AI-对话驱动流程自执行系统-架构设计初稿（侧重前端）

1. 文档概述

1.1 文档目的

本文档旨在设计一个AI对话驱动的流程执行系统，通过三栏布局实现从**理解用户意图**--到-->**确认流程**--到-->**流程执行**的完整闭环。

1.2 系统背景

基于AI能力的营销流程自动化系统，支持自然语言交互、流程节点执行和结果自定义渲染。

1.3 术语定义

WorkFlow: 流程

Node: 流程上的单个节点

Execution: 单个节点执行

2. 系统概述

2.1 核心功能

- 自然语言对话交互
- 流程节点动态渲染
- 执行结果自定义展示
- 多轮对话优化结果

2.2 技术特点

- 三栏布局设计
- 实时双向通信
- 组件化架构
- 状态驱动渲染

3. 需求分析

3.1 需求描述

a. **用户对话输入**

- 创建一个对话输入组件，允许用户通过自然语言输入需求。
- 利用 服务端接口通信 请求对应大模型。

b. **流程节点渲染**

- 用户输入经过解析后，生成相应的流程节点并显示在中间栏。
- 使用可配置的组件库，根据节点类型动态渲染节点内容。

c. **执行过程管理**

- 在右侧创建一个执行过程响应区，该区域监听节点的执行结果。
- 根据不同节点类型渲染自定义内容，动态更新执行结果。

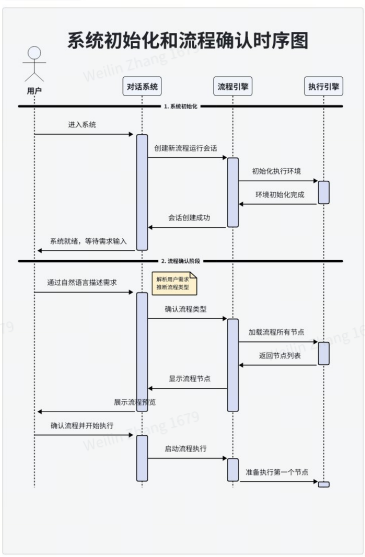
d. **节点间交互处理**

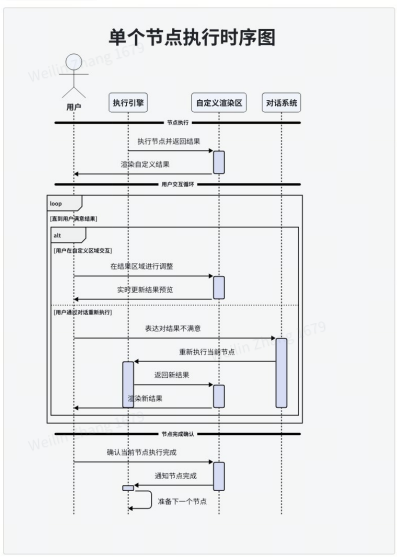
- 引入一个任务队列机制，管理节点执行与交互结果的顺序和逻辑。
- 节点执行完成后，结果传递到下一个节点，并通过再度解析用户输入来触发新的执行。

