基于Django + Vue + MySQL 开发的电商平台

21307140069 田沐钊

1 功能需求

1.1 数据管理功能

- 用户端和商家端均可查看商品、购物车、订单、点赞收藏等表的数据，并根据不同的权限进行增删改查。
- 商家端支持通过Excel批量导入商品进货数据，包括进货单价和数量。
- 商家端支持通过Excel导出商品销售数据，并按日期、地点、商品类型筛选导出。

1.2 数据展示功能

- 用户端支持按销量和售价排序商品，并支持商品与商家间的跳转。
- 商家端支持分页展示上架商品，并自定义每页显示的商品数量。
- 用户端和商家端支持上传和展示用户头像和商品图片。
- 用户端支持商品的模糊搜索。
- 用户端通过标签、颜色、销量等比较商品并推荐相似商品。
- 商家端支持按日期、地点、商品类型比较销售数据。

1.3 业务功能

- 用户端支持商品的模糊搜索和通过标签、颜色、销量等比较商品并推荐相似商品。
- 商家端支持按日期、地点、商品类型比较销售数据。
- 商家端和用户端支持订单管理，涵盖从发货到收货的各个状态。
- 管理员审核商品上架和修改，审核通过后商品展示在用户端。

1.4 统计分析

- 商家端支持通过PDF导出商品销售图表，使用Excel导出商品销售数据，并按日期、地点、商品类型等筛选导出。
- 商家端支持图表展示和Excel导出商品销售数据，按不同数据粒度筛选显示或导出，包括省-市-区逐级销量分析、所有商品-相同类型商品-单个商品统计分析。
- 用户端点击商品可以查看销量、点赞情况，并关联到商家的收藏情况；从商家页面关联到其下商品的销量、点赞情况。

1.5 安全防护

- 系统支持用户、商家、管理员三类用户。
- 用户和商家可以注册和登录。
- 管理员审核用户权限和商品上架、修改，商品数据在审核通过后展示在用户端。
- 管理员页面可以查看用户和商家的访问日志、下单和编辑商品的操作日志。
- 管理员可查询和审核商品上架与修改情况，查看商品审核日志，包括申请上架、申请修改、审核通过和回退申请等状态。

2 各端功能描述与数据流图

2.1 商家端

1. 注册、登录、退出登录；
2. 查看并修改订单状态；
3. 提交商品上架申请，审核通过后成功上传商品；
4. 修改商品信息和库存，通过Excel导入进货数据；
5. 查看已上架商品；
6. 查看销售数据，支持不同粒度的分析，并导出为PDF或Excel；
7. 查看和修改个人信息及密码；
8. 进行安全性检查。

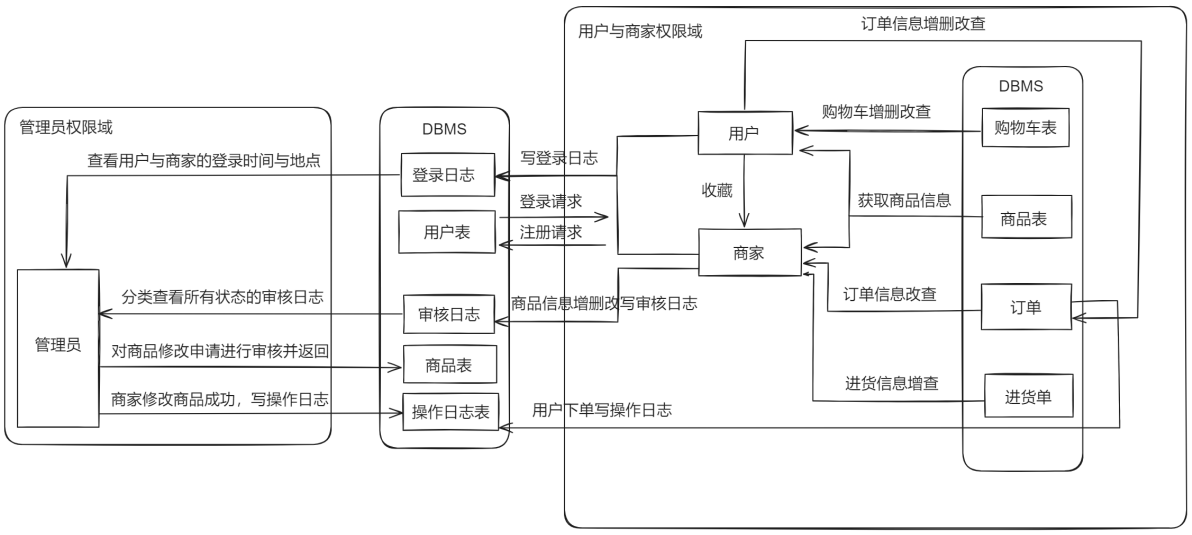
2.2 用户端

1. 注册、登录、退出登录；
2. 浏览所有商品，支持关键词搜索和按标签分类，查看商品详细信息；
3. 点赞商品、收藏店铺；
4. 将商品加入购物车并购买；
5. 查看并修改订单状态；
6. 查看和修改个人信息及密码；
7. 进行安全性检查。

