
北京航空航天大學

BEIHANG UNIVERSITY

# 《Python 程序设计》大作业报告

——HealthTrace 健迹：基于 python 的学生健康管理系统

**1 组**

**成员：周子强，廉晟，童震天**

# 目录

# 一、引言

随着现代大学生健康问题的日益突出，睡眠不足、缺乏运动和饮食不规律等情况已严重影响学生的生活质量和学习效率。因此，构建一套科学的学生健康管理系统具有重要意义。本项目基于 Flask 和 Vue 3 等现代 Web 开发技术，设计并实现了一个集睡眠监测、运动规划和饮食记录于一体的学生健康管理平台。该平台支持用户注册、登录与个人健康档案管理，通过数据采集与可视化分析，为用户提供周期性健康报告和个性化建议，帮助学生自我管理并改善生活习惯。该项目不仅提升了团队对 Python 全栈开发的理解，也锻炼了我们在实际健康数据分析中的建模与应用能力。具体功能模块说明如下：

- **首页：**根据用户记录的睡眠、运动和饮食数据，生成每个周期的健康报告，提供一周睡眠数据的可视化展示。
- **睡眠记录：**记录用户每天都入睡和起床时间，并计算睡眠时长，提供一周睡眠数据的可视化展示。
- **运动记录：**记录用户每天的运动情况，包括运动类型、运动时长和消耗的卡路里数等。
- **饮食记录：**记录用户每天的三餐饮食，包括食物名称、份量和估算的卡路里等。
- **个人中心：**实现用户注册、登录、注销以及个人健康档案管理功能。