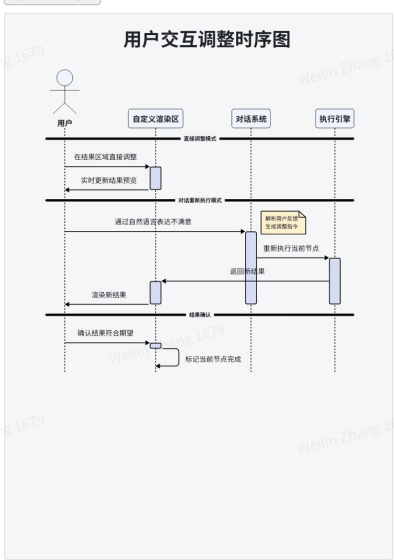
e. **用户交互与节点更新**

- 允许用户选中节点，并在左侧面板通过自然语言接口与选中节点交互。
- 将交互结果返回到对应节点，重新触发节点的执行与渲染。

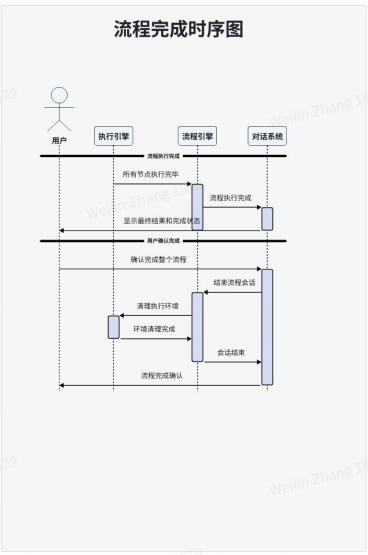
3.2 用户旅程时序图



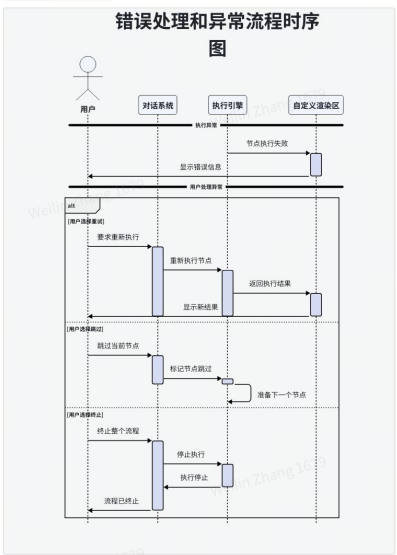




流程完成



错误处理和异常



3.3 功能需求

- 对话交互需求：支持自然语言输入，多轮对话确认
- 流程管理需求：动态生成流程节点，支持节点选择
- 执行展示需求：自定义渲染执行结果，支持交互调整
- 用户交互需求：支持结果调整和重新执行

