**《花开福德系统》**

**详细设计**

**课程名称： 软件项目综合实践**

**指导教师： 梁琦**

北京信息科技大学

|  |  |  |
| --- | --- | --- |
| 文件状态：  [ ] 草稿  [√] 正式发布  [ ] 正在修改 | 文件标识： | LLD-FOOD |
| 当前版本： | 1.0 |
| 作 者： | 赵圣杰，胡西贝，李羿霖 |
| 完成日期： | 2024.10.14 |

# 引言

## 编写目的

本详细设计说明书旨在为花开福德美食推荐系统的开发提供全面的技术设计方案和功能实现指南。通过详细说明每个模块的功能、实现方式、使用的技术栈、框架选择及各模块之间的接口，我们可以确保项目在开发过程中具有明确的目标、功能分解、技术实现路径以及优化策略，帮助项目团队清晰理解项目的核心目标。

本文档的预期读者包括：项目管理人员，设计开发人员，测试人员以及系统的用户。

## 范围

该文档的目的是解决整个项目系统的“怎么做”的问题。在这里，主要是根据用户提出的项目需求进行的全面设计，包括接口设计详细描述和功能模块的详细设计描述。

## 项目背景

在快速变化的美食文化领域，随着人们对生活品质追求的不断提升，传统美食文化的传承与创新成为了社会关注的焦点。消费者在面对琳琅满目的美食选择时，往往面临信息过载、选择困难等问题。同时，美食爱好者们渴望深入了解各地美食背后的文化故事、制作工艺，以及获取个性化的美食推荐，以满足其日益多样化的口味需求。为了响应这一市场需求，弘扬中华美食传统文化，提升用户体验，方便用户发现和享受美食，北京信息科技大学食全食美责任有限公司决定开发一套集美食文化传承、个性化推荐、内容展示与互动功能于一体的“福德系统”。该系统旨在通过数字化手段，打破地域限制，拓宽美食传播的渠道，促进美食文化的交流与共享。

* 待开发的软件系统的名称为：《北京信息科技大学食全食美责任有限公司花开福德系统》（以下简称为：福德系统）。
* 本项目的任务提出者为：北京信息科技大学食全食美责任有限公司
* 开发地点：北京信息科技大学（沙河校区）XXB-402，
* 本系统用户：16-80岁的有动手能力，以及信息检索能力的人。

## 版本更新信息

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 版本 | 作者 | 修改日期 | 修改位置 | 修改内容 |
| 1.0 | 赵圣杰  胡西贝  李羿霖 | 2024-10-14 | 1，3，5部分  4部分  2部分 | 初始发布版本 |

# 架构设计

## 系统架构概述

本系统是一个典型的 Python Web 应用程序，由多个模块组成，分为应用主文件、配置文件、扩展模块、数据模型和控制逻辑。这些模块通过 Flask 框架进行集成与协作，以提供一个功能完备且可扩展的 Web 后端服务。该系统采用 Flask 框架构建，模块化设计使得每个部分的职责清晰可见，易于维护和扩展。以下是对系统中各个模块的功能简要介绍：

1.app.py：主应用入口，负责创建应用实例并整合各个模块。

2.config.py：负责系统的配置管理，包括数据库、密钥等。

3.exts.py：用于初始化各类扩展，包括数据库和缓存等。

4.models.py：定义数据库模型，表示数据结构。

5.views.py：实现应用的业务逻辑和接口。

## 模块介绍

### app.py

app.py是系统的主入口，包含创建 Flask 应用实例的代码。它负责整合所有其他模块并启动应用服务。

主要功能包括：

- 创建 Flask 应用实例。

- 加载配置（引用 `config.py`）。

- 初始化扩展（引用 `exts.py`）。

- 注册蓝图（引用 `views.py` 中定义的路由）。

通过这种设计，`app.py` 实现了对整个系统的初始化和协调。

### config.py

config.py文件包含系统的配置信息，主要用于存储数据库连接字符串、密钥以及其他配置项。

配置项的作用包括：

- 开发环境和生产环境的区分：通过不同的配置类，例如 `DevelopmentConfig` 和 `ProductionConfig`，可以轻松切换环境。

- 数据库配置：提供数据库连接 URI，供扩展模块使用。

这使得系统配置变得灵活且集中，易于修改和管理。

### exts.py

exts.py主要用于初始化系统中的各类扩展，如数据库、缓存等。通过这种方式，应用程序与扩展的初始化解耦，便于管理和测试。

- 数据库扩展：通常会使用 `SQLAlchemy` 或其他类似工具来管理数据库连接。

- 其他扩展：如缓存机制、用户认证等，也在这里进行初始化。

`exts.py` 确保了各类扩展在应用实例创建时得到正确初始化。

### models.py

models.py负责定义系统的数据模型。使用 `SQLAlchemy` 等工具来描述数据库中的实体结构。

主要功能：

- 定义数据表结构：如用户表、订单表等。

- 提供数据库操作方法：如 CRUD（创建、读取、更新、删除）操作。

通过这种设计，`models.py` 把数据逻辑与业务逻辑分离，使得代码更具可读性与可维护性。

### views.py

views.py包含应用的控制逻辑和路由定义，处理用户请求并返回相应的响应。

主要功能包括：

- 定义路由：使用 Flask 的装饰器定义每个 API 的路由，例如 `@app.route('/user')`。

- 处理请求逻辑：调用 `models.py` 中的数据操作方法，处理用户请求。

- 返回响应：根据请求结果返回合适的 JSON 数据或页面。

这种视图层的设计使得业务逻辑清晰明了，易于扩展和修改。

## 模块交互关系

- app.py是整个系统的核心，负责初始化应用和加载所有模块。

- config.py为系统提供配置支持，exts.py和views.py都依赖于它。

- exts.py负责初始化所有扩展，供models.py和views.py使用。

- models.py提供的数据模型被views.py使用，用于处理数据库中的数据。

这种模块化设计使得每个模块职责明确，减少了模块之间的耦合度，提升了系统的可维护性。

## 技术细节

### 应用实例初始化

在 `app.py` 中，Flask 应用实例通过如下步骤进行初始化：

1. \*\*创建 Flask 应用实例\*\*：

app = Flask(\_\_name\_\_)

2. \*\*加载配置\*\*：从 `config.py` 中加载相应的配置类：

app.config.from\_object('config.DevelopmentConfig')

3. \*\*初始化扩展\*\*：调用 `exts.py` 中的扩展初始化函数。

from exts import db

db.init\_app(app)

4. \*\*注册蓝图\*\*：将 `views.py` 中定义的蓝图注册到应用实例：

from views import main\_blueprint

app.register\_blueprint(main\_blueprint)

### 数据库配置与扩展

在 `config.py` 中，数据库连接配置使用 SQLAlchemy URI 的形式提供，例如：

SQLALCHEMY\_DATABASE\_URI = 'mysql+pymysql://user:password@localhost/dbname'

在 `exts.py` 中，数据库扩展通过如下方式进行初始化：

from flask\_sqlalchemy import SQLAlchemy

db = SQLAlchemy()

然后在 `app.py` 中调用 `db.init\_app(app)` 以将数据库与应用实例关联。

### 视图和路由的实现

在 `views.py` 中，控制逻辑通过路由进行定义，例如：

from flask import Blueprint, request, jsonify

from models import User

main\_blueprint = Blueprint('main', \_\_name\_\_)

@main\_blueprint.route('/users', methods=['GET'])

def get\_users():

users = User.query.all()

return jsonify([user.to\_dict() for user in users])

这个视图函数负责处理 `/users` 路径的 GET 请求，调用 `models.py` 中的 `User` 模型来获取数据并返回 JSON 响应。

## 部署与运行

1. \*\*安装依赖\*\*：通过 `pip` 安装所有依赖包：

pip install -r requirements.txt

2. \*\*设置环境变量\*\*：设置 Flask 的环境变量以指定应用入口：

export FLASK\_APP=app.py

export FLASK\_ENV=development

3. \*\*运行应用\*\*：使用 Flask 内置服务器运行应用：

flask run

# 功能模块设计

根据项目需求规格说明书，设计系统功能模块，功能模块分为用户管理模块，个性化推荐模块，美食介绍模块，美食评分评论模块，美食榜单模块，美食信息管理模块，评论信息管理模块以及美食搜索模块。系统功能模块设计如图3-1所示。