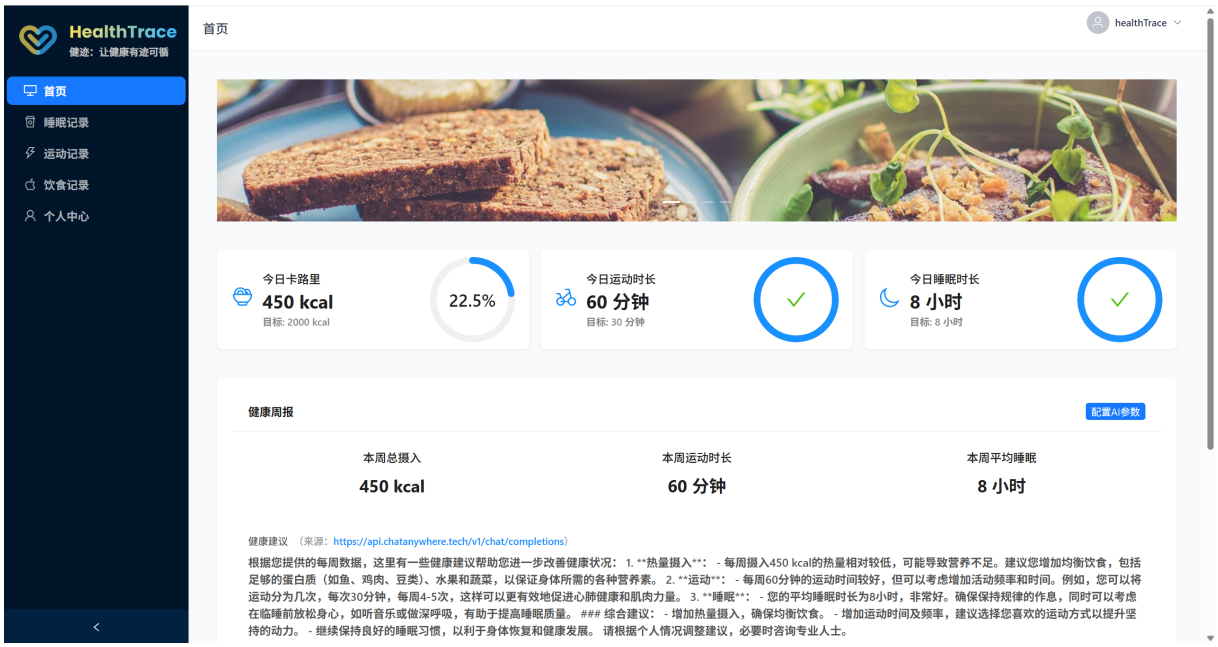


**图 1 网站界面图**

# 二、系统设计

## （一）整体架构

系统采用了前后端分离的 Web 应用架构，整体架构如图2所示。

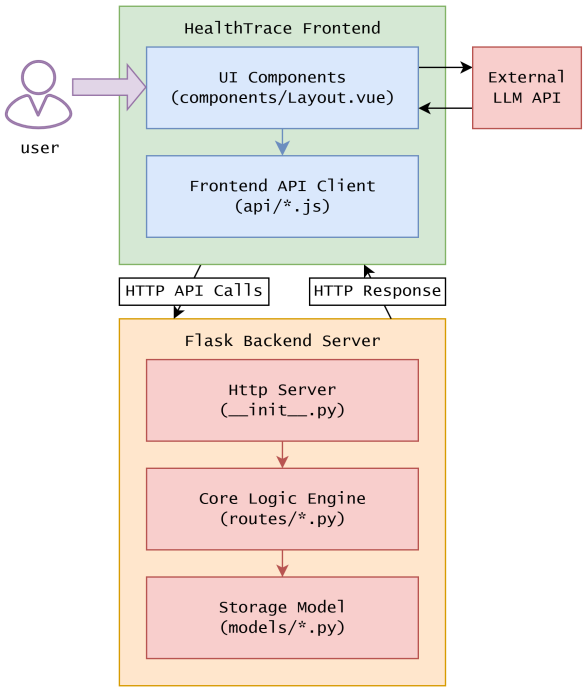


**图 2 系统整体架构图**

前端使用 Vue.js + Ant Design Vue + Element Plus 构建，主要由用户界面组件（Layout.vue）和前端 API 客户端（api/*.js）组成。用户通过界面进行操作，前端 API 客户端负责将这些操作转换为 HTTP API 调用，发送给后端服务器；或通过配置 LLM API 地址以及 API Key 与外部 LLM 交互。

后端基于 Flask 框架，使用 SQLAlchemy 进行 ORM 映射，RESTful API 设计，支持 CORS，（由 __init__.py 实现），用于接收来自前端的请求，并将其分发到相应的核心处理逻辑。核心功能由 routes/.py 提供。数据库中的存储模型由 models/*.py 定义，存储在 SQLite 数据库中，整个系统中数据的流动过程如图3所示。