2.3 管理员端

1. 审核商家提交的商品上架和修改申请，对合法商品上架请求进行审核通过，对非法商品或恶意低价申请予以回退；
2. 查看用户登录日志，记录用户每次登录的时间、地点和用户类型，并写入登录日志。

2.4 数据流图



3 数据库描述

3.1 用户表 (UserInfo)

名称	数据类型	大小	是否必填	是否主键	说明
user_id	Int	10	是	是	用户id
user_password	Varchar	64	是	否	用户密码
user_type	Varchar	10	是	否	用户类型
user_name	Varchar	64	是	否	用户名字
user_mobile	Char	10	否	否	用户电话
user_province	Char	4	否	否	用户所在省
user_city	Char	6	否	否	用户所在市
user_area	Char	6	否	否	用户所在区
user_address	Varchar	64	否	否	用户详细地址
user_avatar	Image		否	否	用户头像
user_createtime	Date		是	否	用户注册时间

3.2 商品表 (Product)

名称	数据类型	大小	是否必填	是否主键/外键	说明
product_id	Int	10	是	主键	商品id
product_name	Varchar	100	是	否	商品名字
product_cost	Float	2位小数	是	否	商品价格
product_color	Varchar	20	是	否	商品颜色
product_business	Int	10	否	外键	商家id
product_brand	Varchar	100	是	否	商品类别
product_imageDetail	Image		否	否	商品细节图片
product_stock	Int	10	是	否	商品库存
product_sales	Int	10	是	否	商品销量
product_image	Image		否	否	商品图片
products_status	Varchar	10	是	否	商品状态

3.3 用户收藏商家关系表 (UserFavoriteBusiness)

名称	数据类型	大小	是否必填	是否主键/外键	说明
user_id	Int	10	是	主键, 外键	用户id
business_id	Int	10	是	主键, 外键	商家id
star_id	Int	10	是	主键	收藏关系id

3.4 用户点赞商品关系表 (UserLikeProduct)

名称	数据类型	大小	是否必填	是否主键/外键	说明
user_id	Int	10	是	主键, 外键	用户id
product_id	Int	10	是	主键, 外键	商品id
like_id	Int	10	是	主键	点赞关系id

3.5 购物车表 (Cart)

名称	数据类型	大小	是否必填	是否主键/外键	说明
cart_id	Int	10	是	主键	购物车id
user_id	Int	10	是	外键	用户id
product_id	Int	10	是	外键	商品id
quantity	Int	10	是	否	商品数量

3.6 订单基本信息表 (OrderInfo)

名称	数据类型	大小	是否必填	是否主键/外键	说明
order_id	Int	10	是	主键	订单id
order_time	Date	无	是	否	下单时间
order_customer_id	Int	10	是	外键	下单用户id
order_business_id	Int	10	是	外键	下单商家id
order_status	Varchar	64	是	否	订单所处状态
order_customer_name	Varchar	64	是	否	用户名字
order_customer_mobile	Char	11	是	否	用户电话
order_customer_province	Char	6	是	否	用户所在省
order_customer_city	Char	6	是	否	用户所在市
order_customer_area	Char	6	是	否	用户所在区
order_customer_address	Varchar	64	是	否	用户详细地址
order_business_name	Varchar	64	是	否	商家名字
order_business_mobile	Char	11	是	否	商家电话
order_business_province	Char	6	是	否	商家所在省
order_business_city	Char	6	是	否	商家所在市
order_business_area	Char	6	是	否	商家所在区
order_business_address	Varchar	64	是	否	商家详细地址

3.7 订单商品关系表 (OrderProduct)

名称	数据类型	大小	是否必填	是否主键	功能
relation id	int	10	是	外键	订单包含商品关系id
relation order id	Int	10	是	外键	订单id
relation product id	Int	10	是	否	订单包含的某个商品id
relation product num	Int	10	是	否	商品在此订单的数量
relation product cost	Float	10	是	否	商品在此订单的单价（2位小数）

3.8 商品进货关系表 (ProductStock)

名称	数据类型	大小	是否必填	是否主键	功能
stock id	Int	10	是	主键	商品进货关系id
stock createtime	Date	无	是	否	进货时间
stock product id	int	10	是	外键	进货商品id
stock product num	Int	10	是	否	商品进货数量
stock product cost	Float	10	是	否	商品进货单价（2位小数）

3.9 商品审核日志表 (ProductAuditLog)