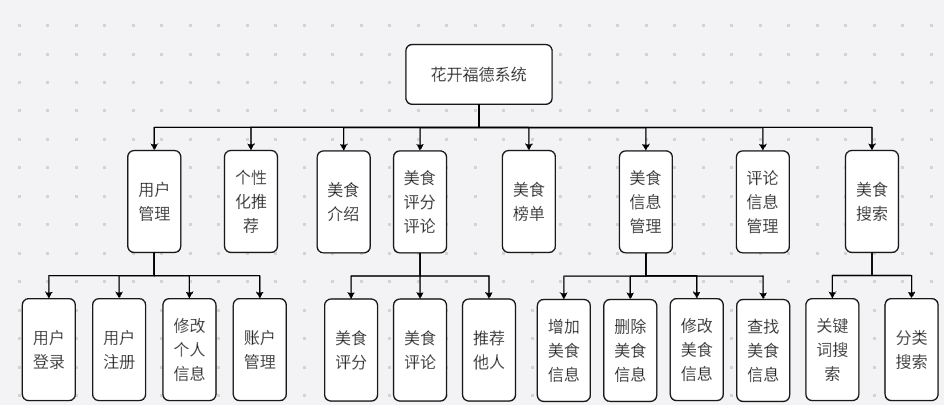


图3-1 功能模块设计图

1.用户管理模块主要负责用户注册登录，修改个人信息，以及管理员对用户的管理。

2.个性化推荐模块主要负责根据用户的行为（搜索历史，浏览记录，收藏夹等）对用户进行个性化美食推荐。

3.美食介绍模块主要负责提供美食信息，包括简介，美食图片，历史背景，口味特点，菜系等等还会提供美食的家常做法。

4.美食评分评论模块主要负责美食的评分评论，用户可以对美食进行评分和评论，系统会将评分和评论显示给他人，以供参考。

5.美食榜单模块可以根据不同条件（评分，推荐人数等）显示受欢迎的美食排行榜，还可以根据时间段（每日，每月，每季度等）生成不用榜单，根据不同美食类型（口味，菜系等）生成不同榜单。

6.美食信息管理模块负责对美食信息的管理，管理员可以对美食信息进行增删改查。

7.评论信息管理模块负责对美食的评论信息的管理，管理员可以对美食的评论信息进行增删改查。审核不当言论并进行删除。

8.美食搜索功能负责对美食信息的查找，包括关键词查找和分类查找，关键词查找用户直接输入关键词进行查找，分类查找用户根据系统给出的分类进行组合查找。

* 1. 用户管理模块详细设计

### 用户登录详细设计

**功能简介：**用户登录属于用户管理模块，用户输入自己的账号和密码，系统在后台数据库进行查询操作后，返回布尔值，表示该输入是否正确，输入正确则进入系统,错误则对用户进行相应提示。

**操作流程：**

1.用户在登录页面输入用户名和密码，点击登录按钮，触发登录请求。

2.接收到用户提交的登录请求后，后端首先通过数据库查找用户名，获取该用户的加密密码。

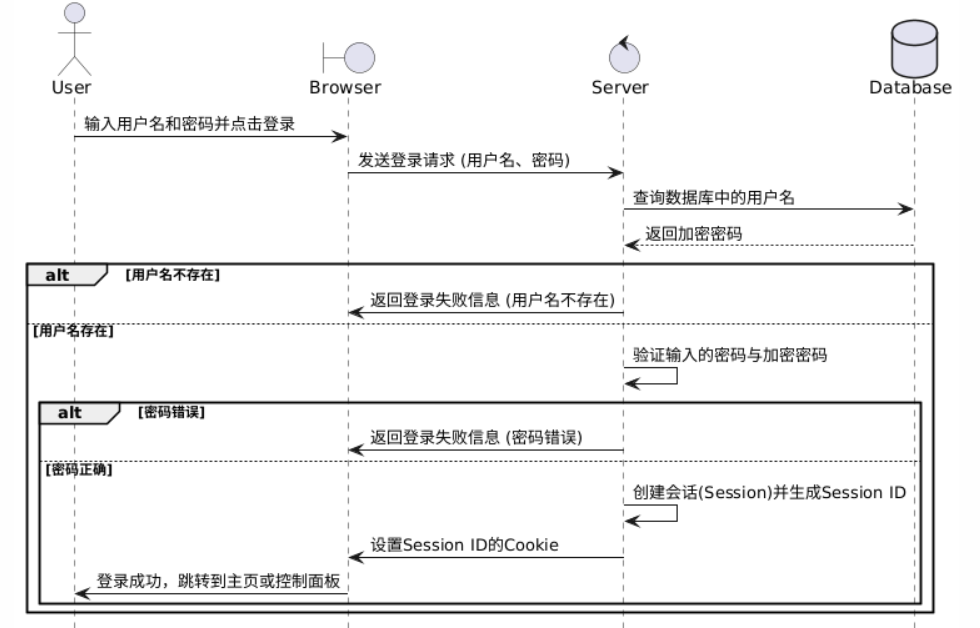
3.验证用户输入的用户名和密码是否与数据库中存储的用户信息匹配。如果用户名和密码匹配成功，则继续执行下一步；如果匹配失败，则返回登录失败信息。

4.如果密码验证通过，后端创建一个会话并设置一个cookie返回给客户端，cookie中包含会话的唯一标识符（session ID）。

5.服务器返回登录成功的响应，前端跳转到用户的主页或控制面板。

6.在用户每次请求时，服务器通过读取cookie中的会话ID来判断用户的登录状态。

**时序图：**



### 用户注册详细设计

**功能简介：**用户注册模块属于用户管理模块，用户输入自己的用户名和密码，系统会先在数据库中查找是否已经存在，如果不存在，系统会在后台数据库录入用户输入的用户名及密码并生成新的账户，返回布尔值，表示用户创建是否成功，成功则返回注册成功，若已被注册则返回注册失败。