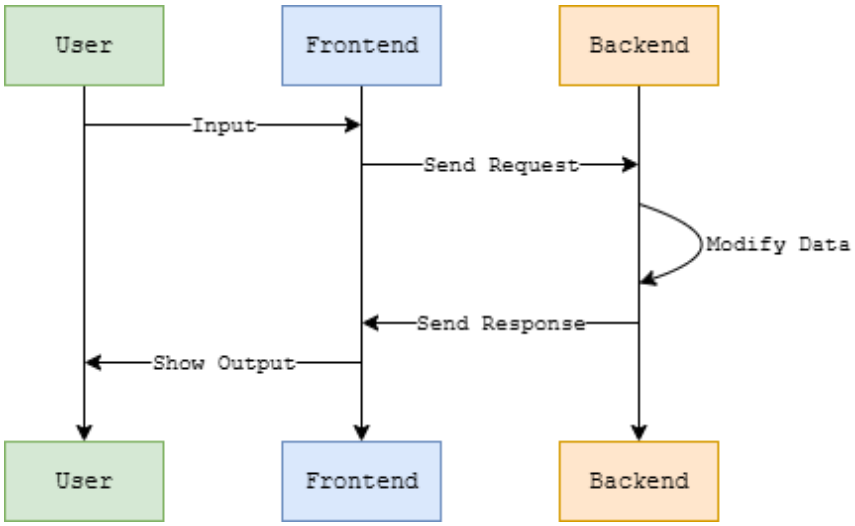


**图 3 系统数据流图**

## （二）后端设计

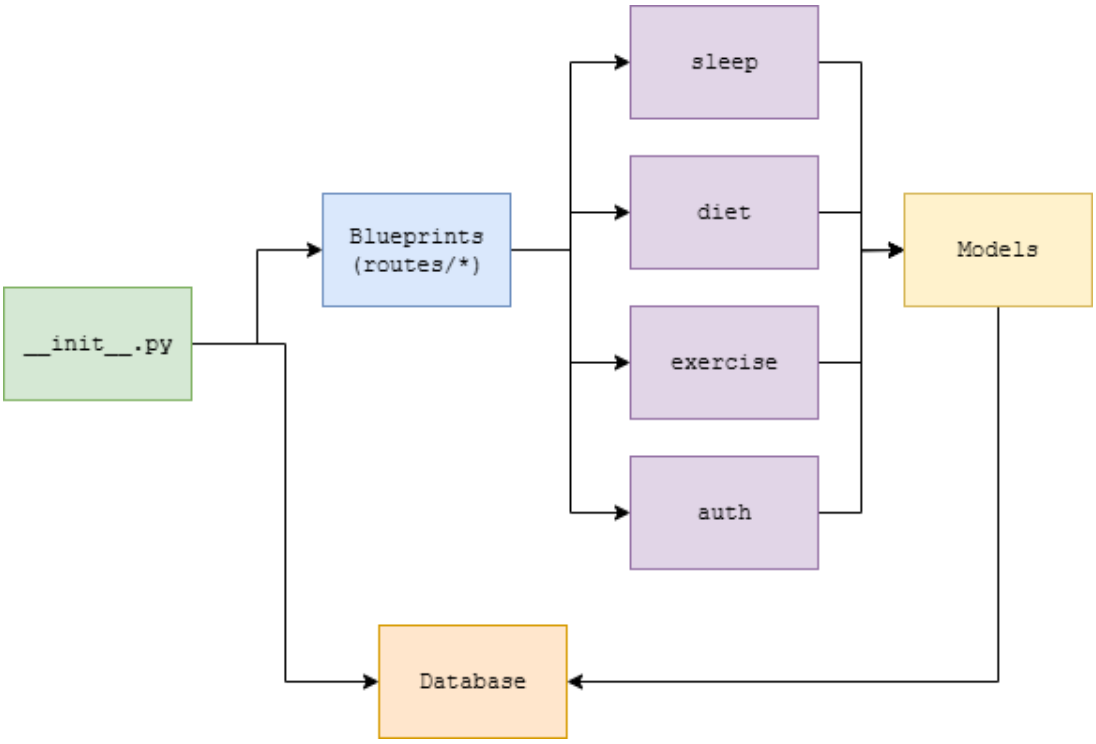


**图 4 后端架构图**

后端架构如图4所示，使用 Python Flask 实现，主要组件包括：

- 主入口 (__init__.py)：初始化和配置 HTTP 服务，创建数据库表
- routes 目录下每个功能模块一个蓝图（auth_bp、sleep_bp、exercise_bp、diet_bp），统一注册到主 app。每个蓝图下实现对应的 RESTful API，包括增删改查和统计等

- utils/response.py 提供统一的 success_response 和 error_response，保证接口返回格式一致。
- 使用 SQLAlchemy 作为 ORM，所有模型集中在 models 目录下（user、sleep、exercise、diet）。

