

每日膳食营养统计系统

课程大作业报告

项目项	说明
项目类型	Qt Widgets 桌面应用 / 生活健康辅助工具
开发语言	C++17
主要技术	Qt Widgets、Qt Network、CSV 数据库、JSONL 记录、DeepSeek API
报告范围	程序功能、模块与类设计、成员分工、总结反思

1. 程序功能介绍

每日膳食营养统计系统面向日常饮食记录场景，用户可以登录个人账号，录入一天内摄入的食物及重量，程序根据本地营养库计算热量、蛋白质、脂肪、碳水化合物、膳食纤维、糖、胆固醇、维生素 C 和钠等指标，并支持将当前饮食数据交给 AI 生成分析报告。

系统定位：本项目选择“生活游戏 / 生活健康”方向，核心目标是把原本繁琐的膳食记录变成更低门槛的桌面工具，通过自然语言输入和智能报告降低用户使用成本。用户不需要接触 JSON 等技术格式，只需要输入类似“今天中午吃了 150 克米饭、100 克猪肉”的自然语句即可完成批量添加。

1.1 已实现功能

功能模块	实现说明
用户账户	支持注册、登录、退出登录；保存用户名、密码、年龄、性别、身高、体重、活动水平和健康目标。
个人资料	登录后展示当前用户信息，计算并显示 BMI，为后续报告推荐提供基础条件。
食物录入	支持手动选择类型、输入名称和重量；也支持自然语言描述批量添加。
营养统计	从 nutrition_db.csv 读取每 100g 营养数据，按实际重量换算并汇总多项营养指标。
食物库扩展	当用户输入本地库中不存在的食物时，通过 DeepSeek 查询估算营养值，并自动追加到 CSV 食物库。
历史记录	将当前饮食保存为按用户区分的 JSONL 文件，支持弹窗查看历史饮食条目与总营养。
AI 报告	可生成当日饮食报告和历史饮食报告，报告内容包含总体评价、问题分析和改进建议。
界面体验	使用 Qt Widgets 构建可视化界面；采用 QSS 美化，状态栏提示操作结果；网络请求异步执行，避免界面卡顿。

1.2 基本使用流程

1. 启动程序后加载本地 nutrition_db.csv 营养数据库，并初始化用户管理模块。

- 2. 用户注册或登录账号，填写年龄、性别、身高、体重、活动水平和健康目标。
- 3. 用户通过手动录入或自然语言输入添加食物，系统自动换算营养摄入。
- 4. 用户可以保存当天记录、查看历史记录，也可以调用 AI 生成饮食分析报告。

2. 项目各模块与类设计细节

项目采用“界面层 + 系统协调层 + 数据与业务层”的结构。Qt 界面负责输入输出和异步网络请求；DietSystem 作为系统门面，协调用户、食物、营养库和记录保存；底层类负责单一职责的数据处理。这样的设计使界面逻辑和核心统计逻辑相对分离，便于后续扩展。

2.1 模块结构

模块	相关类/文件	职责
界面模块	main.cpp / DietWindow	负责窗口布局、按钮事件、表格展示、状态提示、AI 网络请求和报告弹窗。
系统协调模块	diet_system.h / DietSystem	封装注册登录、食物添加、营养汇总、保存记录等对外接口。
用户模块	user_manager.h / UserManager、User	维护用户资料，负责注册、登录、资料更新和本地 users.dat 持久化。
食物模型模块	food.h / Food、StapleFood 等	以抽象基类和派生类表示不同类别食物，按重量计算实际摄入。
食物创建模块	food_factory.h / FoodFactory	根据食物类型创建对应派生类对象，隐藏对象创建细节。
营养库模块	nutrition_db.h / NutritionDatabase	加载 CSV 营养数据库，提供精确匹配和包含式匹配。
饮食管理模块	diet_manager.h / DietManager	保存当前餐次食物列表，支持添加、删除、清空和总营养计算。
记录模块	record_manager.h / RecordManager	按用户生成 diet_record_xxx.jsonl 文件，保存时间、食物列表和总营养。
AI 解析模块	ai_parser.h / AiFoodParser	解析 AI 返回的 JSON 食物数组，提取 type、name、weight。

2.2 主要类设计

NutrientData：营养数据结构，保存热量、蛋白质、脂肪、碳水、膳食纤维、糖、胆固醇、维生素 C、钠等字段，并重载加法运算，用于多种食物的营养累加。

Food 及其派生类：Food 是抽象基类，保存食物名称、重量和每 100g 营养数据；getActualIntake() 按 weight / 100 换算实际摄入。StapleFood、MeatFood、VegetableFood、FruitFood 通过重写 getType() 区分主食、肉类、蔬菜和水果。

FoodFactory：使用工厂方法根据类型字符串创建对应 Food 派生类对象，避免界面层直接依赖具体派生类。

NutritionDatabase：内部使用 unordered_map 存储食物名称到营养数据的映射。loadFromCsv() 从 CSV 文件读取数据，find() 先进行精确匹配，再进行包含式匹配，提高用户输入容错性。

DietManager：使用 vector<unique_ptr<Food>> 管理当前饮食列表，既避免手动释放内存，也体现了 C++ 智能指针在对象生命周期管理中的应用。

