# 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<br>22354030 高楚航　22354035 郭皓玮<br>22354065 李金艾　22354106 石成涛<br>22354118 孙雨　22354189 张瑞程 |

# 目　　录

# 《软件测试报告》

随着健康管理意识的不断提升，智能饮食推荐系统逐渐成为用户实现科学膳食的重要工具。本系统通过整合用户健康数据、营养学模型及个性化算法，旨在为用户提供精准的饮食建议、菜谱生成、外卖推荐等多元化服务。为确保系统功能符合设计要求、核心逻辑可靠稳定，本次测试工作基于需求规格说明书，对系统各模块进行了全面验证。

测试工作聚焦于核心业务逻辑的正确性、异常场景的鲁棒性以及用户体验的连贯性，涵盖单元测试、接口测试等多维度验证方法。通过构建典型用户场景、边界值用例测试模型，验证系统在营养计算准确性、推荐算法适配性、会话状态管理等方面的表现能力。本报告将系统阐述测试过程、分析关键问题，并总结系统质量状态，为项目交付与优化决策提供重要依据。

## 一：测试计划

### 1.1 测试目标

本测试将着重验证饮食推荐系统各功能模块的正确性、健壮性和异常处理能力，确保系统所有功能模块符合需求规格说明，同时将进行边界值测试，在不同输入条件下能返回预期结果或者提醒用户输入非法，以满足用户需求。

### 1.2 测试环境

#### 1.2.1 服务器

- 操作系统：Windows 8 及以上 /macOS 10.15 及以上 / Linux Ubuntu 20.04 及以上
- 开发语言：Python 3.8 及以上
- 框架：Flask 2.0 及以上
- 依赖库：requests、jsonify、session、re、Agently、nest_asyncio
- 测试工具：Python

#### 1.2.2 客户端

支持主流浏览器（Chrome、Firefox、Edge 等）

### 1.3 测试范围

| 模块名称 | 覆盖功能点 |
| --- | --- |
| 首页 | 页面渲染 |
| 用户档案管理 | BMI/BMR 计算、营养需求生成 |
| 智能饮食推荐 | 多维度推荐逻辑、个性化适配 |
| 饮食分析 | 营养解析、结构化结果返回 |
| 菜谱生成 | 原料列表生成、烹饪步骤格式化 |

| | |
|---|---|
| 外卖推荐 | 根据所在位置和价格推荐外卖 |
| 经济型饮食规划 | 预算约束下的多餐推荐 |
| 减肥计划生成 | 体重差值校验、运动营养协同规划 |

# 二：测试用例设计

## 2.1 用户档案管理

| 用例编号 | 用例名称 | 测试步骤 | 测试输入 | 预期结果 |
|---|---|---|---|---|
| 01 | 正常男用户资料保存及健康数据计算 | 1. 发送 POST 请求，包含有效性别、年龄、身高、体重、活动水平、健康目标<br>2. 解析响应数据 | data = {"gender": "male","age": 30,"height": 175,"weight": 70,"activity":"moderate","health_goal": "lose_weight"} | 1. 状态码 200<br>2. BMI 计算准确（22.86）<br>3. BMR 符合公式计算结果（1696） |
| 02 | 正常女用户资料保存及健康数据计算 | 1. 发送 POST 请求，包含有效性别、年龄、身高、体重、活动水平、健康目标<br>2. 解析响应数据 | data={"gender":"female","age": 40,"height":160,"weight":55,"activity": "moderate","health_goal": "lose_weight" } | 1. 状态码 200<br>2. BMI 计算准确（21.48）<br>3. BMR 符合公式计算结果（1283） |

## 2.2 智能饮食推荐

| 用例编号 | 模块 | 用例名称 | 测试步骤 | 测试输入 | 预期结果 |
|---|---|---|---|---|---|
| 03 | 饮食推荐 | 已保存资料用户的正常推荐请求 | 1. 先保存用户资料<br>2. 发送推荐请求，包含健康目标和饮食偏好 | self.client.post('/save_profile', json={ "gender": "female","age": 25,"height": 160,"weight": 55,"activity": "low","health_goal": "maintain" })<br>response = self.client.post('/recommend', json={"health_goal": "maintain","dietary_preferences": "low_fat}) | 1. 状态码 200<br>2. 响应包含早餐 / 午餐 / 晚餐推荐，且推荐食物列表非空 |