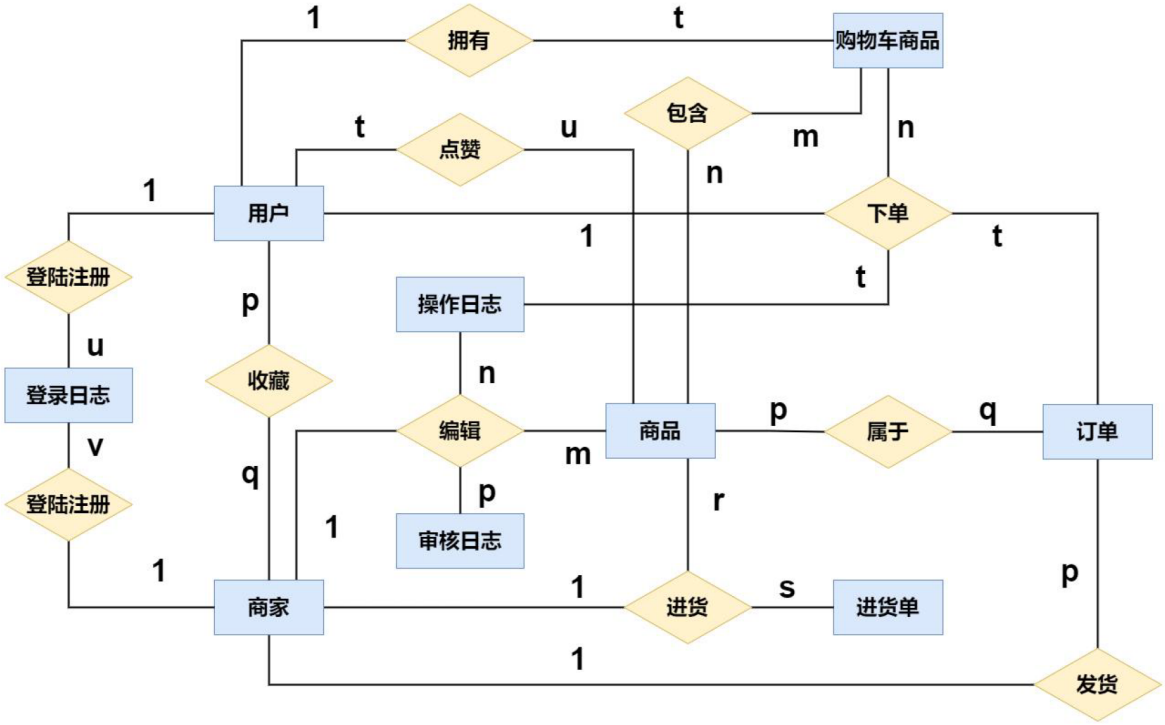
名称	数据类型	大小	是否必填	是否主键	功能
check id	Int	10	是	主键	审核日志id
check product id	Int	10	是	外键	审核商品id
check product name	Varchar	100	是	否	审核商品名字
check product cost	Float	2	是	否	审核商品价格（2位小数）
check product color	Varchar	20	是	否	审核商品颜色
check product brand	Varchar	100	是	否	审核商品品牌
check product imageDetail	Float	10	否	否	审核商品细节图片

名称	数据类型	大小	是否必填	是否主键	功能
check product image	Image	10	否	否	审核商品图片
check products status	Varchar	10	是	否	商品审核类型
check log status	Varchar	10	是	否	商品审核状态
check createtime	Date	10	是	否	提交审核时间

3.10 用户登录日志表 (UserLoginLog)

名称	数据类型	大小	是否必填	是否主键	功能
log_id	Int	10	是	主键	用户登录日志id
log user id	Int	10	是	外键	用户登录id
log user name	Varchar	64	是	否	用户登录名字
log_user _type	Varchar	10	是	否	用户登录身份
log time	Char	11	否	否	用户登录时间
log user province	char	6	否	否	用户登录所在省
log user city	Char	6	否	否	用户登录所在市
log user_area	char	6	否	否	用户登录所在区

3.11 ER图描述



3.12 关系模式范式验证

根据语义来分析函数依赖如下：

- 用户id → 用户名，密码，头像，电话，用户类型，注册时间，地址
- 商品id → 商品名，商品价格，商品品牌，商品颜色，商家id，商品销量，商品库存，商品图片，商品状态
- 点赞id → 点赞商品id，点赞用户id
- 收藏id → 收藏商家id，收藏用户id
- 购物车商品项id → 用户id，商品id，商品数量
- 订单id → 下单时间，用户id，商家id，收货人名，收货电话，收货地址，发货地址，发货电话，发货人名，订单状态
- 订单商品id → 订单id，商品id，订货数量，订货单价
- 进货单单id → 进货时间，商品id，进货数量，进货单价
- 登录日志id → 登录用户名，登录身份，登陆时间，登陆地点
- 操作日志id → 操作用户名，操作身份，操作时间，操作地点，操作类型
- 审核日志id → 审核商品id，审核商品名，审核商品价格，审核商品品牌，审核商品颜色，审核商品图片，申请类型，审核状态