## （三）前端设计

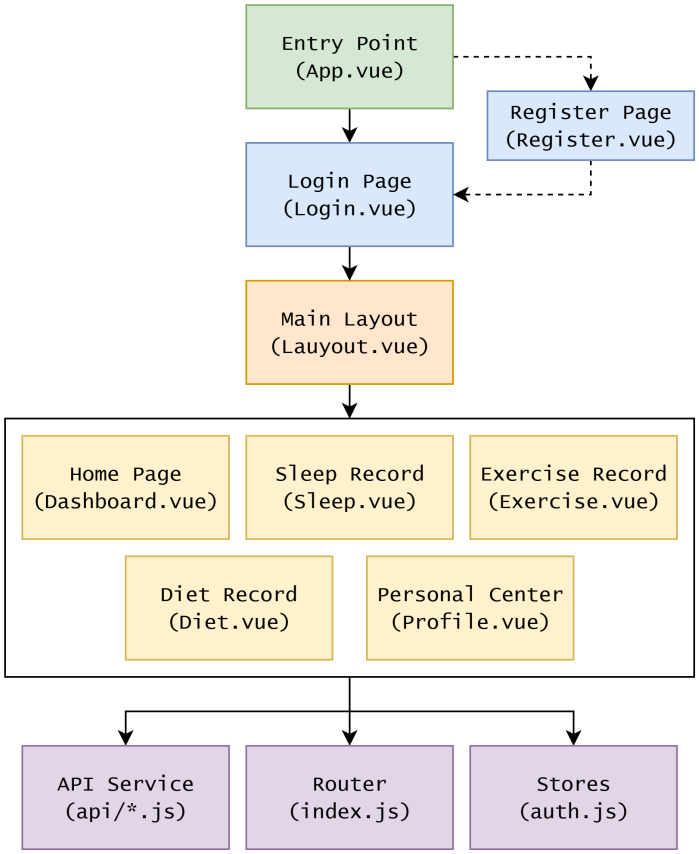


**图 5 前端架构图**

前端架构采用 Vue + Ant Design Vue + Element Plus 设计，如图5所示，主要组件包括：

- 应用入口 (App.vue)：管理全局配置和主题
- 注册页面 (Register.vue)：实现用户注册
- 登录页面 (Login.vue)：实现用户登录
- 主布局 (Layout.vue)：界面整体布局及侧边栏配置
- 其他布局 (Dashboard.vue, Sleep.vue, Exercise.vue, Diet.vue, Profile.vue)：首页、睡眠记录页面、运动记录页面、饮食记录页面及个人中心

- API 服务 (api/*.js)：与后端通信的接口
- 路由跳转 (index.js)：定义路由跳转规则
- 用户管理 (auth.js)：管理用户登陆状态，更新本地存储

# 三、主要功能实现

## （一）后端核心功能实现

### 1. 账户管理

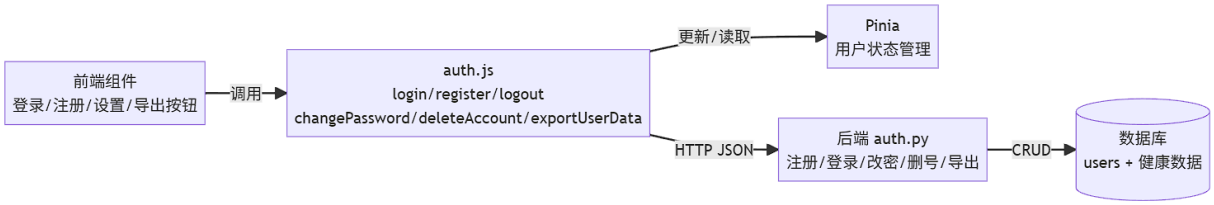


**图 6 账户管理架构图**

接收 JSON 请求体，返回统一格式的 JSON 响应。实现思路（见图6）：

- 注册（/users/register）：支持新用户注册，注册时自动为新用户生成示例饮食、运动、睡眠记录。
- 登录（/users/login）：校验用户名和密码，登录成功返回 userId。
- 修改密码（/users/change-password）：需提供用户名、旧密码和新密码，校验通过后更新密码。
- 注销账户（/users/delete-account）：需提供用户名和密码，校验通过后删除用户（可扩展为级联删除所有健康数据）。
- 导出接口（/users/export-data）：通过 userId 查询用户基本信息，并可扩展导出所有健康数据（饮食、运动、睡眠等），返回 JSON 格式，便于前端下载或展示。