UserManager：负责用户数据读取、注册、登录和资料更新。用户资料字段从最初的用户名和密码扩展到年龄、性别、身高、体重、活动水平和健康目标，使系统能给出更个性化的饮食建议。

RecordManager：采用 JSON Lines 格式保存历史记录，每行是一条完整记录，便于追加写入和逐行读取。文件名中会对用户名做安全处理，避免非法文件名字符。

AiFoodParser：负责将 AI 返回的 JSON 文本解析为 AiFoodInput 列表。它没有依赖额外第三方 JSON 库，而是实现了轻量解析逻辑，能跳过未知字段并返回有效食物项。

DietWindow：继承 QWidget，是程序的主界面类。它负责创建用户区、食物录入区、当前食物表格、总营养区和底部操作按钮，并通过 QNetworkAccessManager 异步调用 DeepSeek API。未知食物查询、自然语言解析、当前报告和历史报告都在该层完成交互控制。

2.3 数据流与关键实现

- 自然语言添加：用户输入中文饮食句子后，程序优先根据本地食物库和重量规则提取食物；必要时调用 DeepSeek 将句子转换为内部 JSON 数组，再交给 DietSystem 添加。
- 未知食物入库：若食物库没有对应条目，程序请求 DeepSeek 返回每 100g 营养估算，写入 nutrition_db.csv 后重新加载数据库，再继续添加食物。

- 营养计算：食物类保存每 100g 营养值，DietManager 遍历当前食物列表，通过 NutrientData 加法得到总摄入。
- 历史记录：保存时将当前食物列表和总营养写入 diet_record_用户名.jsonl，查看历史时再逐行解析并格式化展示。
- 报告生成：程序把用户资料、当前饮食或历史饮食数据组织为提示词，异步发送到 DeepSeek，返回后以弹窗展示分析结果。
- 路径适配：程序会在源码目录、exe 同级目录、当前目录及上级目录查找 nutrition_db.csv，减少发送给他人运行时的路径问题。

3. 小组成员分工情况

成员	负责方向	具体工作
李 楷	总体设计与核心业务，Qt 界面与交互	确定项目功能范围；设计 Food、NutrientData、DietSystem、DietManager 等核心类；完成营养计算、食物管理和系统整合。完成主窗口布局、用户区、食物录入区、当前食物表格、总营养展示、历史记录弹窗和界面美化。
张 大卫	AI 与数据存储	完成 DeepSeek API 调用、异步请求、自然语言解析、未知食物自动入库、报告生成、CSV 和 JSONL 文件读写。
共同完成	测试与修改	共同进行功能测试、路径问题修复、编码问题处理、运行截图整理、PPT 和报告材料完善。

4. 项目总结与反思

4.1 项目成果

- 完成了一个可运行的 Qt 桌面端膳食统计程序，具备用户管理、饮食录入、营养计算、历史保存和报告生成等完整流程。
- 将传统表单录入与自然语言输入结合，隐藏 JSON 技术细节，使普通用户能够更直观地记录饮食。

- 本地 CSV 营养库与 DeepSeek API 结合，既能快速查询常见食物，也能在遇到未知食物时自动扩展数据。
- 通过异步网络请求改善用户体验，避免 AI 解析和报告生成期间界面无响应。
- 项目中使用继承、多态、工厂方法、智能指针、文件持久化等 C++ 面向对象知识，符合课程综合实践要求。

4.2 开发中的问题与解决

问题	原因	处理方式
AI 请求较慢	自然语言解析、未知食物查询和报告生成依赖网络与模型响应。	改为异步请求，使用状态提示告知用户当前操作进度。
JSON 对用户不友好	早期界面暴露 JSON 批量添加输入，不符合普通用户使用习惯。	删除可见 JSON 输入，保留自然语言输入作为主要交互方式。
个性化不足	只记录食物时，报告缺少年龄、性别、体型和目标背景。	扩展用户资料字段，增加年龄、性别、身高、体重、活动水平和目标。

4.3 不足与改进方向

- 用户密码目前以简单文本形式保存，正式应用中应进行加密或哈希存储。
- DeepSeek API Key 写入客户端代码只适合课程演示，实际发布应放在服务端或安全配置中。
- CSV 和 JSONL 适合小型作业，但数据规模增大后可改用 SQLite，提高查询、筛选和维护能力。
- 目前历史记录以文本弹窗展示，后续可增加按日期筛选、营养趋势图、周/月统计和报告导出功能。
- AI 估算营养值存在不确定性，应在界面中标记“AI 估算”，并允许用户手动修正食物库数据。
- 界面已经进行了基础美化，但仍可进一步优化布局密度、响应式适配和操作反馈。

4.4 个人反思

通过本项目，开发过程从最初的命令行和基础数据结构，逐步扩展到 Qt 图形界面、文件持久化、网络 API、异步交互和 AI 报告生成。这个过程说明，一个课程作业如果只关注“能算出结果”，功能会比较单薄；而当加入用户资料、自然语言输入、历史记录和报告反馈后，程序才更接近真实用户会使用的工具。

同时，项目也暴露了桌面程序发布时常见的问题，例如运行目录、外部数据文件、中文编码、网络延迟和密钥安全。后续如果继续完善，可以优先提升数据安全和历史分析能力，让系统不仅记录一餐，而是帮助用户长期理解自己的饮食结构。