通过以上分析，确认该数据库关系模式满足第三范式（3NF）要求，不存在非主属性对主码的部分依赖和传递依赖。

4 项目实现环境

本次项目采用的是前后端分离的web开发。

1. **前端**

   前端采用了 Vue 框架，这是目前最流行的 JavaScript 框架之一。我们同时使用了 Vuetify、ElementUI、FontAwesome 等前端组件库。Vuetify 是建立在 Vue.js 之上的完备界面框架，提供了数百个精心设计的组件，符合 Material 设计规范。在项目中，通过以下命令在 `src/plugins` 文件夹下生成 `vuetify.js` 文件，之后即可使用 Vuetify 组件进行开发。开发完成后，使用以下命令将 Vue 文件打包成 HTML+CSS+JS 形式的前端工程项目，并保存在 `template/src/dist` 文件夹下。

2. **后端**

   后端采用 MySQL 数据库，使用 Python 语言和 Django 框架进行开发。Django 是一个开放源代码的 Web 应用框架，基于 MVC 模型（模型 + 视图 + 控制器）。MVC 模式简化了对程序的修改和扩展，支持模块的重复利用。通过 Django、Python 和 MySQL 的结合，能够极大地简化数据读取和管理操作，实现网站的快速开发、设计和部署。

3. **前后端交互**

   前端使用 Axios 发送 GET 请求获取用户请求，后端接收请求后对数据库进行查询，并将查询结果以 JSON 形式返回给前端进行显示。

   对于用户提交的文件和数据，前端使用 Form 表单形式传输给后端，通过 Axios 发送 POST 请求。后端接收数据后对数据库进行增、删、改、查操作，并将修改结果以 JSON 形式返回给前端显示。

5 触发器、存储过程等部分的代码说明

5.1 触发器

mysql 提供触发器与存储过程机制，能够从后端提供一定的数据处理能力，简化前端的数据处理量，提高页面响应速度。以下以管理员对用户操作的日志查看页面为样例，介绍本项目的触发器和存储过程使用。

mysql 编写触发器，用于用户进行下单和编辑商品时，系统记录用户操作日志：

```
delimiter $$
Create Trigger trigertest3
after insert on myapp_orderinfo
for each row begin
    insert into myapp_useractionlog
    (action_name, action_type, action_province, action_city, action_area, action,
action_time, action_id_id)
    values (new.customer_name, '用户', new.customer_proyince, new.customer_city,
new.customer_area, '下单', now(), new.customer_id_id);
end
$$
delimiter ;
```

5.2 存储过程

前后端交互，调用存储过程查询前端所需数据：

```
def get_user_action_num(request):
    response = {}
```

```
    try:
        from django.db import connection

        with connection.cursor() as cur:
            cur.callproc("testProc3", [0])
            resu = cur.fetchall()
            print(resu[0][0])

        response["num"] = resu[0][0]
        response["msg"] = "success"
        response["error_num"] = 0
    except Exception as e:
        response["msg"] = str(e)
        response["error_num"] = 1
    return JsonResponse(response)
```

mysql 编写存储过程，用于返回用户操作日志的相关数据：

```
delimiter $$
CREATE PROCEDURE testProc3(ouT m int)
BEGIN
select count(*) from myapp_useractionlog;
END $$
delimiter ;
```

5.3 索引

在数据库中可以选择合适的存储管理方式如索引等进行物理结构优化。尽管无法直接对 MySQL 数据库内部的物理结构进行优化，但在本次项目中，我们从索引的物理角度进行了以下考虑：

1. **数据库管理系统的选择**

   根据课程理论，索引在物理结构中是快速搜索的关键。我们选择了 MySQL 数据库管理系统，它配备了优化的 SQL 查询算法，并使用 B+ 树作为索引结构，有效提高了数据库的查询速度。

2. **维护数据索引**

   基于索引的物理结构，我们对数据库中的每行数据添加了索引标识，其中主键索引（Primary Key）是最常见的一种。在创建表格并向数据库添加数据时，我们同时创建主键索引。通过维护这些数据索引，结合 MySQL 数据库管理系统的物理结构，显著优化了查询效率。

3. **最大化索引的优化使用**

   数据库管理系统在处理包含 NULL 值的 WHERE 条件时，可能会导致物理结构索引失效，从而进行全表扫描。为了优化性能，我们应该尽可能避免 NULL 值的出现，并在每次数据操作时对数据字段进行必要的 NULL 值判断和数据限制。这样可以确保物理结构索引能够最大化地发挥作用，提升查询效率。

6 主要功能模块实现说明

1. 用户、商家注册

- **前端文件：** `login.vue`
- **涉及的数据表：** `UserInfo`
- **流程解析：**
  - 后端从前端以 JSON 形式获取用户注册信息。首先检查账号和密码是否为空，若为空则前端提示“账号或密码不能为空”。
  - 其次查询用户名是否重复，若重复则前端提示“用户名重复，请再次尝试”。
  - 接着检查密码和确认密码是否一致，若不一致则前端提示“密码不一致，请重新输入”。
  - 若通过以上检查，则将注册信息存入用户信息表，并提示“注册成功，确定进入网站首页”。
- **代码实现：**

```
def add_userInfo(request):
    response = {}
    try:
        userInfo = UserInfo(
            user_name=request.GET.get("user_name"),
            user_password=request.GET.get("user_password"),
            user_type=request.GET.get("user_type"),
        )
        userInfo.save()
        response["msg"] = "success"
        response["userId"] = userInfo.id
        response["error_num"] = 0
    except Exception as e:
        response["msg"] = str(e)
        response["error_num"] = 1
    return JsonResponse(response)
```

