湖南大学实训报告

课程名称：计算与人工智能概论B 实验类型：团队实训

案例名称：食谱生成系统

组长姓名：文子豪 班级：给水2502 学号：202501120209

同组姓名1： 班级： 学号：

同组姓名2： 班级： 学号：

一、实训目的

1．巩固计算与人工智能概念课程所学基础知识

2．拓展训练学生计算和AI 思维能力

3．加强团队分工与合作，培养学生团队协作能力

二、案例内容和团队分工

本次实验旨在设计并实现一个智能化的*食谱生成系统*。该系统需要解决的核心问题是：如何根据用户个性化、动态的约束条件（如现有食材、口味偏好、烹饪时间等），自动生成合理、可行且多样化的食谱方案，以解决日常饮食规划中的选择困难问题。具体任务要求如下：

系统设计：运用计算思维，将食谱生成问题分解为数据管理、约束处理、推荐算法等模块。

数据构建：创建一个结构化的本地食谱数据库（如使用JSON或SQLite文件），包含菜名、所需食材、烹饪步骤、口味、耗时、难度等关键属性。

核心功能实现：开发一个程序（例如使用Python），使其能够：输入：接收用户输入的“现有食材列表”和“偏好要求”（如“素食”、“快手菜”）。处理：基于算法，从数据库中筛选和匹配符合条件的食谱。输出：向用户呈现一份或多份完整的食谱建议，包括食材清单和烹饪指南。

界面与展示：通过命令行界面或简单的Web界面实现与用户的交互，并展示完整的任务完成过程。

三、案例分析

（一）、问题分解：将复杂问题拆解为可管理的子任务。

1、数据层：如何存储和管理食谱及食材信息？

2、输入层：如何获取并解析用户的需求（有什么、想吃什么）？

3、逻辑层：如何根据输入，从所有食谱中找出最合适的？规则是什么？（例如：匹配最多现有食材的食谱优先；完全符合饮食禁忌的食谱才能入选）。

4、输出层：如何将结果清晰、友好地呈现给用户？

（二）、模式识别：寻找问题中的规律和模式。

1、食谱模式：每个食谱都可抽象为 {菜名， 食材集合， 步骤， 标签集合...} 的数据结构。

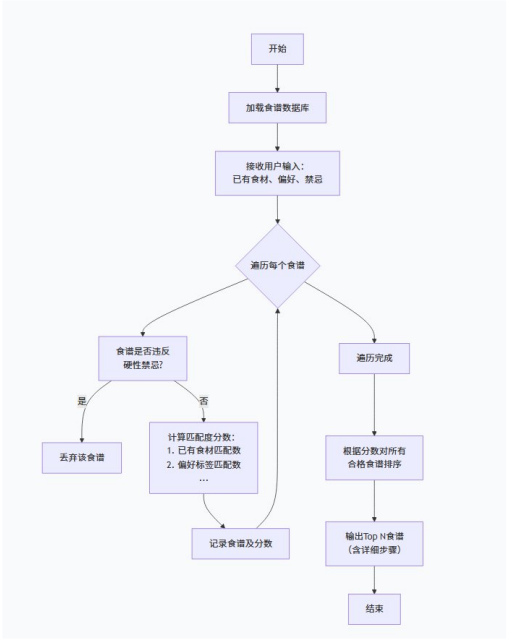
2、匹配模式：用户输入可以抽象为 {已有食材集合， 需求标签集合（如“辣”、“<30分钟”）}。

3、筛选模式：核心逻辑是一个“过滤-排序”模式。过滤掉所有包含禁忌食材或不满足硬性标签的食谱；然后根据匹配到的已有食材数量、口味偏好符合度等规则进行排序。

（三）、抽象与算法设计：将解决方案抽象为通用模型和步骤。

1、抽象：将食谱和用户输入都抽象为带标签的集合。问题转化为集合匹配与优先级排序问题。

2、算法设计（流程图表示）



四、案例实现与展示

步骤1：设计并创建食谱数据库

==================== 第一部分：数据准备 ====================

print("🍳 食谱生成系统初始化...")

print("=" * 60)

recipes = [{"id": 1,"name": "西红柿炒鸡蛋","ingredients": ["西红柿", "鸡蛋", "盐", "油", "糖"],"steps": ["1. 西红柿洗净切块", "2. 鸡蛋打散加少许盐", "3. 热油炒鸡蛋盛出", "4. 炒西红柿至出汁", "5. 加入鸡蛋翻炒均匀"],"tags": ["家常菜", "快手菜", "下饭菜"],"time": 15},{id": 2, "name": "青椒肉丝","ingredients": ["猪里脊肉", "青椒", "生抽", "料酒", "淀粉", "盐", "油"],"steps": ["1. 猪肉切丝，加料酒、淀粉腌制", "2. 青椒切丝", "3. 热油滑炒肉丝至变色", "4. 加入青椒丝翻炒", "5. 加生抽和盐调味"], "tags": ["家常菜", "下饭菜"],"time": 25 },{ "id": 3, "name": "蛋炒饭","ingredients": ["米饭", "鸡蛋", "盐", "油", "葱花"],"steps": ["1. 鸡蛋打散", "2. 热油炒鸡蛋", "3. 加入米饭翻炒", "4. 加盐调味", "5. 撒葱花出锅"], "tags": ["快手菜", "主食"],"time": 10}, { "id": 4, "name": "凉拌黄瓜","ingredients": ["黄瓜", "蒜", "醋", "生抽", "盐", "香油"], "steps": ["1. 黄瓜拍碎切段", "2. 蒜切末", "3. 所有调料混合", "4. 淋在黄瓜上拌匀"],"tags": ["凉菜", "快手菜", "健康"],"time": 5}, {"id": 5, "name": "番茄鸡蛋面","ingredients": ["面条", "西红柿", "鸡蛋", "盐",