### 2. 健康信息记录

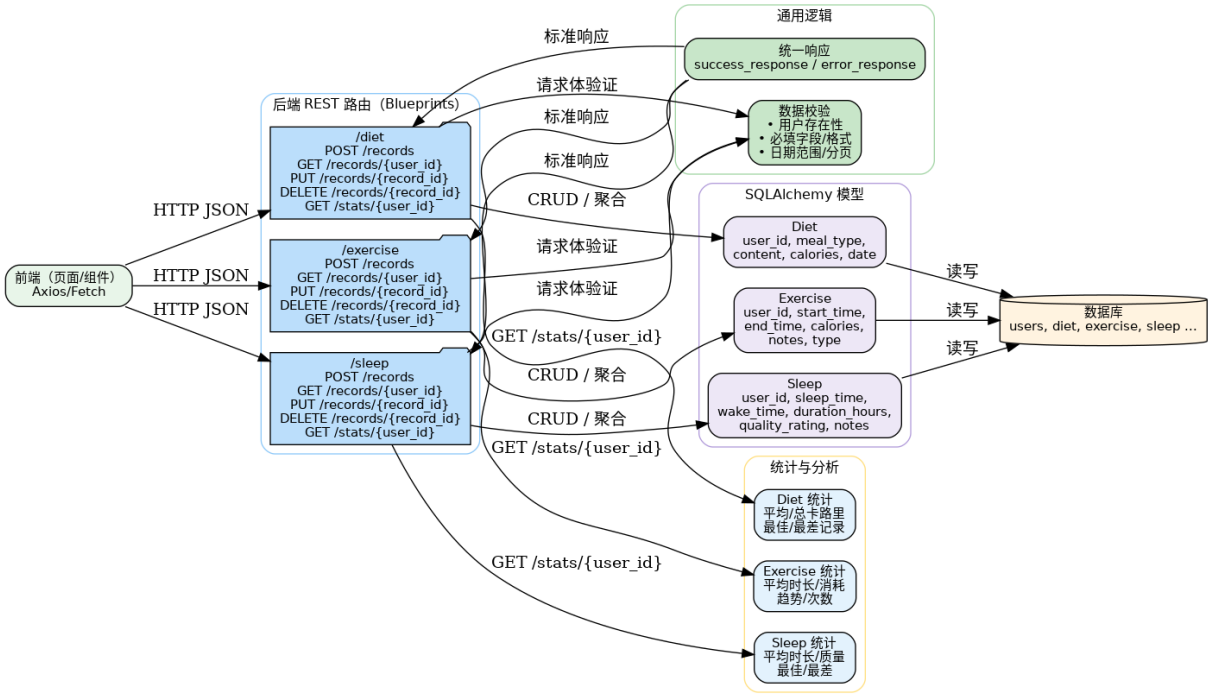


**图 7 信息记录架构图**

下面以睡眠记录的具体实现为例。实现思路（见图7）：

- 后端模型：Sleep，包含 user_id、sleep_time、wake_time、duration_hours、quality_rating、notes 等字段，自动计算睡眠时长。
- 主要接口：
  - POST /sleep/records：添加睡眠记录
  - GET /sleep/records/user_id：获取用户睡眠记录（支持日期筛选、数量限制）
  - PUT /sleep/records/record_id：更新睡眠记录
  - DELETE /sleep/records/record_id：删除睡眠记录
  - GET /sleep/stats/user_id：获取统计（平均时长、平均质量、最佳/最差记录等）
- 返回建议：在 advice_service.py 中有统一的健康建议接口 /advice/get，会根据用户最近 30 天的饮食、运动、睡眠平均值，调用 Judge.judge_sleep(ave_sleep) 返回“请增加睡眠时长”或“睡眠充足”等建议

## （二）前端主要功能实现

### 1. 用户界面与交互

前端整体布局使用了 Ant Design Vue 的 `a-layout` 系列组件实现响应式布局。在 Layout.vue 中，最外层采用 `<a-layout>` 包裹，左侧为可折叠的侧边栏（`<a-layout-`

sider>），右侧为主内容区。侧边栏中包含一个标题栏（logo 及应用名称）和导航菜单（<a-menu>）。用户点击菜单项时，会通过 onMenuClick 事件处理函数根据菜单键值导航到对应的页面路由，并同步更新 selectedKeys，以高亮当前选中项。侧边栏可以折叠/展开（collapsed 绑定到 siderWidth 计算属性），折叠时仅显示图标和简化的标题栏效果（通过动态类控制文字隐藏）。顶部区域为 <a-layout-header>，左侧显示当前页面标题（由路由路径计算而来），右侧则为用户信息区域。用户区域使用 Element Plus 的 el-dropdown 组件，展示用户头像和用户名，点击后弹出下拉菜单，包含“个人资料”和“退出登录”两个选项。点击“退出登录”时使用 ElMessageBox.confirm 弹出确认框，确认后调用 authStore.logout() 并跳转到登录页，实现退出功能。页脚采用 <a-layout-footer>，显示虚拟版权信息。