**操作流程：**

1.用户访问注册页面并填写用户名、密码和邮箱等信息。

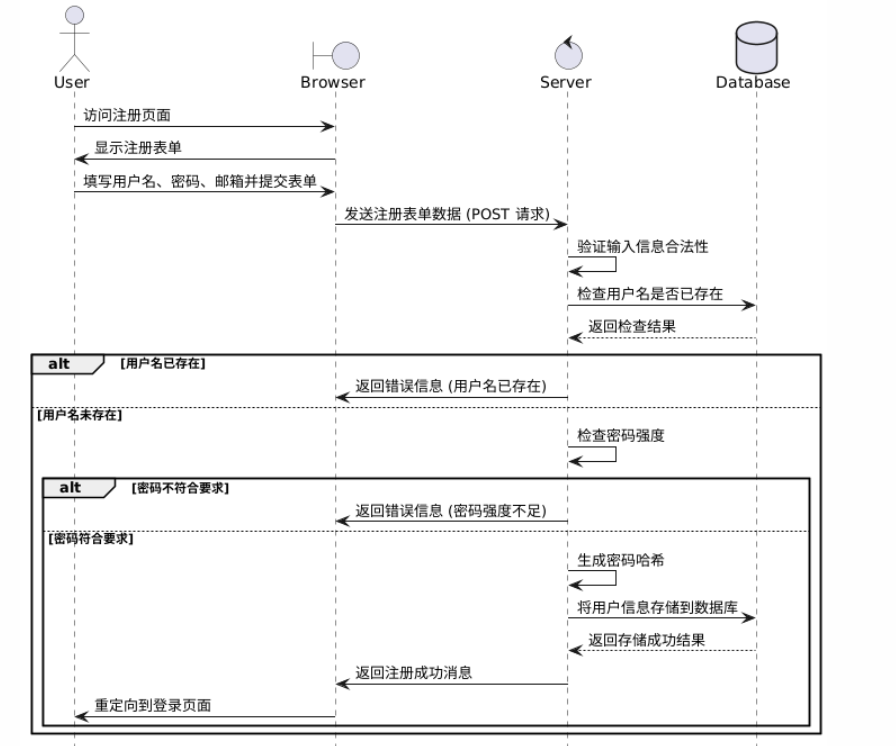
2.提交注册表单，服务器接收请求。

3.服务器验证输入信息的合法性，检查用户名是否已存在，检查密码的强度（长度、复杂性等）。

4.如果所有验证通过，将用户信息（用户名、哈希密码）存储到数据库。

5.返回注册成功的反馈信息，并重定向到登录页面。

**时序图：**



### 修改个人信息详细设计

**功能简介：**用户可以通过该模块修改个人信息，包括用户名、密码、性别、口味偏好、头像等等。但是用户不能上传自定义头像，只能从系统预设的头像列表中选择。该模块需要验证用户的身份，确保只有登录的用户可以修改个人信息。

**操作流程：**

1. 用户进入“修改个人信息”页面，系统显示当前的个人信息（如用户名、性别、口味偏好等）。

2. 用户在表单中输入需要修改的个人信息，并选择新的头像。

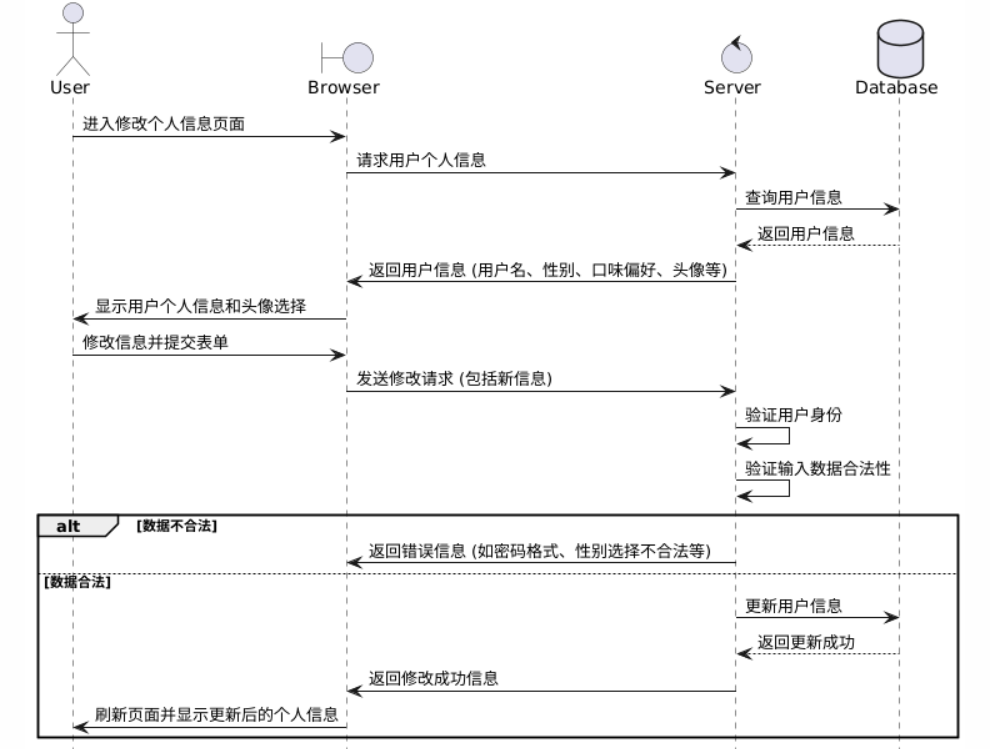
3. 提交表单，服务器接收到修改请求，验证用户身份并更新信息。

4. 验证输入信息的合法性（如密码格式、性别选择、口味偏好等）。

5. 验证通过后，更新用户的个人信息到数据库。

6. 返回修改成功的反馈信息，用户的个人信息页面刷新显示更新后的数据。

**时序图：**



### 账户管理详细设计

**功能简介：**管理员可以通过该模块进行用户账户的管理，包括查看、添加、编辑和删除用户账户。该模块允许管理员管理用户的角色和权限，以维护系统的安全性和有效性。

**操作流程：**

1.管理员登录系统，进入管理员控制面板。

2.管理员选择“用户账户管理”功能。

3.系统显示当前用户账户列表，包含用户的基本信息（如用户名、角色状态等）。

4.管理员可以选择以下操作：

4.1查看用户详情：

1）管理员点击某个用户，查看该用户的详细信息。

4.2添加新用户：

1）管理员点击“添加用户”，填写必要信息（用户名、密码等）。

2）提交信息，系统进行数据验证，确保输入合法性。

3）验证通过后，将新用户信息存储到数据库。

4.3编辑用户信息：

1）管理员点击某个用户的“编辑”按钮。

2）修改用户的相关信息（如角色、权限、状态等）。

3）提交修改，系统验证数据的合法性。

4）验证通过后，更新数据库中的用户信息。

4.4删除用户账户：

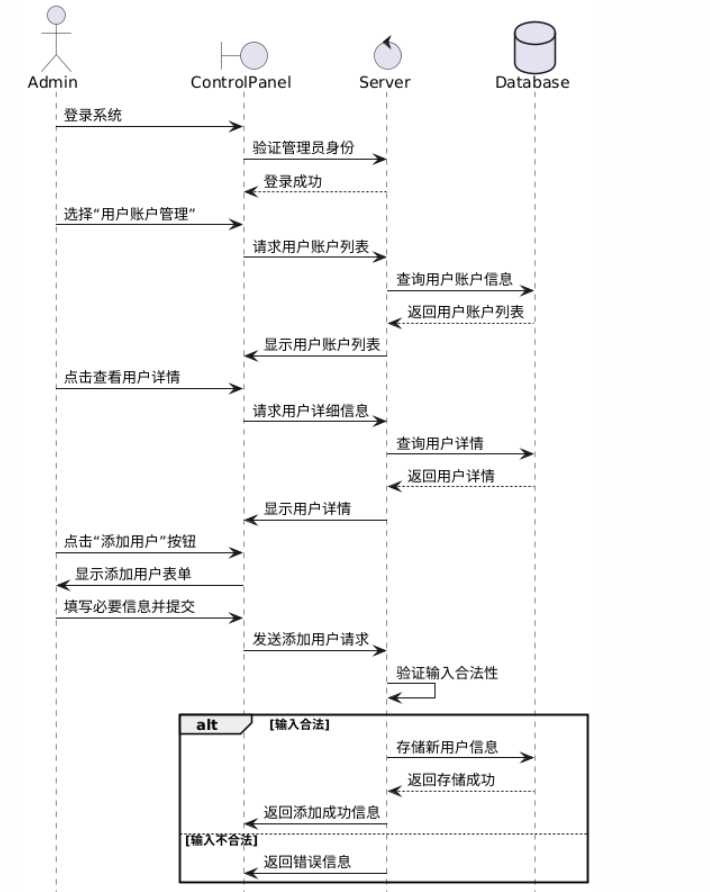
1）管理员点击某个用户的“删除”按钮。

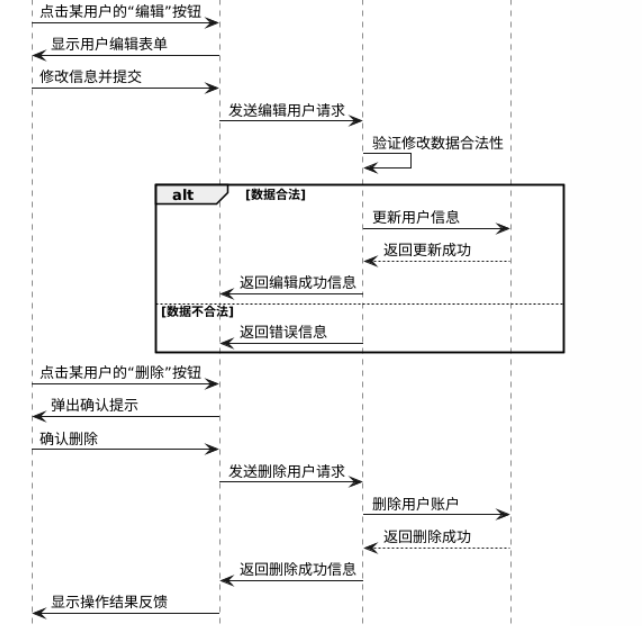
2）系统弹出确认提示，确保管理员确认删除操作。

3）确认后，系统从数据库中删除该用户账户。

5.系统在每个操作后返回操作结果的反馈信息，确保管理员了解操作是否成功。

**时序图：**





## 个性化推荐模块详细设计

**功能简介：**个性化推荐模块旨在根据用户的行为数据，为用户提供量身定制的美食推荐，从而提高用户体验和满意度。

**模块说明：**

1.数据收集：系统记录用户的搜索历史、浏览记录、收藏夹、评分和评论等信息。定期分析用户的偏好变化，更新用户画像。

2.用户画像：系统收集用户的基本信息，包括地区、饮食习惯、过敏信息等。通过用户的行为数据，为每个用户生成个性化兴趣标签（如偏好辣味、素食等）。

3.推荐算法

协同过滤:

基于用户的协同过滤: 通过分析相似用户的喜好，推荐相似用户 喜欢的美食。

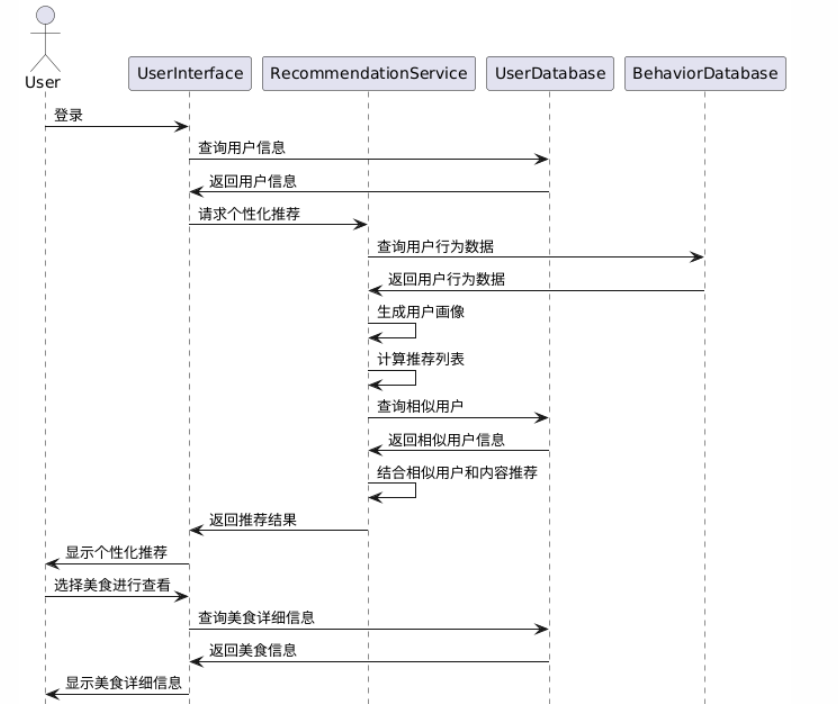
基于物品的协同过滤: 分析美食之间的相似性，推荐与用户喜欢 的美食相似的其他美食。

内容推荐: 根据用户喜欢的美食的特征（如口味、菜系、食材等）推 荐类似特征的其他美食。

混合推荐: 综合使用协同过滤和内容推荐，以提高推荐的准确性和多 样性。

4.推荐生成：根据用户的最新行为实时更新推荐列表。

**时序图：**



## 美食介绍模块详细设计

**功能简介：**美食介绍模块旨在为用户提供详细的美食信息，包括美食的名称、简介、背景、制作方法，美食图片和其他相关内容，从而帮助用户更好地了解和选择美食。

**模块说明：**

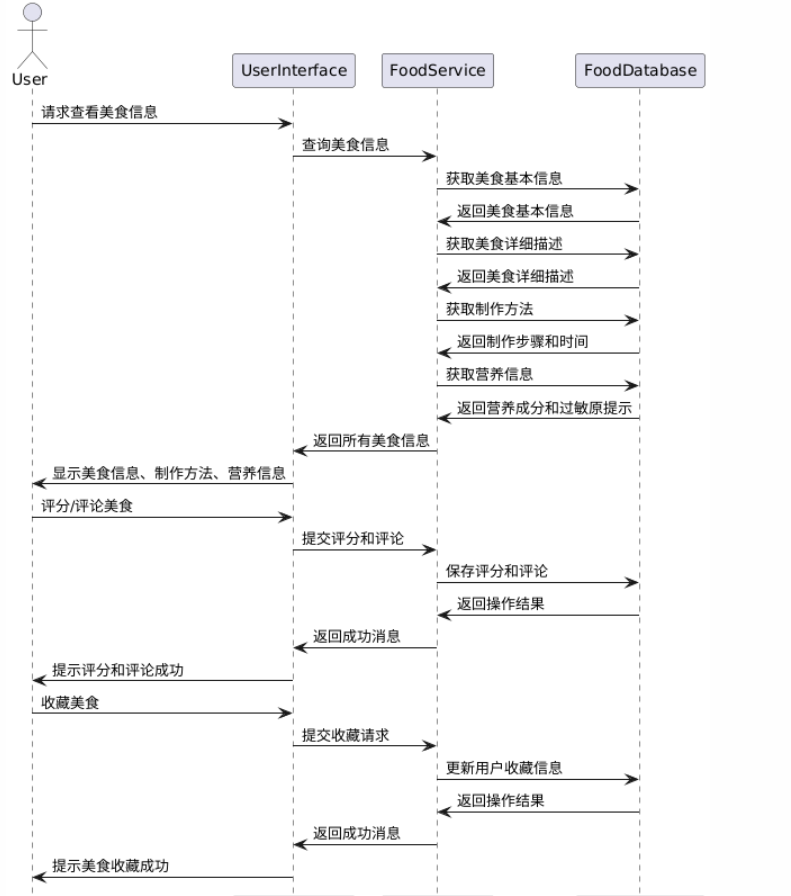
1.美食信息展示：展示美食基本信息，包括美食名称、图片、菜系、口味、主料、配料等基本信息。供美食的详细介绍，包括历史背景、文化意义、地域特色等。

2.制作方法：提供详细的制作步骤，每一步可附带图片或视频说明，帮助用户更直观地了解制作过程。标明所需的准备时间、烹饪时间和总时间，以帮助用户合理安排时间。

3.营养信息：列出美食的营养成分，包括热量、蛋白质、脂肪、碳水化合物等信息，帮助用户关注健康饮食。提示用户可能的过敏成分（如坚果、海鲜等），确保用户在选择时的安全。

4.用户互动：用户可以对美食进行评分和评论，分享自己的制作体验和反馈。用户还可以将感兴趣的美食加入收藏，方便以后查找。

**时序图：**



* 1. 美食评分评论模块详细设计

### 美食评分详细设计

**功能简介：**美食评分模块允许用户对平台上的美食进行评分和评论，以帮助其他用户做出更好的选择，同时为平台提供反馈，提升美食推荐的质量。

**模块说明：**

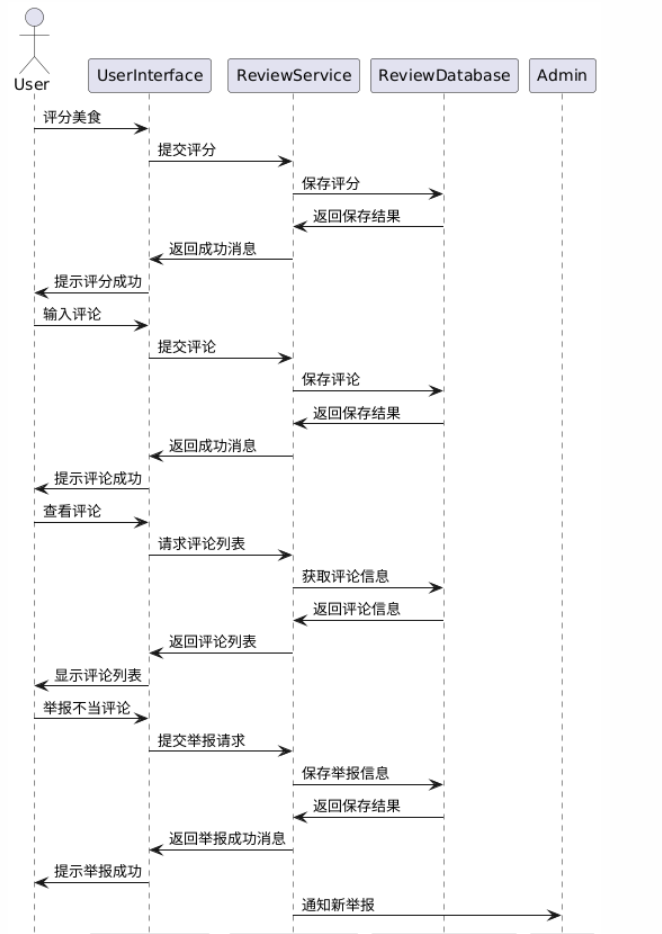
1.用户评分：提供1到5星的评分系统，用户可以根据自己的体验为美食打分。用户只能对每种美食评分一次，若用户已评分，则需先删除或修改原评分才能重新评分。