3.4 非功能需求

- 监控需求

4. 系统架构设计

4.1 设计理念

抽象	扩展	稳定

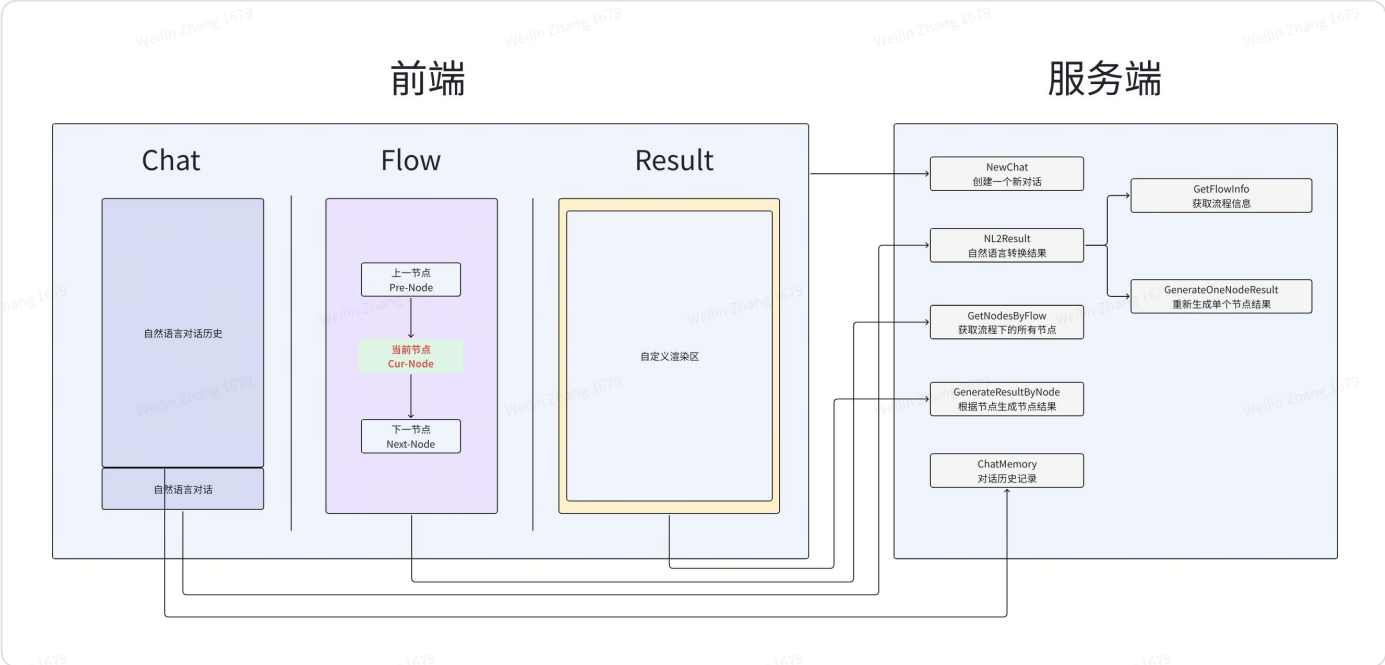
将各模块进行抽象化，后续任何流程均可以使用该系统进行运行和建设。

为了保证可扩展性和业务适应性，右侧区域使用自定义渲染，通过props对齐规范，具体流程节点可自定义渲染该节点右侧区域内容。

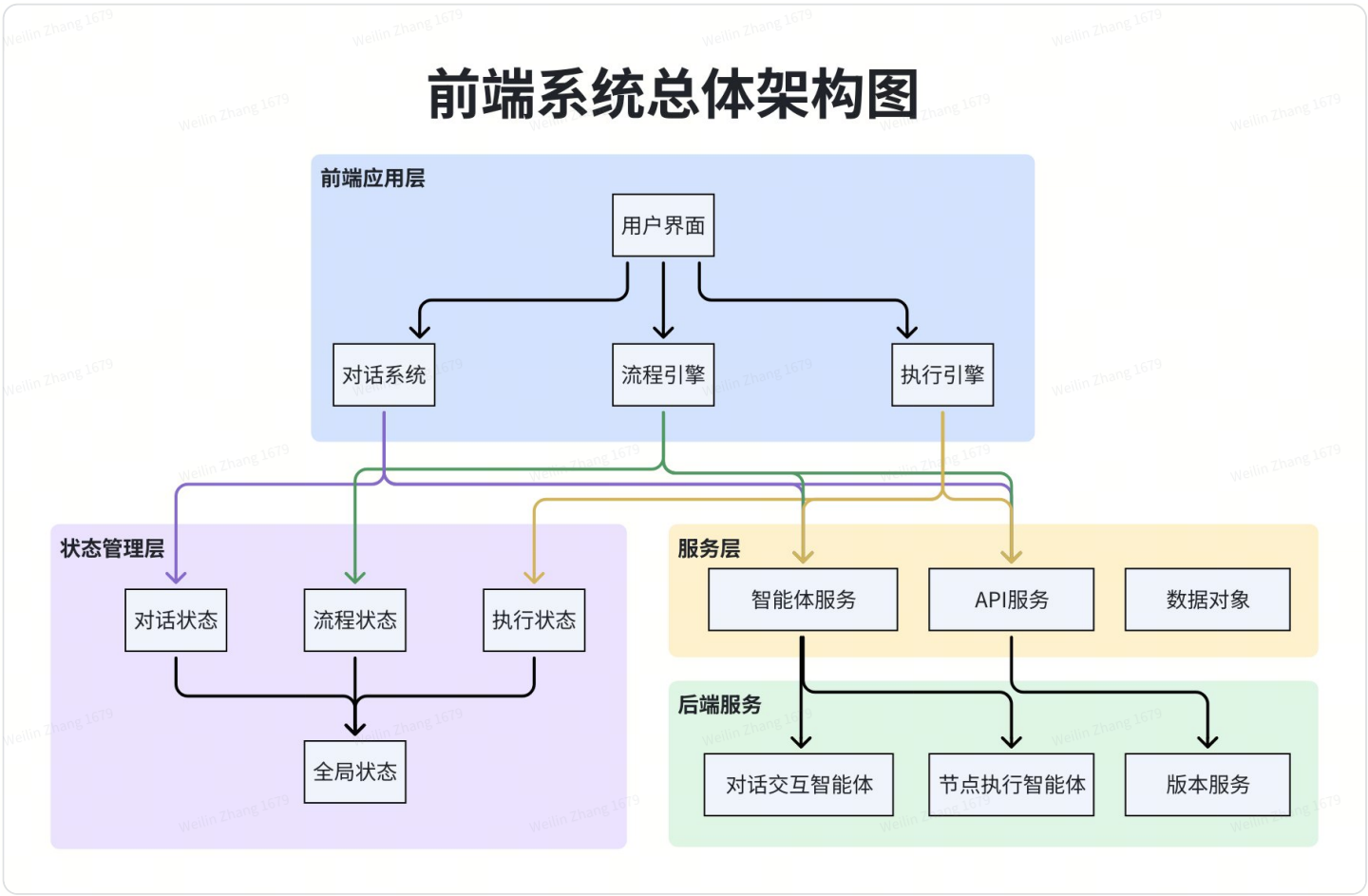
为保证所有用户的教育成本一致、系统体验稳定，将**AI对话**、**视觉设计**、**结果**（进行迭代后稳定化

