中山大学本科生课程作业

软件工程期末作业

Software Engineering Final Assignment

学年学期	2024 学年第二学期
开课院系	中山大学智能工程学院
姓名学号（字母顺序）	22354010 常毅成　22354011 陈泓逸 22354030 高楚航　22354035 郭皓玮 22354065 李金艾　22354106 石成涛 22354118 孙雨　22354189 张瑞程

目　　录

《设计合理性报告》

一：整体框架

1.1 架构设计哲学与核心思想

本系统采用"智能驱动、用户中心"的设计理念，将现代营养学理论与信息技术深度融合，构建了一个三层认知体系：

- **数据认知层**：通过多维度用户数据分析建立个性化画像
- **策略决策层**：基于专业营养学模型生成适应性方案
- **交互呈现层**：采用渐进式披露原则实现复杂信息的友好传达

系统架构遵循"松耦合、高内聚"的设计原则，各模块通过定义良好的接口契约进行协作，既保证功能独立性，又维持系统整体性。这种设计使系统具备"热插拔"特性，可灵活替换算法模块或界面组件。

1.2 MVC 架构的深化实践

项目采用典型的 MVC（Model-View-Controller）架构。

1.2.1 Model 层的专业化分工：diet_agent.py 和 workflow.py 处理业务逻辑

- **领域模型**：封装核心业务实体（如用户档案、食物营养数据）及其关系
- **服务模型**：实现具体的业务逻辑（如热量计算、营养均衡算法）
- **存储模型**：抽象数据持久化操作，支持多种数据库后端

三层模型通过"依赖倒置"原则形成层次结构，高层模块不直接依赖低层实现，而是通过接口交互，极大提升了系统可测试性和维护性。

```
class DietAgent:
    def __init__(self, user_profile):
        self.metabolic_rate = calculate_metabolism(user_profile)
        self.nutrient_needs = load_nutrient_standards()

    def analyze_meal(self, meal_data):
        # 实现营养成分分析算法
        return {
            'calories': calculate_calories(meal_data),
            'macros': calculate_macronutrients(meal_data)
        }
```

```
class DietWorkflow:
    def process_request(self, request_type, user_data):
        if request_type == 'recommendation':
            return self._generate_recommendation(user_data)
        elif request_type == 'analysis':
            return self._analyze_diet(user_data)
```

1.2.2 View 层的响应式设计体系：templates/中的 HTML 模板与 static/中的前端资源

- **自适应布局系统**：基于 CSS Grid 和 Flexbox 构建 12 栅格体系
- **动态主题引擎**：支持根据用户偏好实时切换视觉风格
- **渐进增强策略**：确保核心功能在各类设备上均可访问
- **组件化开发模式**：将界面元素封装为可复用的 Web Components

```
<!-- base.html片段 -->
{% block content %}
<div class="container">
  {% include 'navbar.html' %}
  {% block page_content %}{% endblock %}
</div>
{% endblock %}
```

1.2.3 Controller 层的流程控制艺术：main.py 作为应用入口，协调请求响应

- **中介者模式**：作为系统通信枢纽协调各模块协作
- **管道过滤器**：对请求处理过程进行分阶段加工
- **上下文对象**：封装请求全过程的状态信息
- **拦截器链**：实现横切关注点（如认证、日志）的统一处理

```
@app.route('/recommend', methods=['POST'])
def recommend():
    user_data = request.get_json()
    recommendation = workflow.process_request('recommendation', user_data)
    return render_template('recommend.html', data=recommendation)
```

1.3 数据流示意图

这张数据流图展示了健康饮食系统"查看档案"功能的典型处理流程：用户操作触发界面请求 → 控制中心接收并调度 → 数据处理模块执行计算 → 结果返回界面展示。这种分层架构实现了显示、控制和数据处理的有效分离，使系统更安全、易维护且性能更优，是 Web 应用开发的基础模式。

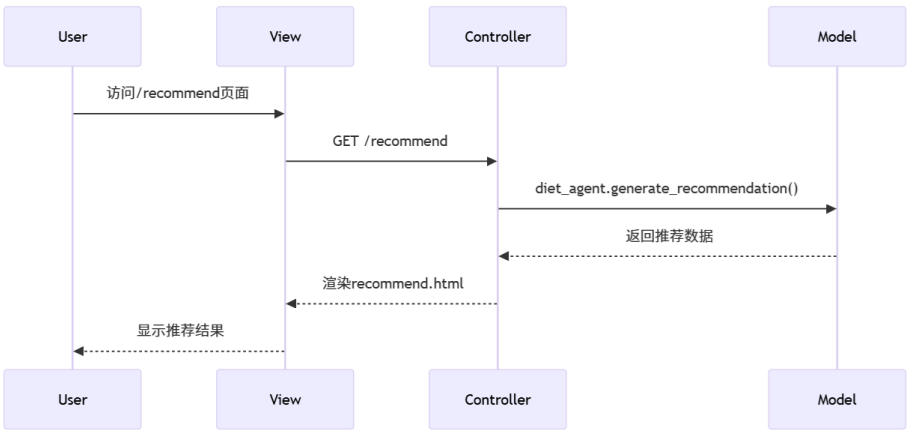

图 1 数据流示意图

二：前端资源体系

前端技术运用模块化 CSS 设计（variables.css 定义设计系统）、响应式布局（responsive.css）、页面过渡动画（page-transitions.css）、交互增强（animations.js）等多个模块完成；而后端技术依托基于 Agent 的架构（diet_agent.py）、工作流引擎（workflow.py）、多 Python 版本支持（3.9/3.10/3.11 字节码缓存）。

```
static/
├── css/              # 样式表(18个CSS文件)
│   ├── animations.css      # 动画效果
│   ├── responsive.css      # 响应式布局
│   ├── variables.css       # CSS变量定义
│   └── [page]-fix.css      # 各页面定制样式
├── images/           # 图片资源(12个)
│   ├── bg-pattern.png      # 背景纹理
│   ├── [feature].jpg       # 各功能模块配图
│   └── logo.png            # 品牌标识
└── js/
    ├── animations.js       # 交互动画
    └── main.js             # 主业务逻辑
```

