ChatGPT

系统整体架构设计

本系统采用典型的前后端分离架构：前端使用现代 JavaScript 框架（如 React 或 Vue）构建用户界面，后端使用 Python（Flask 或 FastAPI）处理业务逻辑并调用大语言模型 API。前端通过 RESTful/JSON 接口与后端通信，可选使用 WebSocket 实现实时推送。后端则负责维护用户日程数据（如活动表、任务表等），并调用 LLM（如 DeepSeek R1）完成自然语言解析与规划。对于后端设计，可采用微服务架构以提高可扩展性，数据库可选用关系型（MySQL/PostgreSQL）或非关系型（MongoDB）存储用户日程和偏好，并部署到云服务（如 AWS、GCP） [1] [2] 。前端界面主要包括文本输入组件、可交互的日历视图和任务列表组件，所有模块共同提供统一的调度助手功能。

日程建模与冲突判断逻辑

系统内部对日程进行结构化建模，每个日程项包含开始时间、结束时间、标题/描述等字段，可存储于数据库或采用 iCalendar（ICS）格式管理。日历中已有的固定安排（如课程、会议）视为不可变时间段；新任务插入时只考虑可用空闲段。冲突判断采用典型的时间区间重叠检测：若新任务的开始时间早于已存在日程的结束时间且结束时间晚于其开始时间，则视为冲突。实现上可使用开源库辅助，例如 Python 的 `icalendar` 库（兼容 iCalendar 标准）可创建和解析日历信息 [3] ，也可直接在数据库中查询重叠记录或调用类似算法。任务存储建议分为“固定事件”（固定在具体时间）和“灵活任务”（可在空闲段安排，包含预估时长、优先级等），后续调度时分别处理。

用户状态判断与任务优先级调整策略

系统需考虑用户的个人习惯和当前状态。例如，下课后可能需要休息，中午或晚饭时间不可安排任务，睡眠时间固定为不可用段。用户可设置或系统学习用户常规偏好（如每天 18:00 晚饭、中午休息等），并将这些时段标记为不可用。同时，任务可分配优先级或紧急度标签。AI 会根据**任务的重要性/紧急程度**和用户日程习惯来排序和决策安排。例如，研究表明 AI 日历可以分析用户的日程习惯和任务优先级，自动优化安排并避免冲突 [4] ；工具如 Motion、Reclaim AI 等会根据历史日程预测任务时长并按用户设定的目标/优先级调整日程，确保重要任务不被忽视 [4] [5] 。具体策略上，可按照紧急重要矩阵或先来后到规则：若一个高优先级任务与低优先级任务冲突，则优先保留高优先级；若多个可弹性任务冲突，可提示用户调整。无合适空闲时间时，AI 应生成备选方案（如推迟到翌日、建议晚间加班等）并征求用户确认。

前端组件结构与界面布局

前端建议分为以下主要组件：
- **聊天/输入区**：用于与 AI 文本交互，可采用聊天界面组件，如 Assistant UI [6] 或 `react-chat-ui` [7] 等开源库，它们提供了消息气泡、输入框等预制组件，可快速构建简洁的对话界面。用户在此输入待办任务，AI 的回复也在此展示。
- **日历视图组件**：展示用户整体日程。可以使用 FullCalendar 等开源日历插件 [8] ，它支持 React/Vue 等，并提供丰富的自定义选项（视图切换、事件渲染等）。也可使用 UI 框架（如 Ant Design）的 Calendar/DatePicker 组件配合自定义渲染。日历视图展示已有日程和规划任务的时间块，用户可直观查看全天或本周安排。
- **任务列表组件**：列出待完成任务和已规划任务。可以使用 UI 库的列表或表格组件（如 Ant Design 的 List、Table）来显示任务标题、预计时长、优先级等。用户可在此对任务进行标记、取消或调整优先级。

- **状态显示栏**（可选）：显示当前用户状态（忙碌/空闲）或快捷设置（如开启“休息模式”）。
- **配置面板**（可选）：允许用户设置个人偏好（如休息时间、习惯标签）。

各组件布局应简洁明了，例如将日历置于主视图中央或左侧，将聊天输入区放在页面底部或右侧，任务列表作为侧栏展示。界面应保证响应式，以适应不同屏幕。

LLM 与业务逻辑交互流程

用户在前端输入文本后，前端将输入通过 HTTP 请求发送到后端接口。后端首先调用 LLM API（如 DeepSeek R1）进行自然语言理解和规划决策。可以采用类似 LangChain 的框架将业务逻辑封装为工具函数：例如使用 `@tool` 装饰器将日程增删查改的基本操作暴露给 LLM [9]。后端为 LLM 构造系统提示和对话上下文，如“你是日程管理助手”，并在用户提示中附加当前时间戳（LLM 默认不具备时间概念）[10]。LLM 根据输入解析出任务内容（描述、时长等）并提出安排方案，然后调用相应工具（如 `add_schedule(start, desc)`）将任务写入数据库 [9] [11]。