2. 用户、商家登录

- **前端文件：** `login.vue`
- **涉及的数据表：** `UserInfo`
- **流程解析：**
  - 后端从前端以 JSON 形式获取用户登录信息。首先检查账号和密码是否为空，若为空则前端提示“账号或密码不能为空”。
  - 其次查询用户名是否存在且密码是否匹配，若不匹配则前端提示“账号或密码错误”。
  - 若通过以上检查，则查询用户类型（商家或用户），并跳转到对应页面，并提示“登录成功，进入网站首页”。
  - 用户登录后，在首页显示用户名。
- **代码实现：**

```
def search_userInfo(request):
    response = {}
```

```
    try:
        userInfo = UserInfo.objects.get(
            user_name=request.GET.get("user_name"),
            user_password=request.GET.get("user_password"),
        )
        response["userId"] = userInfo.id
        response["user_type"] = userInfo.user_type
        response["msg"] = "succes111s"
        response["error_num"] = 0
    except Exception as e:
        response["msg"] = str(e)
        response["error_num"] = 1

    return JsonResponse(response)

def search_usertype(request):
    response = {}
    try:
        userInfo = UserInfo.objects.get(id=request.GET.get("user_id"))

        response["user_type"] = userInfo.user_type
        response["msg"] = "succes111s"
        response["error_num"] = 0
    except Exception as e:
        response["msg"] = str(e)
        response["error_num"] = 1

    return JsonResponse(response)

def userId2userName(request):
    response = {}
    try:
        userInfo = UserInfo.objects.get(id=request.GET.get("user_id"))
        response["user_name"] = userInfo.user_name
        response["msg"] = "succes111s"
        response["error_num"] = 0
    except Exception as e:
        response["msg"] = str(e)
        response["error_num"] = 1

    return JsonResponse(response)
```

3. 用户、商家展示个人信息

- **前端文件：** `business_information.vue`
- **涉及的数据表：** `UserInfo`
- **流程解析：**
  - 后端将数据库中的用户信息传入前端，前端进行展示。
- **代码实现：**

```
def fetch_userInfo(request):
    response = {}
    try:
        userInfo = UserInfo.objects.get(id=request.GET.get("user_id"))
        userInfoss = UserInfo.objects.filter(id=request.GET.get("user_id"))
        response["user_name"] = userInfo.user_name if userInfo.user_name else ""
        response["user_mobile"] = userInfo.user_mobile if userInfo.user_mobile else
""
        response["user_avatar"] = json.loads(serializers.serialize("json",
userInfoss))[
            0
        ]
        response["user_address"] = (
            userInfo.user_address if userInfo.user_address else ""
        )
        response["user_createtime"] = str(userInfo.user_createtime)
        response["user_province"] = (
            userInfo.user_province if userInfo.user_province else ""
        )
        response["user_city"] = userInfo.user_city if userInfo.user_city else ""
        response["user_area"] = userInfo.user_area if userInfo.user_area else ""
        response["msg"] = "succes111s"
        response["error_num"] = 0
    except Exception as e:
        response["msg"] = str(e)
        response["error_num"] = 1

    return JsonResponse(response)
```

4. 用户、商家修改个人信息

- **前端文件：** `business_information.vue`
- **涉及的数据表：** `UserInfo`
- **流程解析：**
  - 后端从前端以 JSON 形式获取用户修改的信息，检查用户名是否重复。
  - 若用户名不重复，则更新用户信息，并刷新前端个人信息展示页面。
- **代码实现：**

```
def edit_userInfo(request):
    response = {}
    global userInfo
    print("111")
    iid = int(str(request.FILES.get("user_id").read())[2:-1])

    try:

        print(iid)
        userInfo = UserInfo.objects.get(id=iid)
        print(iid)
```

```
        save_userInfo = json.loads(request.FILES.get("forms").read())
        same_name = UserInfo.objects.filter(user_name=save_userInfo["user_name"])
        if same_name.count() == 2:
            response["msg"] = "用户名重复"
            response["error_num"] = 1
            return JsonResponse(response)
        elif same_name.count() == 1:
            if str(same_name[0].id) != str(iid):
                print("name1:", str(same_name[0].id))
                print("name2:", str(iid))
                response["msg"] = "用户名重复"
                response["error_num"] = 1
                return JsonResponse(response)

        userInfo.user_name = save_userInfo["user_name"]
        userInfo.user_address = save_userInfo["user_address"]
        userInfo.user_mobile = save_userInfo["user_mobile"]

        if request.FILES.get("user_image"):
            userInfo.user_avatar = request.FILES.get("user_image")

        save_addrInfo = json.loads(request.FILES.get("selectAddr").read())
        userInfo.user_province = save_addrInfo["province"]
        userInfo.user_city = save_addrInfo["city"]
        userInfo.user_area = save_addrInfo["area"]

        userInfo.save()

        response["image"] = str(userInfo.user_avatar)
        response["msg"] = "succes111s"
        response["error_num"] = 0
    except Exception as e:
        response["msg"] = str(e)
        response["error_num"] = 1

    return JsonResponse(response)
```