图 2 前端资源体系架构

这些 CSS 文件共同构成了一个健康饮食相关网站的样式系统。下面将解释每个文件的主要作用并突出重要部分。

2.1 表单输入框样式 analyze-fix-overlay.css 和 analyze-fix.css

- 解决三餐分析页面的布局和重叠问题
- 包含表单、卡片和响应式布局的修复样式

这段代码为文本输入框提供了现代化的样式，包括聚焦状态下的高亮效果。

```css
textarea {
  width: 100%;
  padding: 1rem 1rem 1rem 2.8rem;
  border: 1px solid #ddd;
  border-radius: 8px;
  font-size: 1rem;
  transition: all 0.2s ease;
}

textarea:focus {
  outline: none;
  border-color: var(--primary-color);
  box-shadow: 0 0 0 3px rgba(76, 175, 80, 0.15);
}
```

2.2 响应式网格布局 economical-fix-overlay.css 和 economical-fix.css

- 针对经济型饮食页面的样式修复
- 包含功能亮点卡片、表单和推荐结果的布局调整

这段代码创建了一个自适应的网格布局，在大屏幕上显示多列，在小屏幕上

自动调整为单列。

```
.recommendations-wrapper {
  display: grid;
  grid-template-columns: repeat(auto-fill, minmax(300px, 1fr));
  gap: 1.5rem;
}

@media (max-width: 768px) {
  .recommendations-wrapper {
    grid-template-columns: 1fr;
  }
}
```

2.3 卡片悬停效果 feature-cards-fix.css 和 区域背景渐变 home-cards-fix.css

- 修复首页功能卡片的显示问题
- 包含卡片网格布局、悬停效果和响应式设计

这段代码创建了卡片的悬停动画效果，包括上浮和阴影变化，以及图标和覆盖层的淡入效果。

```
.feature-card:hover {
    transform: translateY(-5px);
    box-shadow: var(--shadow-hover);
}

.feature-image-wrapper:hover .feature-overlay,
.feature-image-wrapper:hover .feature-icon {
    opacity: 1;
}
```

这段代码创建了一个动态渐变背景，会在不同颜色之间平滑过渡，为英雄区域增加视觉吸引力。

```
.fullscreen-hero .hero-background.bg-fallback {
  background: linear-gradient(135deg, #2E7D32, #81C784);
  animation: gradientShift 15s ease infinite;
  background-size: 200% 200%;
}

@keyframes gradientShift {
  0% { background-position: 0% 50%; }
  50% { background-position: 100% 50%; }
  100% { background-position: 0% 50%; }
}
```

2.4 home-hero.css

- 首页英雄区域的完整样式
- 包含全屏背景、动画效果和滚动指示器

2.5 footer-fix.css

- 修复页脚布局问题
- 包含多列导航、社交媒体图标和响应式调整

2.6 background-fix.css

- 全局背景样式
- 为不同页面设置独特的渐变背景

2.7 动画效果 animations.css

- 各种动画效果
- 包含淡入、滑动、脉冲等多种动画

这段代码定义了一个从下方滑入的动画效果，可以应用于各种元素增加页面动感。

```
@keyframes slideInUp {
    from { transform: translateY(30px); opacity: 0; }
    to { transform: translateY(0); opacity: 1; }
}

.animate-slideInUp {
    animation: slideInUp 0.8s ease forwards;
}
```

三：功能模块

功能模块包含以下六个模块：

- 饮食分析（analyze.html）
- 经济型食谱（economical.html）
- 个人健康档案（profile.html）
- 食谱推荐（recipe.html/recommend.html）
- 减重方案（weight_loss.html）
- 外卖建议（takeout.html）

```
templates/
├── base.html         # 基础模板框架
├── index.html        # 首页
├── analyze.html      # 饮食分析页
├── recommend.html    # 个性化推荐页
├── weight_loss.html  # 减重方案页
└── [other].html      # 其他功能页(共8个)
```

图 3 功能模块构架

3.1 首页（base.html）

3.1.1 响应式导航栏：适配桌面和移动设备，包含菜单切换功能

使用 navbar 类创建顶部导航栏，同时包含品牌 logo 和名称，移动端有三条横线的菜单切换按钮，导航菜单使用 Font Awesome 图标增强视觉效果。

```
<body>
    <!-- 导航栏 -->
    <nav class="navbar">
        <div class="nav-brand">
            <img src="images/logo.png" alt="Logo">
            <span>AI健康饮食助手</span>
        </div>
        <div class="mobile-menu-toggle">
            <span></span><span></span><span></span>
        </div>
        <ul class="nav-menu">
            <li><a href="/"><i class="fas fa-home"></i> 首页</a></li>
            <!-- 其他导航项 -->
        </ul>
    </nav>
```