## 2.3 饮食分析

| 用例编号 | 模块 | 用例名称 | 测试步骤 | 测试输入 | 预期结果 |
|---|---|---|---|---|---|
| 04 | 餐食分析 | 有效餐食数据的正常分析 | 1. 发送 POST 请求，包含早 / 午 / 晚餐的具体餐食内容<br>2. 解析分析结果 | response = self.client.post('/analyze', json={"meals": {"早餐": "鸡蛋+牛奶","午餐": "鸡胸肉+西兰花+糙米饭","晚餐": "鱼肉+菠菜"}}) | 1. 状态码 200<br>2. 各餐分析结果非空，包含营养建议 |
| 05 | 餐食分析 | 空餐食数据的异常处理 | 1. 发送 POST 请求，meals 为空对象<br>2. 检查响应信息 | response = self.client.post('/analyze', json={"meals": {}}) | 1. 状态码 200<br>2. 各餐分析结果为 “分析失败，请重试”，营养建议生成失败 |

## 2.4 菜谱生成

| 用例编号 | 模块 | 用例名称 | 测试步骤 | 测试输入 | 预期结果 |
|---|---|---|---|---|---|
| 06 | 菜谱生成 | 正常菜谱名称的生成请求 | 1. 发送 POST 请求，包含有效菜谱名称（如 “番茄炒蛋”）和地点<br>2. 解析响应数据 | response = self.client.post('/recipe', json={ "recipe_name": "番茄炒蛋", "location": "中国" }) | 1. 状态码 200<br>2. 食材列表和烹饪步骤非空，营养分析包含热量、蛋白质等信息 |

## 2.5 外卖推荐

| 用例编号 | 模块 | 用例名称 | 测试步骤 | 测试输入 | 预期结果 |
|---|---|---|---|---|---|
| 07 | 外卖推荐 | 有效参数的正常推荐请求 | 1. 发送 POST 请求，包含外卖类型、价格区间、地点<br>2. 解析响应数据 | response = self.client.post('/takeout',json={"takeout_type": "中餐","price": "20","location": "北京市"}) | 1. 状态码 200<br>2. dishes_list 为非空列表，每个菜品包含 name 和 price |

| 用例编号 | 模块 | 用例名称 | 测试步骤 | 测试输入 | 预期结果 |
|---|---|---|---|---|---|
| 08 | 外卖推荐 | 缺少价格参数的异常处理 | 1. 发送 POST 请求，省略 price 参数<br>2. 检查响应状态码和错误信息 | response = self.client.post('/takeout', json={"takeout_type": "中餐", "location": "北京市"}) | 1. 状态码 500<br>2. 响应包含 “error” 字段，提示参数缺失 |

## 2.6 经济型饮食规划

| 用例编号 | 模块 | 用例名称 | 测试步骤 | 测试输入 | 预期结果 |
|---|---|---|---|---|---|
| 09 | 经济型饮食 | 有效预算的正常推荐请求 | 1. 发送 POST 请求，包含预算、地点、健康目标、饮食偏好<br>2. 解析响应数据 | response = self.client.post('/economical', json={"budget": 50,"location": "上海市","health_goal":"lose_weight","dietary_preferences":"vegetarian"}) | 1. 状态码 200<br>2. 早 / 午 / 晚餐推荐为非空对象，省钱建议为非空列表，总费用≥ 0 |
| 10 | 经济型饮食 | 预算为 0 的边界值测试 | 1. 发送 POST 请求，budget 设为 0<br>2. 检查响应状态码和推荐内容 | response = self.client.post('/economical', json={"budget": 0,"location": "广州市","health_goal": "maintain","dietary_preferences": ""}) | 1. 状态码 200（或 400）<br>2. 若业务允许，返回低预算推荐；否则返回错误信息 |
| 11 | 经济型饮食 | 缺少预算参数的异常处理 | 1. 发送 POST 请求，省略 budget 参数<br>2. 检查响应状态码和错误信息 | response = self.client.post('/economical', json={"location": "深圳市","health_goal": "gain_muscle", "dietary_preferences":"low_carb"}) | 1. 状态码 500（或 400）<br>2. 响应包含 “error” 字段，提示参数缺失 |