PS：迭代后可沉淀归一结果展示、交互模板

4.2 抽象设计



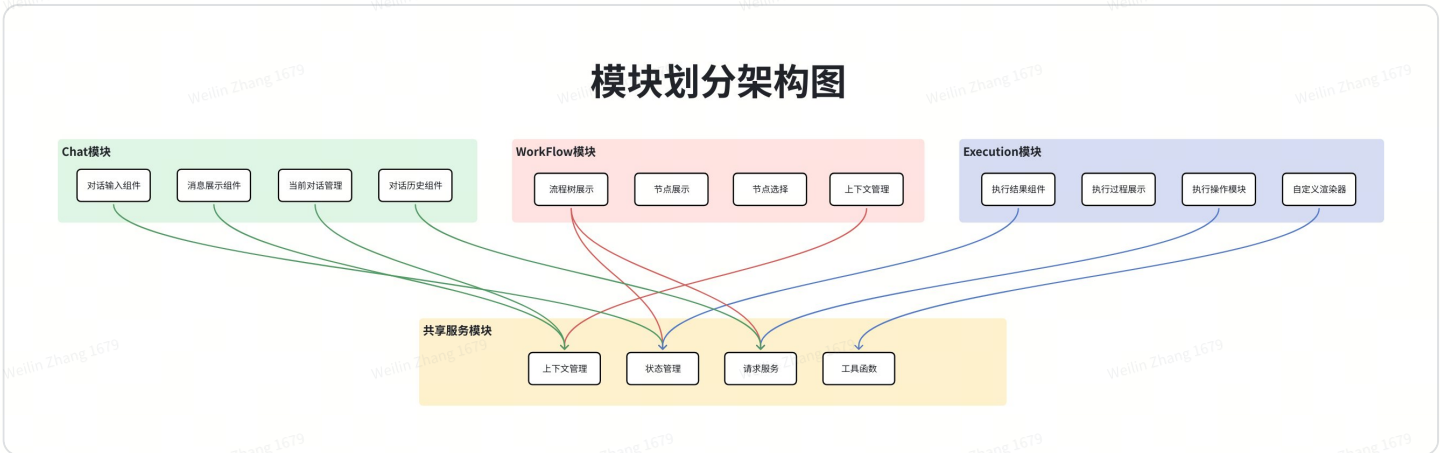
4.3 总体架构



4.4 技术架构

- 前端框架：React + TypeScript
- 状态管理：Context API + Jotai
- 组件库：AIOS-Design + AntDesign
- 样式方案：CSS-in-JS
- 通信方式：SSE + HTTP API (Axios)

5. 模块划分设计



5.1 核心模块

- Chat模块：对话交互、消息处理、上下文管理
- Workflow模块：流程渲染、节点管理、流程导航
- Execution模块：执行展示、交互处理、结果渲染

5.2 模块职责边界

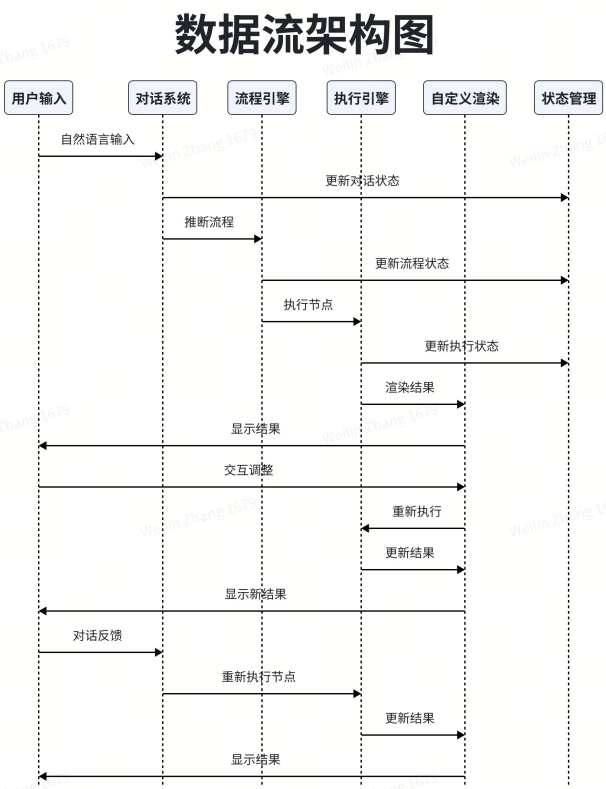
- Chat模块：负责用户输入和AI对话交互
- Workflow模块：负责流程节点展示和选择
- Execution模块：负责执行结果展示和用户交互