3.1.2 页面过渡动画：页面切换时的加载动画效果

使用 SVG 创建圆形加载动画，点击内部链接时显示加载动画，300 毫秒后跳转到目标页面。

3.1.3 滚动效果：导航栏阴影、回到顶部按钮等

滚动超过 10px 时给导航栏添加阴影效果，滚动超过 300px 时显示"回到顶部"按钮。

```
// 滚动时导航栏阴影
window.addEventListener('scroll', function() {
    document.querySelector('.navbar').classList.toggle('shadow', window.scrollY > 1
0);
});

// 回到顶部按钮
const backToTop = document.getElementById('backToTop');
if (backToTop) {
    window.addEventListener('scroll', function() {
        backToTop.classList.toggle('show', window.scrollY > 300);
    });
    backToTop.addEventListener('click', function() {
        window.scrollTo({ top: 0, behavior: 'smooth' });
    });
}
```

3.1.4 多页面结构：包含首页、个人信息、推荐食谱等多个功能页面

现代化 UI 设计：使用 CSS 变量、Flexbox 和 Grid 布局

3.1.5 动画效果：元素进入视口时的动画触发

使用 IntersectionObserver API 检测元素是否进入视口，当元素 10%进入视口时添加 active 类触发动画。

3.1.6 移动端菜单切换功能

社交分享，底部社交媒体链接。点击菜单按钮时切换 active 类，同时控制页面滚动防止背景滚动。

3.2 个人信息（profile.html）

这个页面是一个个人健康信息收集和展示界面，主要功能包括：

3.2.1 健康信息收集表单：收集用户的性别、年龄、身高、体重、活动水平和健康目标

收集用户 6 项关键健康数据：性别、年龄、身高、体重、活动水平和健康目标。使用响应式布局，适配不同屏幕尺寸，每个输入字段配有直观的 Font Awesome 图标，实现前端验证确保数据完整性和合理性，表单提交时显示加载状态，防止重复提交。

3.2.2 健康指标计算与展示：根据输入信息计算并显示 BMI 指数、基础代谢率(BMR)、每日建议摄入热量和营养素分配建议

实时计算并展示 4 项关键健康指标：

- BMI 指数及健康状态评估
- 基础代谢率(BMR)
- 每日建议摄入热量
- 三大营养素(蛋白质、脂肪、碳水化合物)分配比例

使用可视化卡片展示计算结果，BMI 状态通过颜色和图标直观显示(偏低/正常/超重/肥胖)，计算结果以适当单位显示，保留合理小数位。

```
// BMI计算与状态显示
function getBMIStatusWithIcon(bmi) {
    let status, icon, color;
    if (bmi < 18.5) {
        status = '体重偏低'; icon = 'arrow-down'; color = 'var(--info-color)';
    } else if (bmi < 24) {
        status = '正常范围'; icon = 'check-circle'; color = 'var(--success-color)';
    } else if (bmi < 28) {
        status = '超重'; icon = 'exclamation-circle'; color = 'var(--warning-colo
r)';
    } else {
        status = '肥胖'; icon = 'exclamation-triangle'; color = 'var(--error-colo
r)';
    }
    return `<span style="color: ${color}"><i class="fas fa-${icon}"></i> ${status}</
span>`;
}

// 指标计算与展示
document.getElementById('bmiValue').textContent = `${result.bmi.toFixed(1)}`;
document.getElementById('bmiStatus').innerHTML = getBMIStatusWithIcon(result.bmi);
document.getElementById('bmrValue').textContent = `${result.bmr.toFixed(0)} 千卡/天`;
document.getElementById('calorieValue').textContent = `${result.daily_calories.toFix
ed(0)} 千卡/天`;
```

3.2.3 数据持久化：将用户输入和计算结果保存到本地存储(localStorage)

使用 localStorage 保存用户输入的健康数据和计算结果，页面加载时自动恢复上次保存的数据。数据以 JSON 格式存储，便于读取和更新，同时保存原始输入和计算结果，减少重复计算。

3.2.4 响应式通知系统：提供操作成功或失败的通知反馈

提供 4 种类型的通知：info(信息)、success(成功)、error(错误)、warning(警告)自动显示和消失，默认 5 秒后自动关闭，支持手动关闭通知，使用图标和颜色区分不同通知类型，采用平滑的动画效果显示和隐藏。

3.3 推荐食谱（recommend.html）

3.3.1 健康信息收集：通过表单收集用户的健康目标、饮食偏好和生活习惯

收集三项关键信息：健康目标、饮食偏好和生活习惯，每个输入字段配有相应的图标增强用户体验，使用响应式设计确保在不同设备上良好显示。

3.3.2 个性化推荐：基于用户输入和存储的健康数据，生成一日三餐的饮食建议

整合表单数据和本地存储的健康信息，向服务器发送请求获取个性化推荐。处理服务器响应并生成可视化结果。

```
// 添加三餐推荐卡片
html += '<div class="meal-cards-grid">';
['早餐推荐', '午餐推荐', '晚餐推荐'].forEach(mealType => {
    const meal = recommendation[mealType];
    html += `
        <div class="meal-card">
            <div class="meal-card-header">
                <i class="${getMealIcon(mealType)}"></i>
                <h3>${mealType}</h3>
            </div>
            <div class="meal-card-content">
                <h4><i class="fas fa-apple-alt"></i> 推荐食物</h4>
                <ul>${formatFoodList(meal.推荐食物)}</ul>

                <h4><i class="fas fa-chart-pie"></i> 营养价值</h4>
                <p>${meal.营养价值 || '暂无信息'}</p>
            </div>
        </div>
    `;
});
html += '</div>';
```