5. 用户主页显示所有商品

- **前端文件：** `customer_list.vue`
- **涉及的数据表：** `Product`
- **流程解析：**
  - 后端查询所有已上架的商品并传送至前端，前端展示在用户主页。
- **代码实现：**

```
    def show_customer_products(request):
        response = {}
        try:
            products = Product.objects.filter(product_status="上架")
            response["list"] = json.loads(serializers.serialize("json", products))
            response["msg"] = "success"
            response["error_num"] = 0
        except Exception as e:
            response["msg"] = str(e)
            response["error_num"] = 1
        return JsonResponse(response)
```

6. 用户将商品加入购物车

- **前端文件：** `customer_product.vue`
- **涉及的数据表：** `UserInfo`，`Product`，`CartItems`
- **流程解析：**
  - 后端从前端以 JSON 形式获取用户 ID 和商品 ID，检查购物车中是否已存在该商品。
  - 若已存在则增加商品数量，否则添加新的购物车条目。
- **代码实现：**

```
def add_to_cart(request):
    response = {}
    try:
        o1 = UserInfo.objects.get(id=request.GET.get("user_id"))
        o2 = Product.objects.get(id=request.GET.get("product_id"))
        cartItems = CartItems.objects.filter(user_id=o1, product_id=o2)
        if cartItems.count() == 0:
            cartItem = CartItems(user_id=o1, product_id=o2, num=1)
            cartItem.save()
        else:
            cartItem = CartItems.objects.get(user_id=o1, product_id=o2)
            cartItem.num += 1
            cartItem.save()

        response["msg"] = "success"
        response["error_num"] = 0
    except Exception as e:
        response["msg"] = str(e)
        response["error_num"] = 1
    return JsonResponse(response)
```

7. 用户购物车展示

- **前端文件：** `customer_cart.vue`
- **涉及的数据表：** `UserInfo`，`CartItems`
- **流程解析：**
  - 前端传递用户 ID 给后端，后端获取用户购物车中的商品信息并返回前端展示。
  - 计算购物车总费用并展示。
- **代码实现：**

```
def show_cart(request):
    response = {}
    try:
        o1 = UserInfo.objects.get(id=request.GET.get("user_id"))
        products = CartItems.objects.filter(user_id=o1)

        response["list"] = json.loads(serializers.serialize("json", products))
        response["msg"] = "success"
        response["error_num"] = 0
    except Exception as e:
        response["msg"] = str(e)
        response["error_num"] = 1
    return JsonResponse(response)

def cartProductsList(request):
    response = {}
    try:
        o1 = UserInfo.objects.get(id=int(request.GET.get("user_id")))
        items = (
            CartItems.objects.filter(user_id=o1)
            .values("product_id")
            .values_list("product_id", flat=True)
        )
        products = []
        for i in items:
            products.append(Product.objects.get(id=i))
        response["list"] = json.loads(serializers.serialize("json", products))
        response["msg"] = "success"
        response["error_num"] = 0
    except Exception as e:
        response["msg"] = str(e)
        response["error_num"] = 1
    return JsonResponse(response)
```

8. 用户搜索、筛选商品

- **前端文件：** `customer_search.vue`
- **涉及的数据表：** `Product`
- **流程解析：**
  - 前端提供商品类别作为筛选选项，用户可以通过搜索和筛选功能查找商品。
  - 前端传递搜索条件给后端，后端根据商品名模糊查询返回匹配商品信息。
- **代码实现：**

```
def show_searched_products(request):
    response = {}
    try:

        products = Product.objects.filter(product_status="上架").filter(
            product_name__contains=request.GET.get("search_text")
        )
        response["list"] = json.loads(serializers.serialize("json", products))
        response["msg"] = "success"
        response["error_num"] = 0
    except Exception as e:
        response["msg"] = str(e)
        response["error_num"] = 1
    return JsonResponse(response)
```

9. 用户删除购物车内的商品

- **前端文件：** `customer_cart.vue`
- **涉及的数据表：** `UserInfo`, `Product`, `CartItems`
- **流程解析：**
  - 前端传递用户 ID 和商品 ID 给后端，后端删除购物车中对应的商品条目。
- **代码实现：**

```
def delete_item_in_cart(request):
    response = {}
    try:
        o1 = UserInfo.objects.get(id=request.GET.get("user_id"))
        o2 = Product.objects.get(id=request.GET.get("product_id"))
        cartItems = CartItems.objects.get(user_id=o1, product_id=o2)

        cartItems.delete()
        response["msg"] = "success"
        response["error_num"] = 0
    except Exception as e:
        response["msg"] = str(e)
        response["error_num"] = 1
    return JsonResponse(response)
```