在 Dashboard.vue 中实现了首页仪表盘的界面。首先使用 Ant Design Vue 的 <a-carousel> 组件制作了一个自动播放的图片轮播，上面循环展示多张横幅图（通过 <img> 标签加载资源），并在最后一个幻灯片中放置一个「关注我们」的开发者信息面板，内含 GitHub 链接。接下来是三个统计信息卡片（使用 <a-card> 组件），分别展示“今日卡路里”、“今日运动时长”和“今日睡眠时长”。每个卡片内有对应的图标（使用 Element Plus 图标组件），以及标题和数值。通过自定义的 CSS 类（如 hide-xs、hide-sm-xs）控制部分卡片在窄屏设备上的可见性，实现响应式适配。之后是两个图表区域卡片，标题为“本周卡路里摄入”和“运动趋势”，卡片右上角含有跳转详情页的链接（使用 <router-link>）。卡片使用 :loading="chartLoading" 属性，根据后台数据加载状态显示加载动画；数据加载完成后在卡片内容区展示本周累计的数值。最后一个“快速操作”卡片中放置了三个按钮（el-button），分别用不同颜色和图标表示“记录饮食”、“记录运动”、“记录睡眠”功能，点击按钮会通过路由跳转到相应的数据录入页面。以上界面布局和组件结合，使得用户能够直观查看个人健康统计并快速进行记录操作。

其余页面实现类似，此处不再赘述。

### 2.API 调用与结果处理

前端与后端的数据交互集中在 Dashboard.vue 的 loadDashboardData() 函数中。该函数在组件挂载时 (onMounted) 被调用，用于获取用户当天和本周的健康统计数据，流程如下：

- 数据请求与参数准备：首先检查当前用户是否已登录，若未登录则通过 ElMessage.error 提示错误并终止加载。若已登录，则通过并行执行 Promise.all 调用事先定义好的服务函数来获取饮食、运动、睡眠统计数据。
- 参数传递：对于上述统计接口，前端只需传入用户 ID 和统计天数即可。这些参数通过函数调用传递给后端，后端根据路径参数和查询字符串返回相应的数据结构。
- 加载状态管理：在发起数据请求之前，将响应式变量 chartLoading 设为 true，使绑定了 :loading="chartLoading" 的卡片组件展示加载动画。这样用户可以直观地看到正在获取数据的提示。

- 成功结果展示：如果所有 API 请求均成功返回，使用返回的 JSON 数据设置对应的响应式变量：`todayCalories`、`todayExercise`、`lastNightSleep` 分别存储当天饮食热量、运动时长、睡眠时长；`weeklyCalories`、`weeklyExercise` 存储一周内累计的热量和运动时长。然后界面会自动更新：统计卡片上的数值会显示最新数据，图表卡片中显示本周总摄入/总运动时间等信息。

- 失败信息处理：如果在数据获取过程中任意请求失败（例如网络错误或后端返回错误），会进入 `catch` 分支：首先在控制台输出错误日志，并通过 `ElMessage.error` 向用户提示“加载数据失败”。然后对各项统计值赋予默认值（如 0 或固定值），确保页面显示不会留空，避免出现未定义。最后在 `finally` 中将 `chartLoading` 设为 `false`，关闭加载动画。这样即使后端数据暂时无法获取，界面也能正常加载并提示用户当前无可用数据。

综上，前端通过模块化的接口函数向后端请求数据，并根据响应结果动态更新视图。无论是成功还是失败，用户界面都能给予明确反馈，同时保持数据一致性和交互流畅性。

3. **图表功能**

使用 ECharts 库在前端展示用户的睡眠趋势图表。实现思路：

- 在页面模板中添加一个 ECharts 容器（`<div ref="sleepChartRef">`），并将其放在标题为“一周睡眠趋势”的卡片中。

- 页面加载时（`onMounted` 钩子）调用 `loadSleepStats` 函数获取最近 7 天的睡眠记录。首先对已有记录按时间排序找出最新日期，然后生成最近 7 天的日期数组。

- 遍历日期数组：若某日期存在睡眠记录，则将该时长加入 `hours`，并在 `isAverageFlags` 中标记为 `false`；否则加入平均睡眠时长，并标记为 `true`。填充完成后调用 `initSleepChart(dates, hours, isAverageFlags)` 初始化图表。