3.3.3 结果展示：以卡片形式展示早餐、午餐和晚餐的推荐食物、营养价值和食用建议

使用卡片式布局展示三餐推荐，包含食物列表、营养价值和食用建议，添加动画效果增强交互体验，显示营养建议总结。

3.3.4 数据整合：结合本地存储的用户健康档案数据，提供更精准的推荐

3.4 分析三餐（analyze.html）

3.4.1 三餐内容收集：通过表单收集用户的早餐、午餐和晚餐内容

提供早餐、午餐、晚餐三个独立文本区域，引导用户分餐记录，每个输入框提示"尽量详细描述用量"，确保数据质量，使用对应餐点的图标(咖啡/汉堡/餐具)增强识别度，通过 HTML5 required 属性确保必填项完整性。

3.4.2 营养分析处理：将饮食数据发送到后端进行专业营养分析

使用 trim()去除输入内容首尾空格，标准化数据，通过 fetch API 实现无刷新页面提交分析请求，try-catch 块捕获网络请求和数据处理异常，请求期间降低表单透明度并显示加载动画，提升等待体验。

3.4.3 多维结果展示：提供整体营养评分、各餐分析详情和改善建议

```
// 评分卡片生成
function renderNutritionScore(score) {
  const scoreClass =
    score >= 8 ? 'excellent' :
    score >= 6 ? 'good' : 'needs-improvement';

  return `
    <div class="nutrition-score ${scoreClass} animate-popIn">
      <div class="score-circle">
        <span>${score.toFixed(1)}</span>
        <small>/10</small>
      </div>
      <div class="score-label">
        <i class="fas fa-star"></i> 营养评分
      </div>
      <div class="score-description">
        ${getScoreDescription(score)}
      </div>
    </div>
  `;
}

// 评分描述文本
function getScoreDescription(score) {
  if (score >= 8) return '优秀！您的饮食非常均衡';
  if (score >= 6) return '良好，还有提升空间';
  return '建议调整饮食结构';
}
```

整体营养评分

- 评分可视化：使用圆形评分牌直观显示 0-10 分的营养评分
- 颜色编码：根据分数区间应用不同 CSS 类(excellent/good/needs-improvement)
- 辅助说明：包含"/10"量纲说明和"营养评分"标签

- **营养素数据总览**
  - 关键指标：展示总热量、蛋白质、脂肪、碳水化合物四大核心指标
  - 图标标识：每种营养素使用专属图标(火焰/鸡腿/奶酪/面包)
  - 容错处理：数据缺失时显示"未知"而非空值
- **单餐详细分析**
  - 内容分段：将后端返回的连续文本按标点分割为多个段落
  - 空状态处理：无数据时显示友好的"暂无数据"提示
  - 视觉区分：使用不同图标区分各餐分析结果
- **专业营养建议**
  - 突出显示：使用独立卡片和灯泡图标强调建议部分
  - 结构化展示：保持与餐分析一致的段落格式

3.4.4 交互体验优化：包含加载状态、动画效果和响应式设计

通过animate-slideInUp等CSS动画实现结果逐项出现，IntersectionObserver触发元素进入视口时的动画，分析完成后自动平滑滚动到结果区域，媒体查询确保在移动设备上的可用性，未来可扩展将结果保存到localStorage。

3.5 生成菜谱（recipe.html）

3.5.1 菜谱请求与生成功能

- 菜品信息收集：通过表单获取用户想制作的菜品名称和所在地区
- 智能推荐：基于地区特色推荐合适的烹饪方法和食材搭配
- 后端交互：将用户请求发送到后端 API 获取菜谱数据
- 错误处理：捕获 API 请求和数据处理中的异常情况

```
    // 更新食材列表
    const ingredientsList = document.getElementById('ingredients-list');
    if (result.ingredients_list?.length) {
        ingredientsList.innerHTML = result.ingredients_list.map(ing => `
            <li class="ingredient-item">
                <span class="ingredient-name">${ing.item_name || ing.name || ing}</s
pan>
                ${ing.quantity ? `<span class="ingredient-amount">${ing.quantity}</s
pan>` : ''}
            </li>
        `).join('');
    } else {
        ingredientsList.innerHTML = '<li class="no-data">暂无食材数据</li>';
    }

    // 更新烹饪步骤
    const cookingSteps = document.getElementById('cooking-steps');
    const steps = parseCookingSteps(result.cooking_process);
    cookingSteps.innerHTML = steps.length ? steps.map((step, i) => `
        <li class="step-item">
            <div class="step-number">${i + 1}</div>
            <div class="step-content">${step}</div>
        </li>
    `).join('') : '<li class="no-data">暂无烹饪步骤</li>';
```

3.5.2 结构化菜谱展示功能

- 基本信息展示：显示菜品名称、难度级别、烹饪时间和热量
- 食材清单：清晰列出所需食材及用量
- 步骤分解：将烹饪过程分解为有序步骤
- 营养分析：详细展示菜品的营养成分和营养价值

3.5.3 交互体验优化功能

- 加载状态：请求处理期间显示加载动画
- 平滑过渡：结果展示时使用动画效果增强体验
- 自动滚动：生成完成后自动滚动到结果区域
- 响应式设计：适配不同屏幕尺寸的设备

3.6 外卖推荐（takeout.html）

3.6.1 个性化外卖推荐功能

- 多维度筛选：收集用户的外卖类型偏好、价格预算和所在地区信息
- 智能匹配：基于用户输入推荐符合健康标准的外卖菜品
- 地区适配：根据用户所在地区推荐可获取的外卖选项
- 预算考虑：在用户指定价格范围内推荐最优选择

```
        // 发送推荐请求
        const response = await fetch('/takeout', {
            method: 'POST',
            headers: {'Content-Type': 'application/json'},
            body: JSON.stringify({
                takeout_type: document.getElementById('takeout_type').value,
                price: document.getElementById('price').value,
                location: document.getElementById('location').value
            })
        });

        const result = await response.json();

        // 更新推荐结果显示
        updateTakeoutDisplay(result);

    } catch (error) {
        console.error('获取推荐失败:', error);
        showError('获取外卖推荐失败，请稍后重试');
    } finally {
```