6. 数据流设计

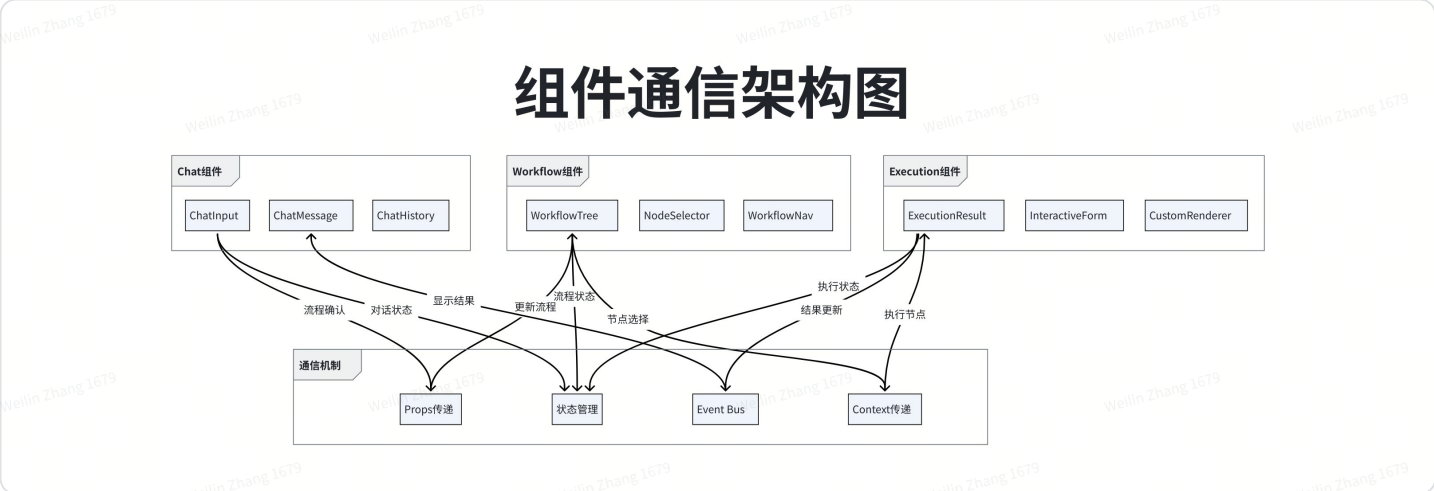
6.1 数据流架构

代码块

```
用户输入 → 对话系统 → 流程引擎 → 执行引擎 → 自定义渲染
    ↑                                    ↓
    └──────────── 用户交互反馈 ←──────────┘
```



6.2 组件通信数据流



6.3 状态管理设计

- **全局状态**：流程状态、会话状态、用户状态
- **局部状态**：节点状态、对话状态、渲染状态
- **持久化**：会话历史、流程历史、用户偏好

7.

8. 用户界面设计

8.1 交互设计图

8.2 三栏布局设计

- **左侧**：对话交互区域
- **中间**：流程节点展示区域
- **右侧**：执行结果展示区域

8.3 交互设计

- **对话交互**：支持多轮对话，上下文管理
- **流程导航**：支持节点选择，流程跳转
- **结果调整**：支持直接调整和对话调整

---

9. 技术实现

9.1 项目结构

代码块

```
ai-workflow-running-frontend/
├── src/
│   ├── components/
│   │   ├── Chat/                    # 对话组件
│   │   ├── Workflow/                # 流程组件
│   │   └── Execution/               # 执行组件
│   ├── pages/
│   │   └── MainPage/                # 主页面
│   ├── store/                       # 状态管理
│   ├── services/                    # 服务层
│   ├── types/                       # 类型定义
│   └── utils/                       # 工具函数
├── .claude/                          # 【AI】Claude配置
│   │   ├── commands/
│   │   └── skills/
│   │   └── agents/
│   │   └── rules/
│   │   └── settings.json
├── package.json
├── tsconfig.json
└── README.md
└── CLAUDE.md                        # 【AI】Claude全局上下文
```

9.2 智能体集成

- **流程确认智能体**：GetWorkFlowByNL
- **节点执行智能体**：由IAM系统承接
- **对话交互智能体**：支持多轮对话优化

---

10. 待解决问题

10.1 对话功能边界

- 初始化对话的多轮确认机制
- 单节点执行时的调整流程
- 对话历史的上下文关系

10.2 流程确认机制

- 流程确认后的执行触发
- 流程结束的确认方式
- 执行历史的记录需求

11. 风险评估

11.1 技术风险

- 多轮对话的上下文管理复杂性
- 自定义渲染的灵活性

11.2 解决方案

- 采用状态机管理对话流程
- 设计可扩展的渲染器架构

12. 排期

https://applink.feishu.cn/client/todo/task_list?guid=59b415fd-ac41-4d2f-93b4-68a5a7c98ce9