"油", "青菜"],"steps": ["1. 煮面条", "2. 同时做西红柿炒鸡蛋", "3. 面条煮好过冷水", "4. 浇上西红柿鸡蛋卤"],"tags": ["主食", "快手菜"],"time": 20}]

print(f"✓ 数据库加载完成！共有 {len(recipes)} 个食谱")

print("=" * 60)

步骤2：编写程序

```
def find_recipes_by_ingredients(user_ingredients):
    """
    根据用户食材匹配食谱
    返回按匹配度排序的食谱列表
    """
    results = []
    user_set = set(user_ingredients)

    for recipe in recipes:
        recipe_set = set(recipe['ingredients'])

        # 计算匹配的食材数量
        matched = len(user_set & recipe_set)
        total = len(recipe_set)

        # 计算匹配百分比
        if total > 0:
            match_percent = (matched / total) * 100
        else:
            match_percent = 0

        # 找出缺少的食材
        missing = list(recipe_set - user_set)

        # 只显示匹配度超过30%的食谱
        if match_percent >= 30:
            results.append({
                'recipe': recipe,
                'match_score': match_percent,
                'matched_count': matched,
                'total_needed': total,
                'missing_ingredients': missing
```

```
                })

        # 按匹配度从高到低排序
        results.sort(key=lambda x: x['match_score'], reverse=True)
        return results
print("\n🎯 请输入您现有的食材")
print("示例：鸡蛋 西红柿 盐 油 米饭")
print("提示：用空格分隔不同食材")
print("-" * 50)

# 获取用户输入
user_input = input("请在此输入：")
user_ingredients = user_input.strip().split()

print(f"\n🔍 正在分析您的食材：{user_ingredients}")
print("正在智能匹配最佳食谱...")

# 运行匹配算法
matched_recipes = find_recipes_by_ingredients(user_ingredients)
) print("\n" + "=" * 60)
print("📋 推荐结果")
print("=" * 60)

if len(matched_recipes) == 0:
        print("😅 很抱歉，没有找到合适的食谱")
        print("💡 建议：添加更多基础食材（如油、盐、酱油等）")
else:
        print(f"🎉 为您找到 {len(matched_recipes)} 个推荐食谱：\n")

        for i, result in enumerate(matched_recipes, 1):
                recipe = result['recipe']

                # 使用表情符号让输出更生动
                print(f"{i}. 【{recipe['name']}】")
                print(f"    ⭐ 匹配度：{result['match_score']:.0f}%
({result['matched_count']}/{result['total_needed']})")
                print(f"    ⏱️ 预计时间：{recipe['time']}分钟")
                print(f"    🏷️ 标签：{', '.join(recipe['tags'])}")

                if result['missing_ingredients']:
                        print(f"    🛒 需要购买：{', '.join(result['missing_ingredients'])}")
                else:
```

```
            print("    ✅ 食材齐全！")

        # 显示简要做法
        print(f"    👩‍🍳 简要做法：{recipe['steps'][0]}")
        print()
```

步骤3：运行与展示，终端运行截图示例：

text

```
$ python recipe_generator.py
为您推荐以下食谱：

1. 西红柿炒鸡蛋
   匹配食材：4种 | 缺失食材：糖，葱花
   标签：家常，快手菜，素食，咸鲜 | 耗时：15分钟
```

五、案例拓展与练习

（一）、案例拓展

1、营养均衡考量：在数据库中添加每道菜的卡路里、蛋白质、碳水等营养信息，算法在推荐时尝试满足用户的每日营养目标。

2、智能替换建议：当食谱缺失某样食材时，系统能基于风味相似性推荐替代品。

3、利用机器学习：收集用户对推荐食谱的反馈例如点击、收藏、评分等，训练一个排序模型，实现个性化推荐。

（二）、案例练习

1、基础题：修改程序，使其能根据用户输入的“最大烹饪时间”进行筛选。

2、进阶题：实现一个功能，当用户输入“西红柿、鸡蛋、牛肉”时，系统能推荐“西红柿炒鸡蛋”+“葱爆牛肉”这样的一荤一素组合套餐，并计算组合的总耗时和缺失食材。

六、总结

1、学习了软件系统的分析与设计过程。通过本次实验，掌握了如何将一个实际生活问题（吃什么）分解为数据、逻辑、交互等多个模块进行系统性解决。

2、实践了计算思维的核心方法。成功运用了问题分解、模式识别、抽象与算法设计，将模糊的需求转化为清晰的“过滤-排序”算法和具体可运行的代码。