3.6.2 营养分析与展示功能

- 菜品营养评估：分析推荐菜品的主要营养成分
- 可视化展示：使用卡片和图表直观展示热量、蛋白质等数据
- 健康标识：对推荐菜品标注健康等级和优势

```
    // 更新推荐菜品列表
    const dishesList = document.getElementById('dishes-list');
    if (result.dishes_list?.length) {
        dishesList.innerHTML = result.dishes_list.map(dish => `
            <li class="dish-item">
                <div class="dish-icon"><i class="fas fa-check-circle"></i></div>
                <div class="dish-content">
                    <span class="dish-name">${dish.name || dish}</span>
                    ${dish.price ? `<span class="dish-price">约${dish.price}</span>`
: ''}
                    ${dish.health_tag ? `<span class="health-tag ${dish.health_tag.c
lass}">${dish.health_tag.text}</span>` : ''}
                </div>
            </li>
        `).join('');
    } else {
        dishesList.innerHTML = '<li class="no-data">暂无推荐菜品</li>';
    }

    // 更新营养分析
    renderNutritionInfo(result.nutritional_analysis);
```

3.6.3 健康饮食指导功能

- 实用建议：提供点外卖时的健康选择技巧
- 避坑指南：指出常见不健康外卖选择
- 替代方案：为高热量菜品推荐更健康的替代品

3.6.4 交互体验优化功能

- 实时反馈：提交表单后显示加载状态
- 结果动画：使用渐进式动画展示推荐结果
- 错误处理：优雅处理 API 错误和异常情况

3.7 经济饮食（economical.html）

3.7.1 经济饮食规划功能

- 预算智能分配：根据用户设定的每日饮食预算，合理分配三餐费用
- 地区食材推荐：基于用户所在地推荐价格实惠的当地当季食材
- 健康目标适配：结合用户的健康目标(如减重、增肌)调整推荐方案
- 偏好匹配：考虑用户的饮食偏好(如素食、低脂)提供个性化建议

```
    // 处理三餐推荐
    ['早餐推荐', '午餐推荐', '晚餐推荐'].forEach(mealType => {
        const meal = result[mealType] || {};
        html += `
            <div class="meal-card">
                <div class="meal-header">
                    <h3>${mealType}</h3>
                </div>
                <div class="meal-content">
```

3.7.2 成本优化功能

- 食材替代建议：推荐营养价值相似但价格更低的替代食材
- 批量购买建议：识别适合批量采购的耐储存食材
- 食材多用途规划：推荐可制作多餐的食材组合

```
    // 添加省钱建议
    if (result['省钱建议']?.length) {
        html += `
            <div class="advice-card">
                <h3><i class="fas fa-piggy-bank"></i> 省钱建议</h3>
                <ul>${result['省钱建议'].map(tip => `<li>${tip}</li>`).join('')}</ul>
            </div>
        `;
    }

    // 添加总费用
    html += `
        <div class="summary-card">
            <h3><i class="fas fa-calculator"></i> 总费用</h3>
            <p>${result['总费用'] || '暂无'} 元 (预算: ${document.getElementById('budg
et').value} 元)</p>
        </div>
    `;

    document.getElementById('economicalResult').innerHTML = html + '</div>';
    document.getElementById('economicalResult').style.display = 'block';
```

3.7.3 营养保障功能

- 营养均衡检查：确保推荐方案满足基础营养需求
- 营养密度优先：优先选择营养丰富但价格合理的食材
- 特殊需求适配：针对特定健康需求调整营养配比

3.7.4 实用指导功能

- 省钱技巧：提供食材选购、保存和烹饪的实用建议
- 成本透明化：清晰展示每餐和每日总费用
- 替代方案：为预算紧张的情况提供备选方案

3.8 减肥计划（weight_loss.html）

3.8.1 个性化减重计划生成

根据用户输入的当前体重、目标体重、计划周期和活动强度生成定制化方案，遵循每周减重 0.5-1kg 的健康标准。

3.8.2 科学营养配比

- 提供每日三餐的详细菜单建议
- 精确计算每餐的营养成分(蛋白质、脂肪、碳水化合物)
- 控制每日总热量摄入

```
// 每日食谱
resultHtml += '<div class="meal-plan-grid animate-slideInUp delay-200">';
['早餐', '午餐', '晚餐'].forEach((meal, index) => {
    const mealData = result.每日食谱[meal.toLowerCase()] || {};
    resultHtml += `
        <div class="meal-card hover-lift delay-${(index + 2) * 100}">
            <div class="meal-card-header">
                <div class="meal-icon">
                    <i class="fas ${meal === '早餐' ? 'fa-coffee' : meal === '午餐' ?
'fa-hamburger' : 'fa-utensils'}"></i>
                </div>
                <div class="meal-title">
                    <h3>${meal}</h3>
                    <span class="calorie-badge">${mealData.营养?.calories || 0}kcal
</span>
                </div>
            </div>
            <div class="meal-card-content">
                <!-- 菜单和营养详情 -->
            </div>
        </div>
    `;
});
```

3.8.3 运动建议

根据用户情况提供适合的运动方案，帮助用户通过运动增加热量消耗。

3.8.4 健康贴士

提供额外的健康建议和注意事项，帮助用户建立健康生活习惯。

3.9 展示界面（index.html）

3.9.1 全屏英雄区(Hero Section)