2.用户评论：用户可以在美食详情页面输入自己的评论，包括对美食的评价、制作过程的感受等。支持基本文本格式，允许用户添加换行、表情等。

3.评论列表：在美食详情页面展示所有用户评论，包括用户名、评分、评论内容和时间戳。提供根据时间、评分高低等排序和筛选功能，帮助用户快速找到有用的评论。

4.评论审核：系统允许管理员对用户评论进行审核，删除不当评论（如恶意评论、广告等）。用户可以举报不当评论，系统记录举报信息供管理员审核。

**时序图：**



### 美食推荐详细设计

**功能简介：**美食推荐模块允许用户对感兴趣的美食进行推荐，帮助其他用户发现受欢迎的美食，同时收集用户偏好数据以优化个性化推荐。

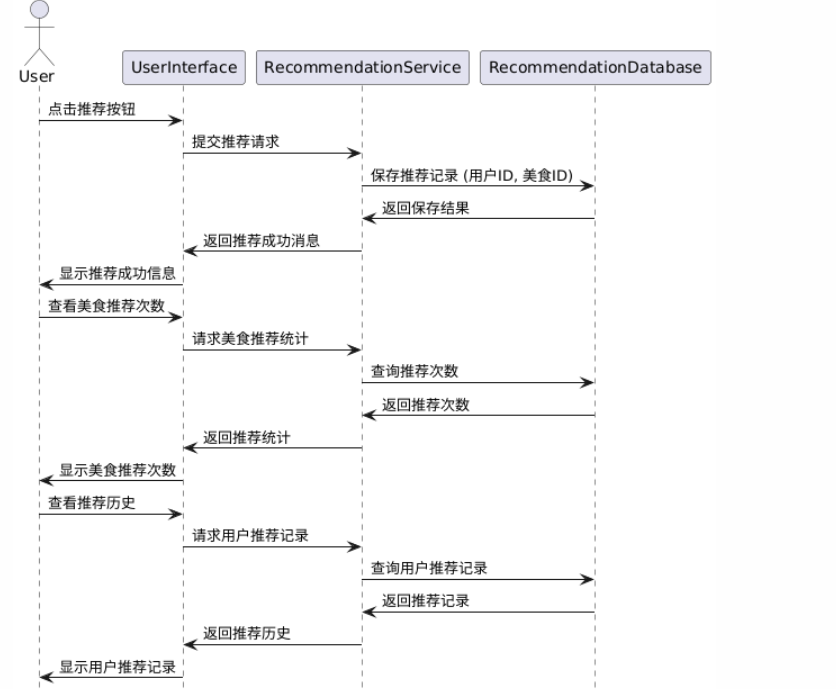
**模块说明：**

1.推荐功能：在美食详情页面和美食列表中设置“推荐”按钮，用户点击后可以将该美食推荐给其他用户。用户点击推荐按钮后，系统显示反馈信息（如“推荐成功！”），并更新推荐计数。

2.推荐统计：在美食详情页面展示该美食的推荐次数，作为参考指标，帮助用户判断美食受欢迎程度。用户可以查看自己推荐过的美食列表，方便管理自己的推荐记录。

3.推荐记录管理：每次用户推荐的操作都会在数据库中记录，包括用户ID、美食ID等信息。

**时序图：**



## 美食榜单详细设计

**功能简介：**美食榜单模块旨在根据用户评分、推荐次数、评论数量等指标生成美食排行榜，帮助用户快速发现受欢迎和高评分的美食。

**模块说明：**

1.榜单类型

热门美食榜单: 根据推荐次数和评论数量生成的排行榜，反映用户的 推荐意愿和兴趣。

高评分美食榜单: 根据用户评分生成的排行榜，展示评分最高的美食， 帮助用户找到质量较高的选择。

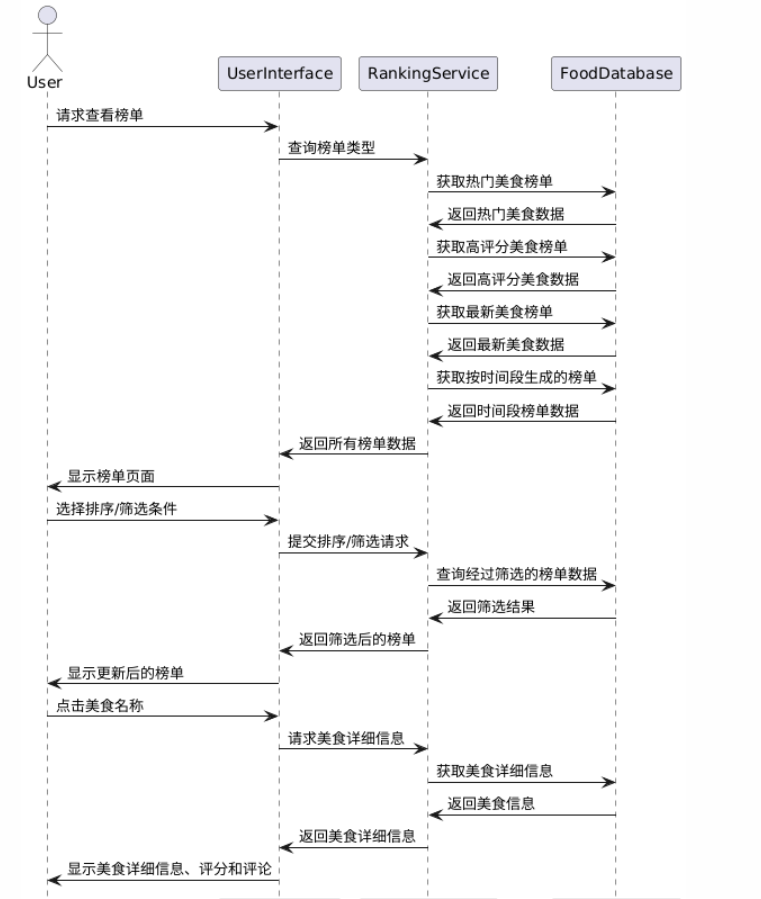
最新美食榜单: 根据美食添加到平台的时间生成的排行榜，展示近期 新增的受欢迎美食。

按时间段生成榜单: 支持按日、周、月等时间段生成不同榜单，帮助 用户查看不同时期的热门美食。

2.榜单展示：在榜单页面展示美食的基本信息，包括美食名称、图片、评分、推荐次数等。提供用户根据不同条件（如评分、推荐次数、时间，口味等）对榜单进行排序和筛选的功能。

3.用户互动：用户可以点击榜单中的美食名称，查看该美食的详细信息、评分和评论。

**时序图：**



## 美食信息管理详细设计

**功能简介：**美食信息管理模块允许管理员对美食信息进行全面管理，包括新增、删除、查看和修改美食信息。管理员能够确保平台上的美食信息准确、完整，并符合用户需求。

**操作流程：**

1.管理员通过用户名和密码登录系统。系统检查登录凭证，确认用户是否具备管理员权限。

2.登录成功后，管理员在后台管理界面选择“美食信息管理”模块。

3.系统显示当前美食信息列表，包含美食的基本信息（如美食名称、图片、简介、口味等）。

4.管理员可以选择以下操作：

4.1新增美食信息

1）点击“添加”按钮，进入新增美食信息页面

2）系统返回一个表单以输入美食信息，包括美食名称、图片、简 介、历史背景、口味特点等等。

3）输入完成后，点击提交，系统必填字段和格式填写正确，图片 格式符合要求等。

4）验证通过，信息保存至数据库，并返回列表页面。

4.2修改美食信息

1）管理员可以在列表中选择需要修改的美食，点击编辑，进入编 辑页面。

2）在编辑页面，管理员可以修改已有的美食信息，并提交更新。

3）系统进行数据验证，确保信息完整性和正确性。

4）若验证通过，修改后的信息更新至数据库，返回列表页面。

4.3删除美食信息

1）管理员可以在列表中选择需要删除的美食，点击删除按钮。

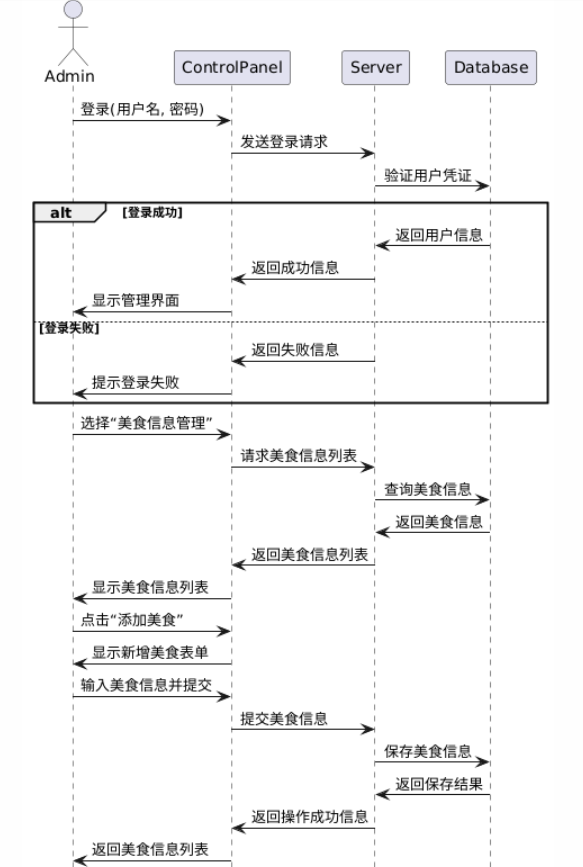
2）弹出确认对话框，询问管理员是否确定删除该美食，避免误操 作，确认后执行删除。