10. 用户修改购物车内的商品数量

- **前端文件：** `customer_cart.vue`
- **涉及的数据表：** `UserInfo`，`Product`，`CartItems`
- **流程解析：**
  - 前端传递用户 ID、商品 ID 和修改后的数量给后端，后端更新购物车中商品的数量信息。
- **代码实现：**

```
def edit_num_in_cart(request):
    response = {}
    try:
        o1 = UserInfo.objects.get(id=request.GET.get("user_id"))
        o2 = Product.objects.get(id=request.GET.get("product_id"))
        cartItems = CartItems.objects.get(user_id=o1, product_id=o2)

        if request.GET.get("num") == "1":
            cartItems.num += 1
            cartItems.save()
        else:
            if cartItems.num != 1:
                cartItems.num -= 1
                cartItems.save()
            else:
                cartItems.delete()
        response["msg"] = "success"
        response["error_num"] = 0
    except Exception as e:
        response["msg"] = str(e)
        response["error_num"] = 1
    return JsonResponse(response)
```

11. 用户下单

- **前端文件：** `customer_cart.vue`
- **涉及的数据表：** `OrderInfo`，`UserInfo`，`CartItems`，`Product`，`OrderProducts`
- **流程解析：**
  - 前端传递用户 ID 给后端，后端根据用户 ID 获取购物车商品信息。
  - 根据购物车中的商品逐个生成订单商品信息，并存入数据库表 `OrderProducts`。
  - 前端提示“购买成功”。
- **代码实现：**

```
def add_order(request):
    response = {}
    try:
        o1 = UserInfo.objects.get(id=request.GET.get("user_id"))
```

```
        user_cart = CartItems.objects.filter(user_id=o1)
        items = user_cart.values("product_id").values_list("product_id", flat=True)

        # 先找出购物车里涉及的所有商家
        business = []
        for i in items:
            business.append(Product.objects.get(id=i).product_business)

        business = list(set(business))

        # 遍历每个商家，写订单信息
        # 再找购物车里这个商家的所有商品，写一个订单的所有商品信息
        for k in business:
            o2 = k

            orderInfo = OrderInfo(
                customer_id=o1,
                business_id=o2,
                # order_createtime = ,
                order_status="买家未付款",
                customer_name=o1.user_name,
                customer_address=o1.user_address,
                customer_mobile=o1.user_mobile,
                customer_province=o1.user_province,
                customer_city=o1.user_city,
                customer_area=o1.user_area,
                business_name=o2.user_name,
                business_address=o2.user_address,
                business_mobile=o2.user_mobile,
                business_province=o2.user_province,
                business_city=o2.user_city,
                business_area=o2.user_area,
            )
            orderInfo.save()

            for i in user_cart:
                raw_product = Product.objects.get(id=i.product_id.id)
                if raw_product.product_business == o2:
                    order_items = OrderProducts(
                        order_id=orderInfo,
                        product_id=i.product_id,
                        num=i.num,
                        order_product_cost=raw_product.product_cost,
                    )
                    order_items.save()
                    raw_product.product_sales = raw_product.product_sales + i.num
                    raw_product.save()

        response["msg"] = "success"
        response["error_num"] = 0
    except Exception as e:
        response["msg"] = str(e)
```

```
        response["error_num"] = 1
    return JsonResponse(response)
```

12. 商家上架商品

- **前端文件：** `business_add_commodity.vue`
- **涉及的数据表：** `Product`
- **流程解析：**
  - 商家上架商品，前端将商品信息传递给管理员审批。
  - 管理员同意后，前端将商品信息传递给后端，后端将商品存入数据库表 `Product`。
- **代码实现：**

```
def add_product(request):
    global product
    iid = int(str(request.FILES.get("user_id").read())[2:-1])
    product_business_obj = UserInfo.objects.get(id=iid)
    product_Info = json.loads(request.FILES.get("forms").read())

    if request.FILES.get("product_image1"):
        product_image1 = request.FILES.get("product_image1")
    else:
        product_image1 = "avatar/default_pic.jpg"
    if request.FILES.get("product_image2"):
        product_imageDetail1 = request.FILES.get("product_image2")
    else:
        product_imageDetail1 = "avatar/default_pic.jpg"

    response = {}
    if request.method == "POST":
        # goods_image = request.FILES.get('goods_image')
        t = 0.0
        print(product_Info["product_cost"])
        if product_Info["product_cost"].find("."):
            t = float(product_Info["product_cost"])
        else:
            t = 1.0 * int(product_Info["product_cost"])
        product = Product(
            product_name=product_Info["product_name"],
            product_brand=product_Info["product_brand"],
            product_sales=0,
            product_stock=0,
            product_cost=t,
            product_color=product_Info["product_color"],
            product_image=product_image1,
            product_imageDetail=product_imageDetail1,
            product_business=product_business_obj,
            product_status="申请上架",
        )
        product.save()
```

```
        productcheck = CheckProduct(
            product_id=product,
            product_name=product_Info["product_name"],
            product_brand=product_Info["product_brand"],
            product_cost=t,
            product_color=product_Info["product_color"],
            product_image=product_image1,
            product_imageDetail=product_imageDetail1,
            product_business=product_business_obj,
            product_status="申请上架",
            check_status="待审核",
        )
        productcheck.save()
    try:

        response["msg"] = "success"
        response["error_num"] = 0
    except Exception as e:
        response["msg"] = str(e)
        response["error_num"] = 1

    return JsonResponse(response)
```