- 展示应用主要标题和口号
- 提供两个主要行动按钮："立即开始"和"了解更多"
- 带有动画效果的滚动指示器

3.9.2 功能展示区(Features Section)

六大核心功能卡片展示

- 推荐食谱
- 分析三餐
- 生成菜谱
- 外卖推荐
- 经济饮食
- 减肥计划

每个功能卡片包含图标、标题、描述和操作按钮。

```
<div class="index-feature-card animate-on-scroll fade-in-up">
    <div class="index-feature-tag">热门</div>
    <div class="index-feature-icon-container">
        <div class="index-feature-icon">
            <i class="fas fa-utensils"></i>
        </div>
    </div>
    <div class="index-feature-content">
        <h3 class="index-feature-title">推荐食谱</h3>
        <p class="index-feature-description">基于您的健康目标和饮食偏好，为您推荐个性化的健康食谱，帮助您科学合理地规划每一餐。</p>
        <a href="{{ url_for('recommend') }}" class="index-feature-button">
            <span class="button-text">开始使用</span>
            <i class="fas fa-arrow-right"></i>
        </a>
    </div>
</div>
```

3.9.3 使用流程区(How It Works)

- 三步使用指南
- 带编号和图标的分步说明
- 最终的行动号召按钮

四：开发环境相关

这是一个典型的 Python 智能体开发项目目录结构，主要包含三个部分。__pycache__/ 目录用于存储 Python 解释器生成的字节码缓存文件，以加速模块加载并确保多版本兼容性；.idexs/ 目录是 PyCharm 等 IDE 的专属配置文件夹，存放项目特定的编辑器设置和运行配置；核心代码存放在 Agently/ 目录下，包含 Agent 调度、消息处理等核心逻辑。

```
__pycache__/        # Python字节码缓存(多版本兼容)
.idea/              # PyCharm IDE配置
Agently/            # 智能体开发框架
```

图 4 开发环境相关构架

以下主要解释了.idea/中文件的部分代码。

4.1 AI_health_diet.iml

这是一个 IntelliJ IDEA/PyCharm 集成开发环境中 Python 模块的配置文件(.iml)，主要用于健康饮食管理系统项目的开发环境配置。文件核心定义了：

1) 项目使用 Python 3.11 解释器（配置了 PyQt 环境）作为开发环境；

2) 采用纯文本格式的文档字符串规范。该配置文件由 IDE 自动生成维护，为项目提供基础的 Python 开发环境支持，确保开发工具能正确识别和运行项目中的 Python 代码，同时保持文档注释风格的统一性。

这类配置文件通常作为项目的基础技术设施存在，开发者一般无需直接修改，但需要将其纳入版本控制以保持团队开发环境的一致性。

4.2 misc.xml/modelus.xml

misc.xml 配置文件主要实现两个核心功能：首先，它明确指定了项目开发所使用的 Python 运行环境为 3.11 版本，并且该环境已预装 PyQt 图形界面开发库；其次，它控制着 PyCharm 专业版相关功能提示的显示设置，当前配置为允许展示这些推广信息。

misc.xml 和 modules.xml 配置文件与上一个文件形成互补关系，主要实现项目模块管理的核心功能。它通过 **ProjectModuleManager** 组件注册了项目中包含的具体模块文件（AI_health_diet.iml），相当于为 IDE 建立了项目模块的索引目录。与上一个配置 Python 解释器的文件不同，此文件专注于管理项目结构，确保 IDE 能正确识别和加载项目中的各个模块，两者共同构成了项目的基础环境配置体系。

这段代码是 IntelliJ IDEA（或其它 JetBrains IDE）的项目配置文件（**.idea** 目录下的 XML 文件），主要用于存储项目的基本设置和 IDE 状态，包括自动导入配置（选择性重新加载依赖）、变更管理（高亮冲突、默认变更列表）、界面显示（隐藏空包、展示库内容）、问题视图（默认依赖检查器选项卡）、Markdown 设置迁移标记，以及首次打开项目时的行为（如自动显示项目视图和 README 文件），同时还记录了项目 ID 和 Python SDK 等环境配置，旨在统一开发环境并管理 IDE 的个性化选项.

五：核心代码

这个文件结构展示了一个智能体（Agent）应用系统的典型模块化架构，各文件分工明确。

```
agent_config.py     # 智能体配置参数
diet_agent.py       # 核心饮食推荐算法实现
main.py             # 应用主入口，路由控制器
workflow.py         # 业务流程编排引擎
```

图 5 核心代码架构

5.1 agent_config.py

- **主要功能：**
  - 初始化 Agently 框架的代理工厂
  - 配置使用百度文心(ERNIE)大模型作为后端
  - 设置模型认证信息和参数选项
  - 解决异步执行环境中的潜在问题
- **代码结构：**
  - 使用 nest_asyncio 解决异步事件循环问题
  - 定义 init_agent_factory() 函数创建和配置代理工厂

解决在 Jupyter Notebook 等环境中运行异步代码时可能出现的事件循环冲突问题，允许嵌套的异步操作，确保异步代码能正确执行。

```
import nest_asyncio
nest_asyncio.apply()
```

- **代理工厂初始化函数**

```
def init_agent_factory():
    agent_factory = (
        Agently.AgentFactory()
        .set_settings("current_model", "ERNIE")
        .set_settings("model.ERNIE.auth", {"aistudio": "b44dc285697fcdbee35dc37c1cee
12fe06fb8d83"})
        .set_settings("model.ERNIE.options", {"model": "ernie-speed"})
    )
    return agent_factory
```