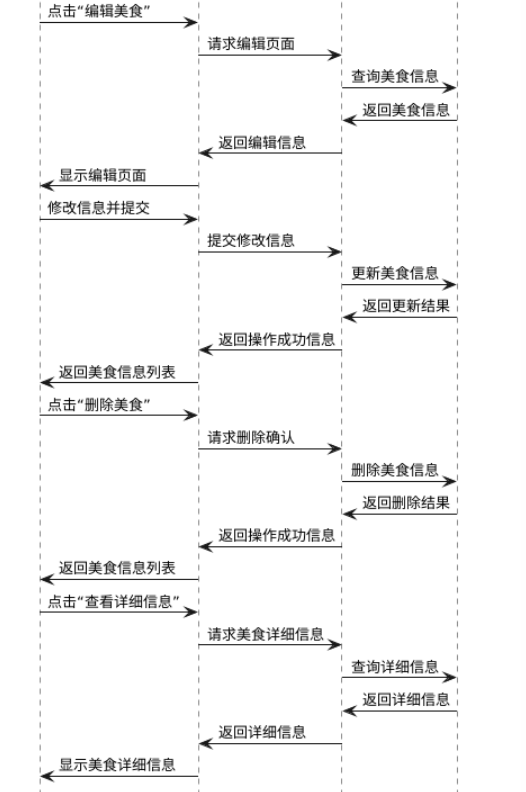
3）确认后，系统从数据库中删除该美食信息，并返回更新后的列 表页面。

4.4查看美食详细信息

1）管理员可以点击美食名称进入详细信息页面，查看该美食的所 有信息，包括简介，图片，历史背景、口味、评分评论等。

**时序图：**





## 评论信息管理详细设计

**功能简介：**评论信息管理模块允许管理员对用户的美食评论进行增删查改操作，确保评论内容的质量和合规性。管理员可以审核不当言论，进行删除或修改。

**模块说明：**

1.查看评论信息：管理员在后台可以查看所有用户的美食评论信息。包括评论ID，用户ID，美食ID，评论内容，评论时间。系统还提供按时间、美食、用户等条件筛选评论的功能，帮助管理员高效管理评论。支持根据关键字快速搜索评论，方便定位特定评论。

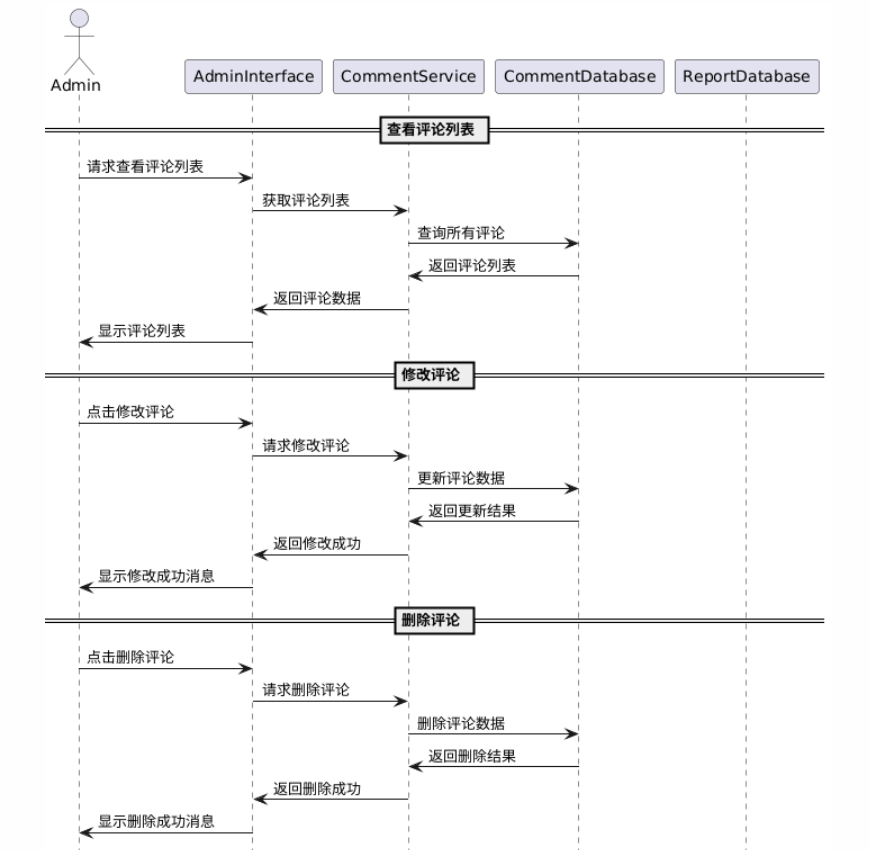
2.修改评论信息：管理员可以选择特定评论，进入编辑页面，对评论内容进行修改。

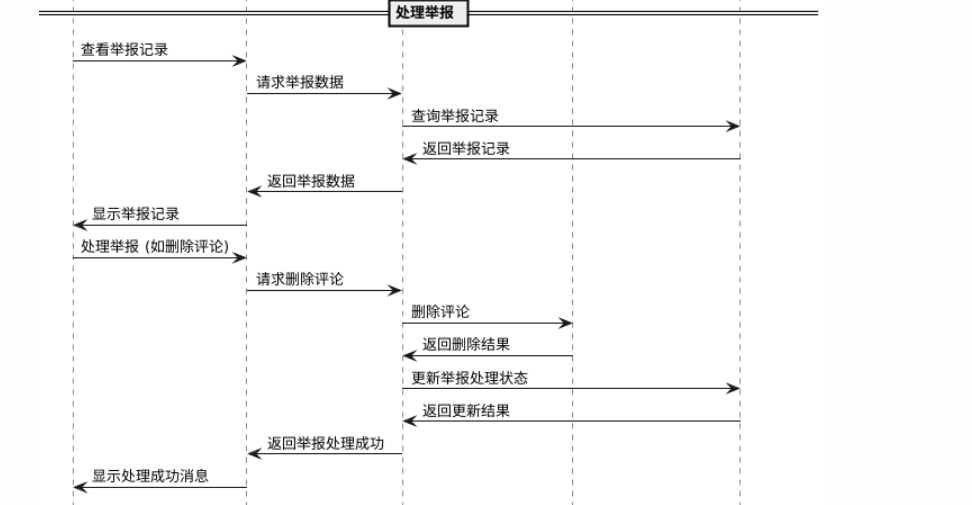
3.删除评论信息：管理员可以删除不当评论或不符合社区规定的评论。删除操作需要弹出确认框，防止误操作。支持批量选择多个评论进行删除操作，方便管理员处理大量不当评论。

4.举报处理：管理员可以查看用户举报的评论记录，包括举报原因、举报用户、举报时间等。管理员可以对被举报的评论进行审核，决定是否删除或修改评论。处理完毕后，管理员可以选择将处理结果反馈给举报用户。

5.评论状态管理：管理员可以给评论设置不同状态，例如“已审核”、“待处理”、“已删除”等，方便后续管理。对所有被修改或删除的评论记录，系统保留审核历史，管理员可以随时查阅。