具体流程示例：用户输入“明天下午做一个报告（2小时）”，后端按模板加入当前时间发送给 LLM，LLM 返回建议：“你的日程中 15:00-17:00 闲置，安排报告合适，是否确认？”后台收到确认后执行 `add_schedule("2024-05-04 15:00:00", "做报告")` 并更新日历。若无合适空闲，LLM 会提出备选方案（如“次日早上”或“下周”）并再次与用户确认。若任务未在预定时间完成，系统会在任务结束前后向用户询问进度（可视为新的用户输入触发），并调用后端重算剩余任务：固定日程保留，未完成的灵活任务被重新安排到之后的可用时段。通过多轮对话，LLM 和业务逻辑协同完成任务分解和日程更新 [9] [11]。

可扩展建议

- **集成第三方日历**：后续可对接用户真实日历（如 Google Calendar、Apple Calendar）。使用 iCalendar（ICS）标准或 Google Calendar API，同步导入/导出日程；Python 可用 `icalendar` 库处理 iCal 文件 [3]。
- **语音输入/输出**：利用 Web Speech API 实现语音交互，将语音转文本作为输入，或让助手朗读建议，提升无障碍体验。
- **移动端支持**：可将前端封装为 PWA（渐进式 Web 应用），或使用 React Native/Vue 跨平台框架开发原生 App，以支持手机/平板。
- **多模型与本地部署**：可集成本地部署的 LLM 或多种模型提供商（如 Anthropic、Mistral 等），以降低依赖和成本。
- **通知与提醒**：可扩展推送功能，定时提醒用户将要开始的任务（邮件、短信、App 通知等）。
- **多用户与协作**：未来可支持团队日程，共享项目任务与协同安排。

推荐开源组件/库

- **前端框架**：React 或 Vue.js（现代响应式前端框架），配合 UI 库如 Ant Design、Element UI 等快速构建界面 [12] [2]。
- **日历组件**：FullCalendar（支持 React/Vue/原生 JS，配置丰富）[8]。也可使用 Ant Design 的 Calendar/DatePicker 简化实现。
- **聊天/对话 UI**：Assistant UI（React 组件库，专为 AI 对话设计）[6]；`react-chat-ui`（开源聊天组件库）[7]。两者可快速搭建消息气泡、输入框、对话列表等界面。
- **任务列表**：Ant Design 的 List/Table 组件，或开源看板组件如 `react-trello` 等，用于显示和管理任务。
- **后端框架**：Flask、FastAPI（Python），或者 Spring Boot（Java）均可；均支持方便地与 LLM API 集成和构建 RESTful 服务 [1]。

- **LLM 集成**：LangChain（Python 库）可用于管理提示、工具调用和链式逻辑 [9] [11]；OpenAI 官方 SDK 或 DeepSeek R1 的客户端库。
- **日历数据处理**：`icalendar`（Python 库）处理 iCal/ICS 日历文件 [3]；如需时区转换可用 `pytz` 或 `dateutil`。
- **其他工具**：Axios 或 Fetch（HTTP 请求），Socket.IO（实时通信），SQLAlchemy/Django ORM（数据库操作），以及常用日期处理库 `moment.js` 或 `date-fns` 等。

上述设计和技术方案既满足 Web 部署和跨平台需求，又借助现有开源组件加速开发，实现高效的 AI 智能日程规划助手 [12] [9] [4]。

---

[1] [12] Likeday：利用AI打造的个性化日程管理助手-CSDN博客
https://blog.csdn.net/gitblog_00071/article/details/137394329

[2] Build a full-stack LLM application with OpenAI, Flask, React, and Pinecone (Part 1) | by Ashwin Kumar | Medium
https://shwinda.medium.com/build-a-full-stack-llm-application-with-openai-flask-react-and-pinecone-part-1-f3844429a5ef

[3] Internet Calendaring and Scheduling (iCalendar) for Python — icalendar 6.2.1.dev8 documentation
https://icalendar.readthedocs.io/

[4] [5] 从AI日历到项目追踪：解锁专业人士高效工作流 | Bika.ai
https://bika.ai/zh-CN/blog/template/project-tracker/project-tracker-status-updates-ai-calendor

[6] Assistant UI: 打造智能对话界面的React组件库_react ai组件库-CSDN博客
https://blog.csdn.net/m0_56734068/article/details/142462379

[7] 构建聊天UI的利器 —— `react-chat-ui`-CSDN博客
https://blog.csdn.net/gitblog_00087/article/details/139057184

[8] FullCalendar - 开源的多功能 JavaScript 日历插件-CSDN博客
https://blog.csdn.net/nmgwap/article/details/126357727

[9] [10] [11] 用LangChain打造一个可以管理日程的智能助手_langchain 使用时间处理 工具-CSDN博客
https://blog.csdn.net/xindoo/article/details/138448675