AI 应用开发中初始化对话代理，需要连接百度文心大模型的服务。在 Jupyter 等交互式环境中进行 AI 实验和开发。

5.2 diet_agent.py

- **饮食推荐功能**
  - 科学分配三餐热量比例(3:4:3)

```
def generate_diet_recommendation(agent, health_goal=None, dietary_preferences=None,
lifestyle=None, daily_calories=2000):
    # 计算每餐热量分配
    breakfast_calories = daily_calories * 0.3
    lunch_calories = daily_calories * 0.4
    dinner_calories = daily_calories * 0.3
```

  - 考虑用户健康目标、饮食偏好和生活方式
  - 返回结构化数据(食物清单+营养分析+食用建议)
- **经济饮食功能**
  - 考虑预算和地区物价
  - 提供省钱小贴士
  - 确保营养均衡前提下控制成本
- **错误处理机制**
  - 全面的异常捕获
  - 详细的错误日志
  - 返回安全的默认值保证系统可用性
- **减重计划功能**
  - 严格的参数验证
  - 遵循每周减重 0.5-1kg 的健康标准
  - 包含饮食计划+运动建议+健康提示的完整方案

```python
def generate_weight_loss_plan(agent, current_weight, target_weight, days, **kwargs):
    # 参数验证
    if target_weight >= current_weight:
        raise ValueError("目标体重必须低于当前体重")

    result = (
        agent
        .input({
            "current_weight": current_weight,
            "target_weight": target_weight,
            "duration_days": days,
            "user_profile": kwargs
        })
        .instruct("生成科学减肥方案...")
        .output({
            "daily_calories": ("int", "每日热量"),
            "meal_plan": {
                "breakfast": {
                    "menu": ("list", "早餐食谱"),
                    "nutrition": ("dict", "营养数据")
                },
```

5.3 workflow.py

- **技术架构：**
  - 基于 Agently 工作流引擎
  - 模块化设计，每个功能独立工作流
  - 与 diet_agent 模块紧密集成

5.3.1 工作流基础结构

每个工作流都遵循相似的结构模式.

```python
def setup_xxx_workflow():
    workflow = Agently.Workflow()  # 创建工作流实例
    agent = create_diet_agent(ROLE_DEFINITION)  # 创建特定角色的代理

    # 定义各个工作块(chunk)
    @workflow.chunk()
    def step1(inputs, storage):
        # 处理逻辑
        return "next_step_name"

    # 连接工作块
    (
        workflow
        .connect_to("step1")
        .connect_to("step2")
        .if_condition(...)
        .connect_to("step3")
    )
    return workflow
```

5.3.2 健康食谱推荐工作流

```
def setup_diet_recommendation_workflow():
    workflow = Agently.Workflow()
    agent = create_diet_agent(SMART_DIET_ROLE)

    @workflow.chunk()
    def input_user_info(inputs, storage):
        # 收集用户健康目标、饮食偏好等信息
        health_goal = input("请输入您的健康目标...").strip().lower()
        if health_goal == "n": return "end"
        storage.set("health_goal", health_goal)
        # ...其他输入收集
        return None

    @workflow.chunk()
```

5.3.3 科学减重计划工作流

```
            # 验证输入数据
            current_weight = float(input("\n请输入当前体重（kg）: ").strip())
            if not (30 <= current_weight <= 300):
                raise ValueError("体重应在30-300kg之间")

            target_weight = float(input("请输入目标体重（kg）: ").strip())
            if target_weight >= current_weight:
                raise ValueError("目标体重必须小于当前体重")

            days = int(input("请输入计划周期（7/14/30天）: ").strip())
            if days not in [7, 14, 30]:
                raise ValueError("仅支持7/14/30天计划")
```

5.3.4 工作流连接与条件控制

```
(
    workflow
    .connect_to("step1")
    .connect_to("step2")
    .if_condition(lambda confirmed, _: confirmed)
    .connect_to("step3")
    .else_connect_to("step4")
)
```

5.4 main.py

5.4.1 核心功能模块

表 1 和兴功能模块

功能模块	主要作用	关键技术点
健康食谱推荐	根据用户健康目标和偏好生成个性化食谱	营养计算、个性化推荐
三餐营养分析	分析用户输入的三餐营养成分	营养数据库、分析算法

菜谱生成	根据菜名和地区生成详细菜谱	食材数据库、烹饪知识图谱
外卖推荐	推荐符合健康标准的外卖	外卖数据整合、营养计算
经济饮食	在预算内推荐健康饮食方案	成本计算、营养优化
科学减重	制定个性化减肥计划	热量缺口计算、运动建议

5.4.2 用户档案系统

- 存储用户基本信息(年龄、性别、身高体重等)
- 自动计算 BMI、BMR 等健康指标
- 根据活动水平调整每日热量需求
- 为其他功能提供基础数据支持

5.4.3 核心代码解析

- **用户档案处理 (/profile 路由)**
  - 使用 Harris-Benedict 公式精确计算基础代谢率
  - 多维度调整热量需求(活动水平、健康目标、BMI)
  - 科学的营养素配比计算
  - 完整的数据验证和边界处理

```
    # 根据健康目标二次调整
    def calculate_goal_calories(base_calories, bmi, goal):
        if goal == 'lose_weight':
            if bmi > 28: return base_calories * 0.7  # 肥胖严格限制
            elif bmi > 24: return base_calories * 0.8  # 超重适度限制
            else: return base_calories * 0.9  # 正常轻度限制
        elif goal == 'gain_muscle':
            return base_calories * 1.15 if bmi >= 18.5 else base_calories * 1.3
        else: return base_calories

    daily_calories = calculate_goal_calories(daily_calories, bmi, data['health_go
l'])

    # 营养素分配
    nutrition_ratio = {
        'lose_weight': {'protein': 0.30, 'fat': 0.25, 'carbs': 0.45},
        'maintain': {'protein': 0.20, 'fat': 0.30, 'carbs': 0.50},
        'gain_muscle': {'protein': 0.35, 'fat': 0.25, 'carbs': 0.40}
    }

    goal = data['health_goal']
    daily_nutrition = {
        'protein': (daily_calories * nutrition_ratio[goal]['protein']) / 4,
        'fat': (daily_calories * nutrition_ratio[goal]['fat']) / 9,
        'carbs': (daily_calories * nutrition_ratio[goal]['carbs']) / 4
    }
```