**时序图：**





## 美食搜索详细设计

### 关键词搜索

**功能简介：**关键词搜索模块旨在让用户通过输入关键词，快速检索平台上的美食信息。该功能支持模糊搜索、精准搜索、以及根据美食名称、菜系、口味、历史背景等字段进行检索。

**模块说明：**

1.搜索框：用户可以在页面顶部或专门的搜索页面输入关键词进行搜索。系统保留用户的搜索历史，方便用户快速进行重复搜索。

2.搜索类型

模糊搜索：用户输入部分关键词时，系统根据关键词匹配美食名称、 菜系、口味、简介等字段，返回相关结果。

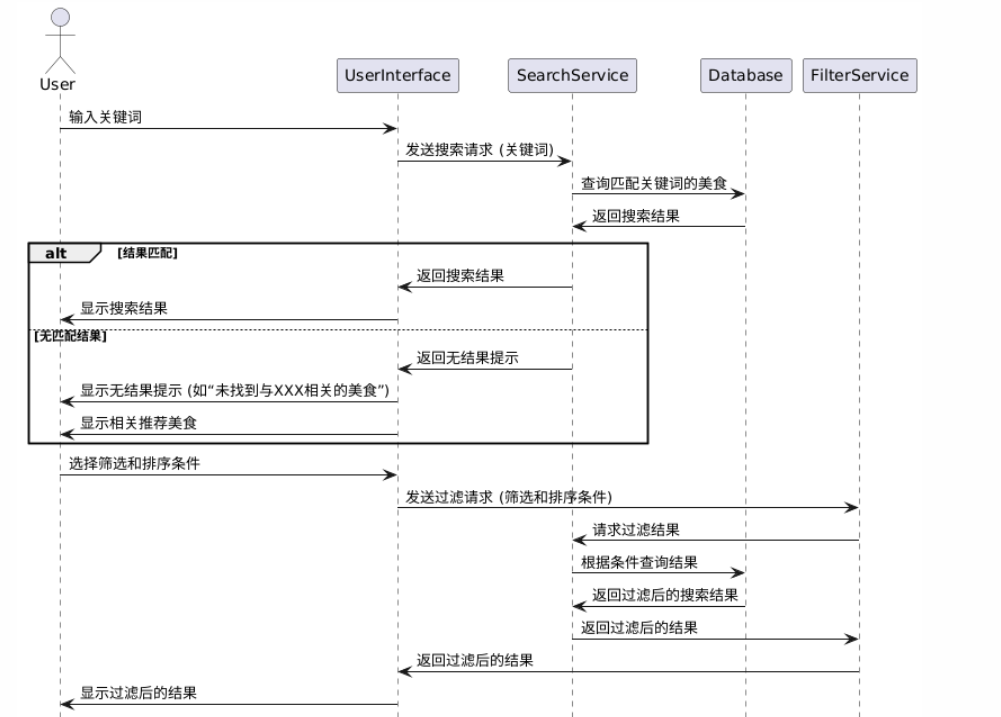
精准搜索：用户输入完整的关键词时，系统返回与该关键词完全匹配 的结果。

多字段搜索：用户可以通过关键词同时检索多个字段，如美食名称、 菜系、口味等。

3.搜索过滤：搜索结果页面顶部应有筛选和排序选项，用户可以根据菜系、口味、评分等条件对结果进行进一步操作。

4.无匹配结果：如果没有匹配的搜索结果，系统应显示友好的提示信息（如“没有找到与‘XXX’相关的美食”），并提供相关推荐美食列表以供参考。

**时序图：**



### 分类搜索

**功能简介：**分类搜索模块允许用户根据预设的分类条件（如菜系、口味、食材、地域等）进行快速、精确的美食搜索。该模块旨在提升用户体验，帮助用户通过直观的分类导航找到感兴趣的美食。

**模块说明：**

1.系统支持多种分类条件，用户可以根据不同的维度组合选择来缩小搜索范围。主要分类条件包括：

菜系: 按不同的菜系（如川菜、粤菜、鲁菜等）进行搜索。

口味: 按美食的主要口味（如辣、甜、酸、咸等）进行搜索。

地域: 按美食的地域来源（如中国不同省份、国家或地区）进行搜索。

食材: 按主要食材（如牛肉、鱼、蔬菜等）进行筛选。

烹饪方式: 按美食的制作方法（如炒、煮、烤、蒸等）进行搜索。

2.多重筛选: 用户可以选择多个分类维度组合进行搜索。例如，用户可以同时选择“川菜”+“辣”+“蒸”来寻找符合该条件的美食。

3.用户每选择或取消一个筛选条件时，系统会立即更新搜索结果，而无需用户手动点击“搜索”。

**时序图：**



# 数据库设计

## E-R图

通过对用户，食物，评分，评论等实体进行分析得到实体的属性，同时分析实体与实体之间的关系，得到如图4-1所示的实体关系图

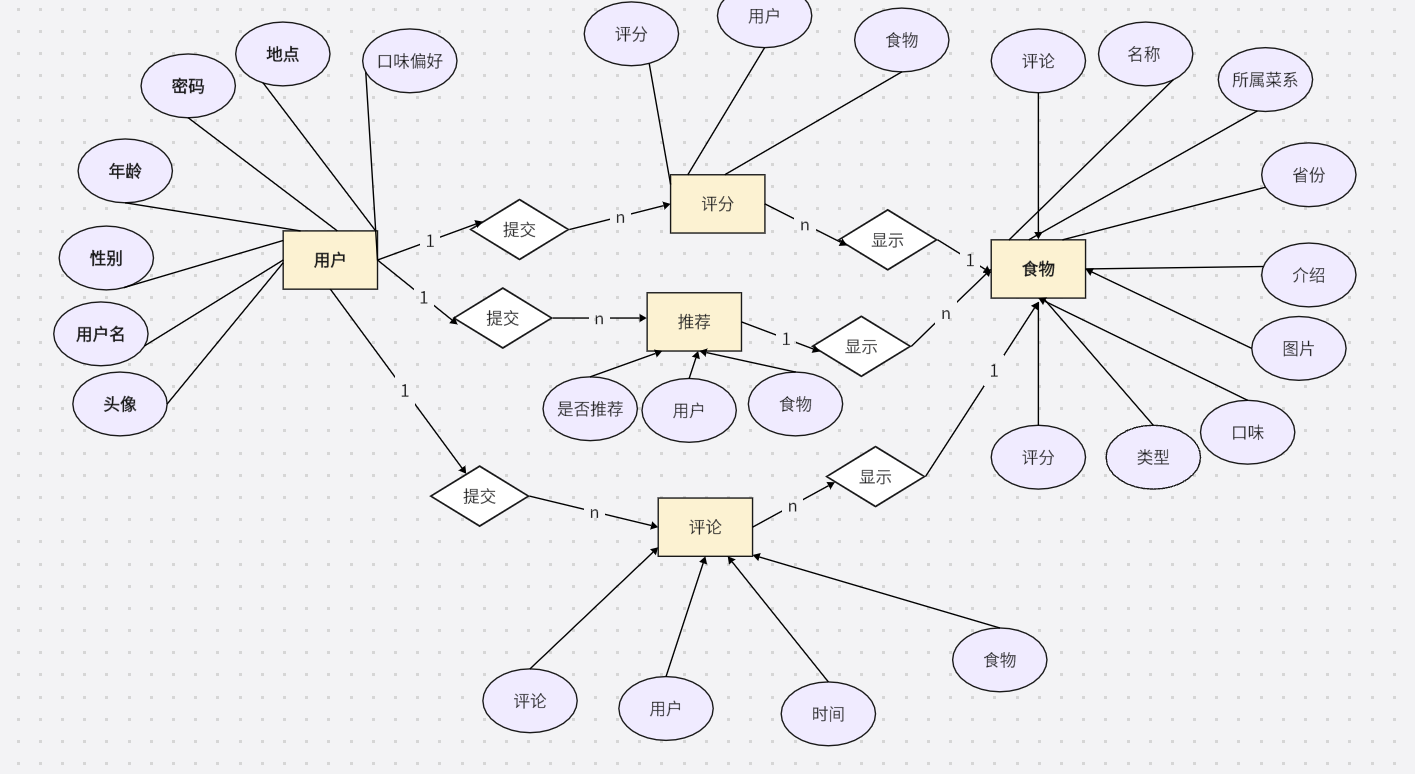


图4-1 实体关系图

## 类图

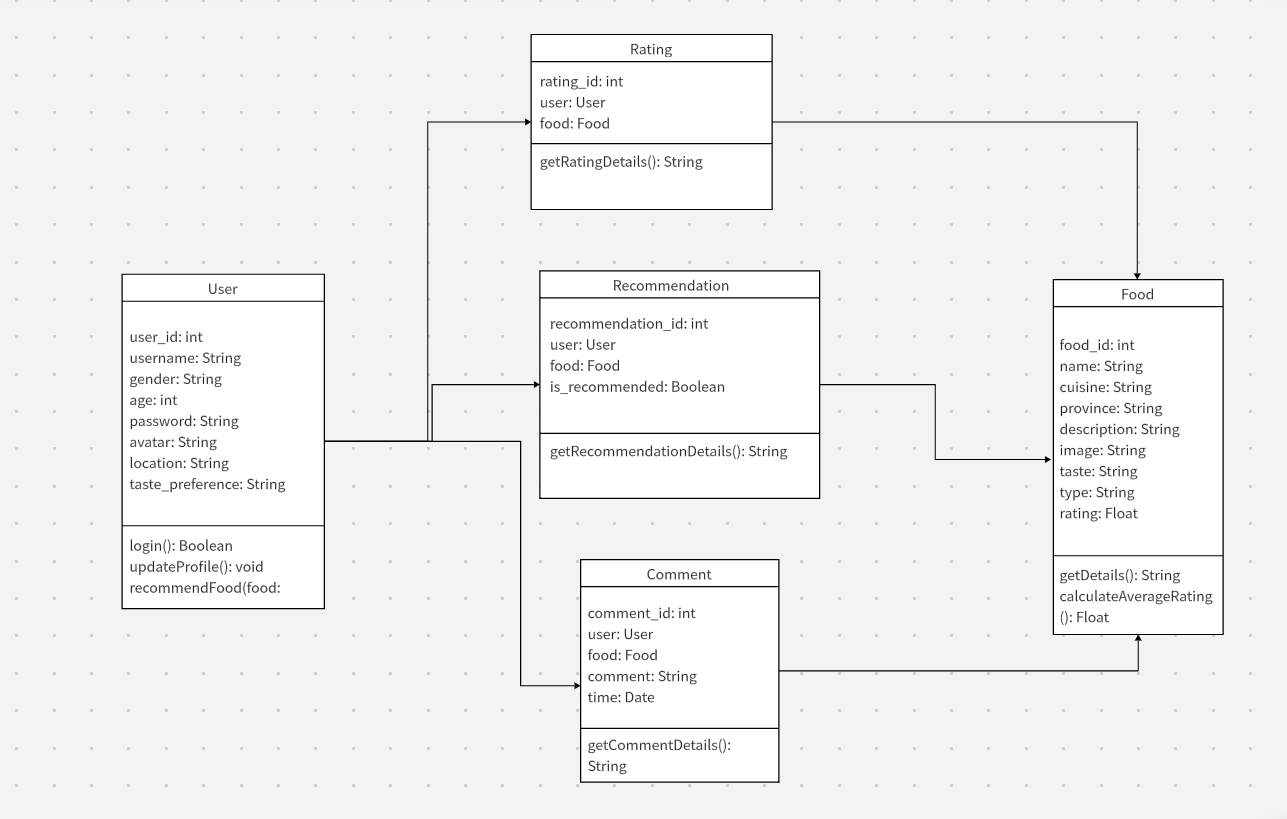


图4-2 类图

## 数据库表

### 食物表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 英文字段 | 中文含义 | 数据类型 | 字段长度 | 备注 |
| food\_id | 食物ID | INT |  | 主键 |
| name | 食物名称 | VARCHAR | 255 | NOT NULL |
| cuisine | 所属菜系 | VARCHAR | 255 |  |
| province | 省份 | VARCHAR | 255 |  |
| description | 介绍 | TEXT |  |  |
| image | 图片 | VARCHAR | 255 |  |
| taste | 口味 | VARCHAR | 255 |  |
| type | 类型 | VARCHAR | 255 |  |
| rating | 评分 | FLOAT | DEFAULT 0 |  |
| comment | 评价 | TEXT |  |  |

### 用户表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 英文字段 | 中文含义 | 数据类型 | 字段长度 | 备注 |
| user\_id | 用户ID | INT |  | 主键 |
| username | 用户名称 | VARCHAR | 100 | NOT NULL |
| gender | 性别 | VARCHAR | 255 |  |
| age | 年龄 | INT | 255 |  |
| password | 密码 | VARCHAR |  | NOT NULL |
| avatar | 头像 | VARCHAR | 255 | default\_avatar.jpg |
| location | 位置 | VARCHAR | 255 |  |
| taste\_preference | 口味偏好 | VARCHAR | 100 |  |

### 评分表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 英文字段 | 中文含义 | 数据类型 | 字段长度 | 备注 |
| rating\_id | 评分ID | INT |  | 主键 |
| user\_id | 用户ID | INT |  | NOT NULL |
| food\_id | 食物ID | INT |  | NOT NULL |
| rating | 评分 | FLOAT | DEFAULT 0 | NOT NULL |

### 评价表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 英文字段 | 中文含义 | 数据类型 | 字段长度 | 备注 |
| comment\_id | 评价ID | INT |  | 主键 |
| user\_id | 用户ID | INT |  | NOT NULL |
| food\_id | 食物ID | INT |  | NOT NULL |
| comment | 评价 | FLOAT | DEFAULT 0 | NOT NULL |
| time | 时间 | TIMESTAMP |  |  |

### 推荐表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 英文字段 | 中文含义 | 数据类型 | 字段长度 | 备注 |
| Recommendation\_id | 推荐ID | INT |  | 主键 |
| user\_id | 用户ID | INT |  | NOT NULL |
| food\_id | 食物ID | INT |  | NOT NULL |
| is\_recommended | 是否推荐 | BOOLEAN | DEFAULT 0 | NOT NULL |

# 接口设计

## 用户模块接口

### 用户注册

HTTP方法： POST

URL路径： <http://localhost:5000/users/register>

请求参数：

{

"username": "string",

"password": "string"

}

