## 系统架构设计

### 1. 系统架构概述

外卖管理系统采用 **B/S架构**，即浏览器/服务器架构，通过浏览器与服务器进行数据交互。整体架构分为前端、后端和数据库三大部分，前端与后端通过 **RESTful API** 进行通信，前后端分离，确保系统的扩展性与可维护性。

#### 架构层次

1. **前端层（客户端）**：
   * 使用 **Vue.js** 框架搭建单页应用（SPA），确保良好的用户体验和响应速度。
   * 通过 **Axios** 实现与后端的 API 调用。
2. **后端层（服务端）**：
   * 使用 **Laravel（PHP）** 或 **Django（Python）** 作为后端框架，处理业务逻辑、用户认证、订单管理等。
   * 提供 **RESTful API** 服务，供前端进行调用。
3. **数据库层**：
   * 使用 **MySQL** 或 **PostgreSQL** 存储系统中的数据，如用户信息、订单信息、菜品信息等。
   * 配置数据库备份、数据恢复、以及必要的索引优化。
4. **第三方服务**：
   * **支付网关**：如支付宝、微信支付。
   * **短信/邮件通知**：用于订单状态更新等。

## 详细设计 - 模块划分与功能说明

### 1. 用户管理模块

#### 功能：

* 用户注册、登录、注销。
* 用户资料管理：如昵称、联系方式等。
* 账户安全：修改密码、找回密码。

#### 设计：

* 前端：用户通过输入邮箱/手机号和密码进行登录，支持社交平台（如微信、QQ）的第三方登录。
* 后端：使用 JWT 进行用户认证，存储加密后的密码，支持 Session 方式的身份管理。
* 数据库：User 表，包含用户基本信息（如 id、username、password、email、role 等）。

### 2. 餐厅与菜品管理模块

#### 功能：

* 餐厅信息管理：餐厅名称、地址、营业时间、联系方式。
* 菜品管理：添加菜品、修改菜品信息、删除菜品、设置价格和库存。

#### 设计：

* 前端：餐厅界面显示菜品的列表，支持搜索、排序（按销量、价格）、筛选。
* 后端：商家通过后台管理系统进行菜品的添加、修改、删除。
* 数据库：Restaurant 表（餐厅信息）、MenuItem 表（菜品信息）。

### 3. 订单管理模块

#### 功能：

* 用户浏览餐厅菜单并选择菜品，生成订单。
* 用户选择配送地址并确认下单。
* 支付订单（支持支付宝、微信等支付方式）。
* 订单状态管理：未支付、已支付、配送中、已完成、已取消。

#### 设计：

* 前端：用户界面显示购物车、订单详情，支持支付接口。
* 后端：订单生成后，系统将订单状态存入数据库并等待商家接单。
* 数据库：Order 表（订单信息，包括用户id、餐厅id、菜品id、支付状态等）。

### 4. 支付管理模块

#### 功能：

* 用户选择支付方式（如支付宝、微信支付）。
* 支付完成后更新订单状态。
* 订单支付失败时，支持退款流程。

#### 设计：

* 前端：集成支付宝或微信支付SDK，用户支付界面。
* 后端：与支付网关进行对接，支付成功后更新订单状态。
* 数据库：Payment 表，记录支付信息（如订单id、支付方式、金额、支付状态等）。

### 5. 配送管理模块

#### 功能：

* 配送员接收配送任务。
* 配送员查看订单信息、获取配送路线。
* 配送状态更新：配送中、已配送。

#### 设计：

* 前端：配送员移动端应用，查看待配送订单，更新配送状态。
* 后端：系统将已支付的订单分配给配送员，并监控配送状态。
* 数据库：Delivery 表，记录配送任务（如订单id、配送员id、配送状态等）。

### 6. 评价管理模块

#### 功能：

* 用户对餐厅或订单进行评价（评分、文字评论）。
* 展示评价信息。

#### 设计：

* 前端：用户订单完成后，可以对订单进行评分和评论。
* 后端：评价信息存储在数据库，并提供评价汇总功能。
* 数据库：Review 表，记录评价信息（如用户id、订单id、评分、评论文本等）。

### 7. 后台管理模块

#### 功能：

* 管理员管理系统中的所有用户（用户、商家、配送员）信息。
* 系统监控：查看订单情况、商家表现、配送进度等。

#### 设计：

* 前端：后台管理界面，管理员可以对各类数据进行增删改查。
* 后端：提供权限控制、数据统计、报表等功能。
* 数据库：Admin 表，记录管理员信息；相关的操作记录存入日志表。

## 系统模块间的交互

1. **用户流程**：
   * 用户注册 → 登录 → 浏览餐厅与菜品 → 添加至购物车 → 下单 → 支付 → 配送员配送 → 订单完成 → 用户评价。
2. **商家流程**：
   * 商家登录 → 管理菜品信息 → 查看订单 → 确认接单 → 配送员配送。
3. **配送员流程**：
   * 配送员登录 → 查看待配送订单 → 接单 → 完成配送。
4. **管理员流程**：
   * 管理员登录 → 管理用户、商家、配送员 → 查看系统状态 → 生成报表。

## 数据库设计

### 表结构

1. **User 表**：存储用户的基本信息。
   * user\_id, username, password, email, phone, role (用户类型: 用户/商家/配送员/管理员)
2. **Restaurant 表**：存储餐厅信息。
   * restaurant\_id, name, location, contact, rating
3. **MenuItem 表**：存储菜品信息。
   * item*id, restaurant*id, name, price, stock, description
4. **Order 表**：存储订单信息。
   * order*id, user*id, restaurant*id, total*price, order*status, order*time, delivery\_time
5. **Payment 表**：存储支付信息。
   * payment*id, order*id, payment*method, payment*status, amount
6. **Delivery 表**：存储配送信息。
   * delivery*id, order*id, delivery*status, driver*id, delivery\_time
7. **Review 表**：存储用户评价。
   * review*id, user*id, order*id, rating, review*text, review\_time

## 接口设计（API）

* **POST /api/register**：注册新用户。
* **POST /api/login**：用户登录，获取 JWT token。
* **GET /api/restaurants**：获取餐厅列表。
* **GET /api/menu/{restaurant\_id}**：获取指定餐厅的菜品。
* **POST /api/order**：提交订单。
* **POST /api/payment**：处理支付。
* **GET /api/delivery/{order\_id}**：查看配送状态。