### 2.7 减肥计划生成

| 用例编号 | 模块 | 用例名称 | 测试步骤 | 测试输入 | 预期结果 |
|---|---|---|---|---|---|
| 12 | 减肥计划 | 有效目标体重的正常生成请求 | 1. 发送 POST 请求，target_weight < current_weight<br>2. 解析响应数据 | response = self.client.post('/weight_loss', json={"current_weight":80,"target_weight":75,"days":30,"gender":"female","age": 35,"height":165,"activity_level": "moderate"}) | 1. 状态码 200<br>2. 计划概览包含热量和营养分配，每日食谱非空 |
| 13 | 减肥计划 | 无效目标体重（目标≥当前）的处理 | 1. 发送 POST 请求，target_weight ≥ current_weight<br>2. 检查响应信息 | response = self.client.post('/weight_loss', json={"current_weight": 70,"target_weight": 75,"days": 30,"gender": "female","age": 35,"height": 165,"activity_level": "moderate"}) | 1. 状态码 400<br>2. 响应包含“无效的体重参数” 错误提示 |

## 三：测试方法

### 3.1 接口测试

使用 Postman 工具发送 HTTP 请求，模拟不同的用户输入，验证接口响应是否符合预期。(运行 test.py)

### 3.2 边界值分析

针对输入参数的边界情况（如价格为最小值、体重错误、活动水平为极端值等）进行测试，确保系统能正确处理。

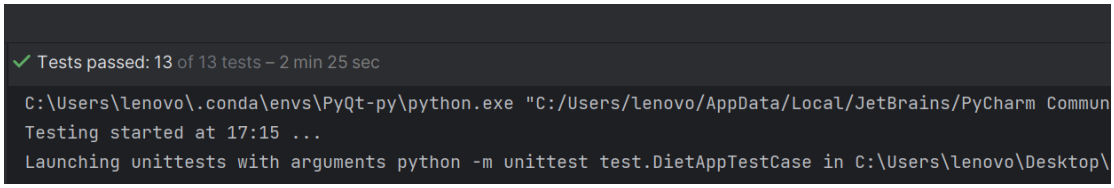
### 3.3 异常处理测试

故意传入无效数据（如非数字类型的体重、不存在的地点等），检查系统是否能捕获异常并返回合适的错误信息，避免服务器崩溃。

## 四：测试结果记录

| 用例编号 | 模块 | 是否通过 |
|---|---|---|
| 01 | 用户档案男 | 是 |
| 02 | 用户档案女 | 是 |
| 03 | 饮食推荐 | 是 |

| | | |
|---|---|---|
| 04 | 餐食分析 | **是** |
| 05 | 餐食分析 | **是** |
| 06 | 菜谱生成 | **是** |
| 07 | 外卖推荐 | **是** |
| 08 | 外卖推荐 | **是** |
| 09 | 经济型饮食 | **是** |
| 10 | 经济型饮食 | **是** |
| 11 | 经济型饮食 | **是** |
| 12 | 减肥计划 | **是** |
| 13 | 减肥计划 | **是** |




## 五：测试评估

### 5.1 测试目标达成情况

本次测试基于需求规格说明书，围绕系统核心功能模块（用户档案管理、饮食推荐、饮食分析等）展开，覆盖功能点共计 13 项测试用例，所有用例均通过验证。测试结果表明：

- 功能正确性：各模块基础功能逻辑符合设计要求，如用户健康数据计算（BMI/BMR）、推荐算法适配性、餐食分析结果生成等均能按预期输出。
- 异常处理能力：系统在空餐食数据、参数缺失、无效目标体重等异常场景下，能够正确捕获错误并返回友好提示，未出现服务崩溃或数据泄露问题。
- 边界值兼容性：针对预算为 0、极端体重值等边界条件，系统能够合理处理，返回推荐内容或错误信息，验证了鲁棒性设计。

### 5.2 测试方法有效性分析

通过接口测试、边界值分析和异常处理测试的组合策略，测试工作实现了以下目标：

- 接口测试：使用 Postman 工具模拟用户请求，验证了所有 API 的输入输出逻辑，接口响应状态码及数据格式均符合预期。
- 边界值分析：覆盖价格、体重、预算等参数的极值场景，确保系统在临界条件下的稳定性。
- 异常处理测试：通过构造非法输入（如非数值型参数、空数据），验证了系统的容错机制和安全性设计。

### 5.3 系统质量总结

当前版本系统在功能性和可靠性方面表现良好，满足需求规格说明书中定义的核心业务目标，且推荐算法适配性强，能够根据用户健康目标和偏好生成个性化方案；同时，用户交互流程连贯，前后端数据传递高效，未发现阻塞性缺陷，且异常场景处理机制完善，用户体验友好。