- 在 `initSleepChart` 中使用 `echarts.init(chartDom)` 创建 ECharts 实例，并监听窗口 `resize` 事件以自动调整图表尺寸。

- 最终图表绘制在“一周睡眠趋势”区域，可直观展示用户最近七天每天的睡眠时长趋势，帮助用户了解睡眠规律。

4. **外接 LLM**

前端提供 AI 参数配置和调用外部 LLM 接口的功能，实现思路：

- 在界面上添加一个“配置 AI 参数”按钮，点击后弹出配置弹窗。弹窗内包含三个输入项：LLM 接口地址、API Key、模型名称。用户填写后点击“保存”按钮，将临时变量的值保存到正式的 `apiUrl`、`apiKey`、`model` 中。

- 使用 Vue Composition API 定义响应式变量：`dialogVisible` 控制弹窗显示状态，`tempApiUrl/tempApiKey/tempModel` 存储用户输入，`apiUrl/apiKey/model` 存储当前配置，`llmAdvice` 存储 LLM 返回的健康建议文本。

- 定义 `createPrompt(calories, exercise, sleep)` 函数，根据每周摄入热量、运动时长、睡眠平均时长拼接提示语，指导 LLM 生成针对性的健康建议。例如：

  ```
  你是一名健康顾问。请根据以下每周数据生成简短的健康建议：
  - 每周摄入热量：${calories} kcal
  - 每周运动时长：${exercise} 分钟
  - 每周平均睡眠时长：${sleep} 小时
  请给出具体且易懂的建议。
  ```

- 定义异步函数 `fetchAdvice(calories, exercise, sleep)` 发送 HTTP POST 请求到 LLM 接口。请求头携带 `Authorization: Bearer ${apiKey}`，请求体为 JSON（包含 `model`、`messages`、`max_tokens` 等字段）。

- 处理 LLM 响应：如果返回的 JSON 中 `choices[0].message.content` 存在，则将其赋值给 `llmAdvice`，否则提示“暂无建议”；如果请求失败或超时，则捕获错误并将 `llmAdvice` 设置为错误提示（如“无法加载建议，请稍后重试”）。最终用户可在界面其他位置（如建议面板）看到由 LLM 返回的健康建议文本。

# 四、遇到的问题与解决方案

项目在开发过程中遇到不少问题。

首先是如何确保 API 设计与前端调用一致。前端调用 API 时，参数名、请求体结构、返回字段可能与后端实际实现不一致，导致数据无法正确提交或解析。例如，前端传递的字段名与后端模型不匹配，或前端期望的响应格式与后端返回的不同（如 data 字段缺失、msg 字段名不统一）。我们的解决办法是明确接口文档，前后端协商统一 API 请求和响应格式，并要求后端所有接口返回统一结构（如 success, message, data ）。

其次是路由设计与模块划分不清的问题。后端路由未按功能模块分组，或路由路径与前端页面路由不一致，导致前端找不到对应 API。例如，/api/sleep 与 /sleep-/records/user_id 混用，或部分功能未注册蓝图。最后，我们的后端采用 Blueprint（Flask）或 Router（Express）等机制，按功能模块（如 /users, /diet, /sleep, /exercise）分组路由。同时保持前后端路由命名一致，便于维护和调用。

# 五、总结与展望

## （一）项目总结

项目完成度较高，基本功能全部实现，也实现了部分扩展功能。

该项目今后努力完善方向有很多，如接入校园网，面向高校学生使用，完善平台多线程的并发支持，适配多用户场景等。

## （二）组内分工与合作

团队在编写代码的过程中使用微信群交流工作，使用 Github 私有仓库对项目代码进行文件和合并管理。最终的报告文档在 Overleaf 上共同完成，PPT 使用 canvas 平面设计工具辅助完成，演示视频使用剪映剪辑完成。文档中的配图使用 draw.io 等工具绘制。成员分工具体如下：

- 周子强：负责前端界面设计与实现，接入 LLM，并制作 PPT 和演示视频。
- 廉晟：负责数据可视化和健康建议功能，并协调前后端联调。
- 童震天：负责后端大部分主要逻辑，并搭建前后端整体框架及测试。