOT NULL	关联生成代码 ID
visual_id	INT	FK→visual_results.id、NOT NULL	关联可视化结果 ID
requirement	TEXT	NOT NULL	用户分析需求文本
create_time	DATETIME	NOT NULL	分析创建时间
sync_status	TINYINT	NOT NULL	0=未同步，1=已同步

4.2.3 其他核心表简述

• raw_datas（原始数据表）：存储标准化后的原始数据（JSON 格式），含 data_source（数据来源：file/search/crawl）、data_size（数据大小）等字段；

• generated_codes（生成代码表）：存储 Python/SPSS/R 代码及操作步骤，字段包括 python_code、spss_code、r_code、operation_steps；

• visual_results（可视化结果表）：存储图表数据（ECharts 格式）与导出信息，

含 chart_type（图表类型）、chart_data（图表数据）、export_path（导出路径）；

• sync_records（同步记录表）：记录多端同步详情，含 device_id（设备 ID）、sync_status（同步状态）、last_sync_time（最后同步时间）；

• devices（设备表）：存储用户绑定设备信息，含 device_id（设备唯一标识）、device_type（设备类型）、bind_time（绑定时间）。

4.3 数据存储策略

系统采用“**云端+本地**”混合存储策略:

• 云端存储：核心数据（用户信息、分析记录、代码、可视化结果）存储于 MySQL 数据库，确保多端同步时的数据一致性；

• 本地存储：移动端/桌面端采用 SQLite 存储离线数据（如最近访问的分析记录、缓存的图表），提升离线使用体验，联网后自动同步至云端；

• 数据安全：敏感数据（如密码）采用 MD5 加密存储，原始数据与分析结果仅对所属用户可见，通过用户 token 控制数据访问权限。

5. 接口设计

5.1 接口设计原则

系统接口采用 RESTful 风格设计，遵循以下原则：

• 资源导向：以数据资源为核心，URL 命名体现资源属性（如/api/v1/users 表示用户资源）；

• HTTP 方法语义：GET（查询）、POST（创建）、PUT（更新）、DELETE（删除），避免语义混淆；

• 统一响应格式：所有接口返回 JSON 格式数据，包含 code（状态码）、msg（提示信息）、data（业务数据）、requestId（请求唯一标识）；

• 版本控制：URL 中包含版本号（如 v1），便于后续接口迭代与兼容性维护；

• 权限校验：除登录/注册接口外，其余接口均需在请求头携带 token 进行身份验证。

5.2 核心模块接口详情

5.2.1 用户管理模块（M01）

接口名	URL	请求方	请求参数（JSON）	响应参数（JSON）	说明

称		法			

用户注册	/api/v1/users/register	POST	{
			"email":"student@school.edu",
			"username":"张三",
			"password":"12345678"
			}
				{
				"code":200,
				"msg":"注册成功",
				"data":{
				"userId":1001,
				"token":"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."},
				"requestId":"20251205090001"
				}
					email需为校园邮箱格式，密码长度≥8位，注册成功后直接返回token

用户登录	/api/v1/users/login	POST	{
			"email":"student@school.edu",
			"password":"12345678"
			}
				{
				"code":200,
				"msg":"登录成功",
				"data":{
				"userId":1001,
				"username":"张三",
				"token":"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."},
				"requestId":"20251205090002"
					token有效期为7天

				}	，过期后需重新登录

5.2.2 数据输入模块（M02）

接口名称	URL	请求方法	请求参数	响应参数（JSON）	说明
文件上传解析	/api/v1/data/upload	POST	form-data 格式：file（文件，CSV/Excel），fileName（文件名）	{ "code":200, "msg":"解析成功", "data":{ "rawDataId":2001, "dataPreview":[{"name":"张三","math":85},...]}, "requestId":"20251205090003" }	文件大小≤10MB，支持 CSV、Excel（.xlsx/.xls）格式，返回前 5 条数据预览

6. 系统安全性设计

6.1 数据安全

• 敏感数据加密：用户密码采用 MD5 加盐加密存储，避免明文泄露；token 采用 JWT 算法生成，包含用户 ID 与有效期信息；

• 数据访问控制：基于用户 ID 与 token 进行数据权限校验，确保用户仅能访问自身创建的分析记录与数据；

• 数据备份策略：MySQL 数据库每日凌晨自动备份，备份文件保留最近 7 天，

防止数据丢失。

6.2 应用安全

• 代码执行沙箱：智能分析模块生成的代码在隔离沙箱环境中执行，禁止文件读写、网络请求、系统命令调用等危险操作；

• 接口防攻击：登录/注册接口限制单 IP 每分钟请求次数≤10 次，防止暴力破解；所有接口均校验请求头 token，拒绝非法请求；

• 输入校验：客户端与服务端双重校验用户输入（如文件格式、邮箱格式、密码强度），过滤恶意输入内容。

7. 开发与测试计划

7.1 开发进度计划

项目周期为 8 周，分 4 个阶段推进：

第 1-2 周（需求确认与基础搭建）：完成需求评审、技术栈搭建（Flutter 项目初始化、Flask 后端框架搭建、MySQL 数据库创建）、核心表结构创建；

第 3-4 周（核心模块开发）：开发用户管理模块、数据输入模块、智能分析模块（基础代码生成功能）；

第 5-6 周（功能扩展与多端适配）：开发可视化模块、多端同步模块、辅助功能模块，完成移动端/网页端/桌面端 UI 适配；

第 7-8 周（测试与优化）：进行功能测试、性能测试、兼容性测试，修复 bug 并优化用户体验，最终交付 MVP 版本。

7.2 测试计划

测试分为 3 个层面，确保系统质量：

• 单元测试：针对各模块核心函数（如数据解析函数、代码生成函数）编写测试用例，覆盖率≥80%；

• 接口测试：使用 Postman 测试所有 RESTful 接口，验证参数合法性、响应格式正确性、异常处理能力；

• 用户验收测试：邀请 10-15 名本校学生进行实际使用测试，收集反馈并优化功能与 UI。

8. 结论

本概要设计说明书通过对数据分析智能体多端系统的总体架构、模块划分、数据模型、接口设计等核心内容的详细阐述，为项目开发提供了清晰的技术蓝图。

系统采用分层架构与跨端混合模式，平衡了功能完整性、开发可行性与用户体验，能够有效解决高校学生数据分析过程中的工具操作复杂、跨端协作不便等痛点。

后续开发团队将严格依据本设计文档推进工作，确保 8 周内完成 MVP 版本交付。同时，文档预留了功能扩展接口，为后续支持更多分析工具、图表类型及复杂分析需求奠定了基础。建议开发过程中定期进行设计评审与进度复盘，及时调整方案以应对潜在风险。