成功响应示例：

{

"message": "注册成功",

"userId": 1

}

失败响应实例：

{

"error": "用户名已存在"

}

{

"error": "密码长度必须大于6个字符"

}

### 用户登录

HTTP方法： POST

URL路径： http://localhost:5000/users/login

请求参数：

{

"username": "string",

"password": "string"

}

成功响应示例：

{

"message": "登录成功"

}

失败响应示例：

{

"error": "用户名或密码错误"

}

### 用户信息管理

HTTP方法： GET

URL路径： http://localhost:5000/users/{userId}

成功响应示例：

{

"userId": 1,

"username": "user1",

"preferences": "辛辣"

}

失败响应示例：

{

"error": "用户不存在"

}

## 美食模块接口

### 获取美食分类

HTTP方法： GET

URL路径： http://localhost:5000/foods/categories

成功响应示例：

[

{

"categoryId": 1,

"name": "川菜"

},

{

"categoryId": 2,

"name": "粤菜"

}

]

### 根据省份、口味、菜系等获取美食

HTTP方法： GET

URL路径： http://localhost:5000/foods

请求参数：

{

"province": "string",

"taste": "string",

"cuisine": "string"

}

成功响应示例：

[

{

"foodId": 1,

"name": "麻婆豆腐",

"image": "http://example.com/image.jpg",

"rating": 4.5

}

]

失败响应示例：

{

"error": "没有找到符合条件的美食"

}

### 美食搜索

HTTP方法： GET

URL路径： http://localhost:5000/foods/search

请求参数：

{

"keyword": "string"

}

成功响应示例：

[

{

"foodId": 1,

"name": "宫保鸡丁",

"description": "经典川菜，味道鲜香",

"image": "http://example.com/image.jpg"

}

]

### 获取美食详细信息

HTTP方法： GET

URL:路径： http://localhost:5000/foods/{foodId}

成功响应示例：

{

"foodId": 1,

"name": "麻婆豆腐",

"description": "麻辣鲜香的豆腐菜肴",

"history": "起源于四川",

"taste": "麻辣",

"cuisine": "川菜",

"image": "http://example.com/image.jpg",

"rating": 4.5

}

## 评论模块接口

### 添加评论

HTTP方法:： POST

URL:路径： http://localhost:5000/comments

请求参数：

{

"userId": 1,

"foodId": 1,

"content": "很好吃！",

"rating": 5

}

成功响应示例：

{

"message": "评论成功",

"commentId": 1

}

### 获取美食评论

HTTP方法： GET

URL:路径： http://localhost:5000/comments/{foodId}

成功响应示例：

[

{

"commentId": 1,

"userId": 1,

"content": "很好吃！",

"rating": 5,

"timestamp": "2024-10-16T10:00:00Z"

}

]

## 个性化推荐模块接口

### 获取个性化推荐

HTTP方法： GET

URL路径： http://localhost:5000/recommendations/{userId}

成功响应示例:

[

{

"foodId": 1,

"name": "麻婆豆腐",

"image": "http://example.com/image.jpg",

"rating": 4.5

}

]

## 排行榜模块接口

### 获取美食排行榜

HTTP方法:： GET

URL路径： http://localhost:5000/leaderboard

请求参数:

{

"criteria": "rating" // or "popularity"

}

成功响应示例：

[

{

"foodId": 1,

"name": "麻婆豆腐",

"rating": 4.5,

"recommendCount": 100

},

{

"foodId": 2,

"name": "宫保鸡丁",

"rating": 4.2,

"recommendCount": 80

}

]

## 管理模块接口

### 添加美食

HTTP方法： POST

URL路径： http://localhost:5000/admin/foods

请求参数：

{

"name": "麻婆豆腐",

"description": "麻辣鲜香的豆腐菜肴",

"cuisine": "川菜",

"image": "http://example.com/image.jpg",

"rating": 4.5

}

成功响应示例：

{

"message": "美食添加成功",

"foodId": 1

}

失败响应示例：

{

"error": "美食名称不能为空"

}

### 更新美食信息

HTTP方法： POST

URL: http://localhost:5000/admin/foods/{foodId}

请求参数：

{

"name": "麻婆豆腐",

"description": "经典麻辣豆腐菜",

"cuisine": "川菜",

"image": "http://example.com/new\_image.jpg",

"rating": 4.6

}

成功响应示例：

{

"message": "美食信息更新成功"

}

### 删除美食

HTTP方法： POST

URL路径：http://localhost:5000/admin/foods/{foodId}

成功响应示例：

{

"message": "美食删除成功"

}