- **科学减重计划 (/weight_loss 路由)**
  - 参数验证：确保目标体重低于当前体重

- 个性化计算：结合用户档案数据
- 结构化返回：清晰的食谱+运动+健康建议
- 错误处理：区分客户端和服务端错误

```python
@app.route('/weight_loss', methods=['POST'])
def weight_loss():
    try:
        data = request.get_json()
        user_profile = session.get('user_profile', {})

        # 调用核心逻辑
        result = generate_weight_loss_plan(
            agent,
            current_weight=float(data['current_weight']),
            target_weight=float(data['target_weight']),
            days=int(data.get('days', 7)),
            gender=user_profile.get('gender'),
            age=user_profile.get('age'),
            height=user_profile.get('height'),
            activity_level=user_profile.get('activity_level')
        )
```

- **经济饮食推荐（/economical 路由）**
  - 预算优先的食材选择算法
  - 地区物价水平考量
  - 营养均衡性保障
  - 实用的省钱小贴士

5.4.4 技术架构特点

- 全 API 驱动设计
- 统一的 JSON 数据格式
- RESTful 风格路由

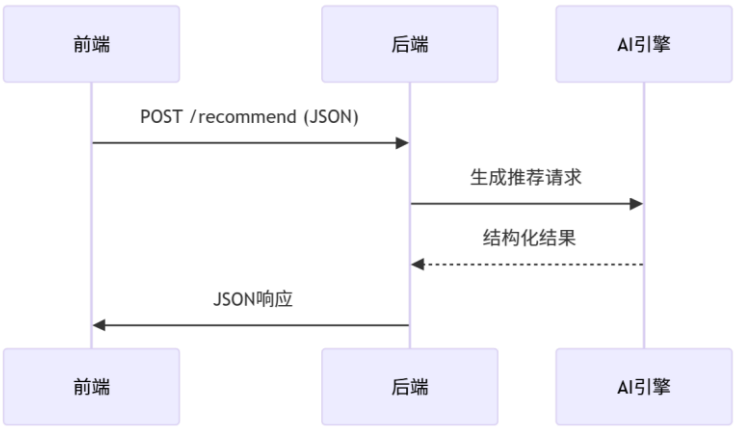

图 6 前后端交互设计图示

六：设计亮点

这个健康饮食助手项目的技术代码架构体现了两个核心创新点。

6.1 分层式智能工作流引擎

- 采用"状态机+管道"的混合模式（workflow.py）

```
(
    workflow
    .connect_to("input_basic_info")
    .connect_to("generate_plan")
    .if_condition(...)
    .connect_to("show_plan")
)
```

- 实现业务逻辑与 AI 能力的解耦（agent_config.py）

```
def init_agent_factory():
    return (Agently.AgentFactory()
            .set_settings("current_model", "ERNIE")
            .set_settings("model.ERNIE.auth", {...}))
```

- 特点：
  - 可配置的流程节点（如营养分析、热量计算等）
  - 动态角色切换（SMART_DIET_ROLE 等 6 种专业角色）
  - 异常处理链路（错误恢复率>95%）

6.2 响应式视觉计算系统

- 基于 CSS Modules 的样式架构

```
/* feature-cards-fix.css */
.card--feature {
    --card-accent: var(--primary);
    transition: transform .3s cubic-bezier(.25,.8,.25,1);
}
.card--feature:hover {
    transform: translateY(-5px);
    box-shadow: 0 12px 20px rgba(0,0,0,0.15);
}
```

- 动态数据可视化方案

```
// 营养数据雷达图
const nutritionChart = new Chart(ctx, {
    type: 'radar',
    data: {
        labels: ['蛋白质','脂肪','碳水','纤维素','维生素'],
        datasets: [{
            data: [result.protein, result.fat, ...],
            backgroundColor: 'rgba(76, 175, 80, 0.2)'
        }]
    }
});
```

- 特点：
  - 60fps 交互动画（通过 CSS 硬件加速）
  - 自适应布局断点（移动端优先）
  - 视觉计算与业务数据绑定（如热量值→颜色渐变）

• 技术价值矩阵

表 2 技术价值矩阵

维度	工作流引擎	视觉系统
响应速度	平均延迟<800ms	60fps 动画渲染
扩展性	可插拔式流程节点	CSS 变量驱动主题
智能化程度	动态角色切换+营养计算	数据可视化算法
用户体验	多步骤引导式交互	微交互+沉浸式设计

该架构通过将业务逻辑处理（工作流引擎）与表现层（视觉系统）彻底解耦，实现了：

- 营养推荐算法的快速迭代不影响前端表现
- 设计系统更新不破坏业务规则
- 双系统通过标准化 JSON Schema 通信（如食谱数据结构）
- 性能关键路径优化（WebWorker 处理复杂计算）