13. 商家对已上架商品的信息进行修改

- **前端文件：** `business_edit_commodity.vue`
- **涉及的数据表：** `Product`
- **流程解析：**
  - 商家修改商品信息，前端将修改后的商品信息传递给管理员审批。
  - 管理员同意后，前端将修改后的商品信息传递给后端，后端更新数据库表 `Product` 中对应的商品信息。
- **代码实现：**

```
def edit_product(request):

    # iid = int(str(request.FILES.get('user_id').read())[2:-1])
    pid = int(str(request.FILES.get("product_id").read())[2:-1])

    # product_business_obj = UserInfo.objects.get(id=iid)
    product_Info = json.loads(request.FILES.get("forms").read())

    p = Product.objects.get(id=pid)

    response = {}
    if request.method == "POST":

        t = 0.0
```

```
        if product_Info["product_cost"].find("."):
            t = float(product_Info["product_cost"])
        else:
            t = 1.0 * int(product_Info["product_cost"])

        """
        p.product_name=product_Info['product_name']
        p.product_brand=product_Info['product_brand']


        p.product_cost=t
        p.product_color=product_Info['product_color']
        #product_image=request.FILES.get('product_image1')
        #product_imageDetail=request.FILES.get('product_image2')
        if request.FILES.get('product_image1'):
            p.product_image=request.FILES.get('product_image1')
        if request.FILES.get('product_image2'):
            p.product_imageDetail=request.FILES.get('product_image2')

        p.save()
        """

        if request.FILES.get("product_image1"):
            product_image1 = request.FILES.get("product_image1")
        else:
            product_image1 = p.product_image
        if request.FILES.get("product_image2"):
            product_imageDetail1 = request.FILES.get("product_image2")
        else:
            product_imageDetail1 = p.product_imageDetail

        productcheck = CheckProduct(
            product_id=p,
            product_name=product_Info["product_name"],
            product_brand=product_Info["product_brand"],
            product_cost=t,
            product_color=product_Info["product_color"],
            product_image=product_image1,
            product_imageDetail=product_imageDetail1,
            product_business=p.product_business,
            product_status="申请修改",
            check_status="待审核",
        )
        productcheck.save()
    try:

        response["msg"] = "success"
        response["error_num"] = 0
    except Exception as e:
        response["msg"] = str(e)
        response["error_num"] = 1
```

```
    return JsonResponse(response)
```

14. 展示商家已上架的所有商品

- **前端文件：** `business_commodity.vue`, `business_saledata.vue`
- **涉及的数据表：** `Product`
- **流程解析：**
  - 后端根据商家 ID 查询其上架的所有商品信息，并传递给前端展示。
- **代码实现：**

```
def show_products(request):
    response = {}
    try:
        products = Product.objects.filter()
        response["list"] = json.loads(serializers.serialize("json", products))
        response["msg"] = "success"
        response["error_num"] = 0
    except Exception as e:
        response["msg"] = str(e)
        response["error_num"] = 1
    return JsonResponse(response)
```

15、商家增加库存

- **前端文件：** `edit_commodity_stock.vue`
- **涉及的数据表：** `Product`, `StockInfo`
- **流程分析：**
  - 商家通过前端界面输入商品和增加的库存量，传递至后端。
  - 后端验证并更新 `Product` 表中的商品信息，并更新 `StockInfo` 表中的库存信息。
  - 更新成功后，前端提示“库存添加成功”。
- **代码实现：**

```
def add_product_stock(request):
    response = {}
    try:
        product = Product.objects.get(id=request.GET.get("product_id"))

        stock = int(request.GET.get("stock"))

        product.product_stock += stock
        product.save()

        cost = 0.0

        if request.GET.get("cost").find("."):
            cost = float(request.GET.get("cost"))
```

```
        else:
            cost = 1.0 * int(request.GET.get("cost"))

        stockInfo = StockInfo(
            business_id=product.product_business,
            product_id=product,
            stock_num=stock,
            stock_cost=cost,
            business_province=product.product_business.user_province,
            business_city=product.product_business.user_city,
            business_area=product.product_business.user_area,
        )
        stockInfo.save()

        response["msg"] = "success"
        response["error_num"] = 0
    except Exception as e:
        response["msg"] = str(e)
        response["error_num"] = 1
    return JsonResponse(response)
```

16、展示商家营业状况

- **前端文件：** `business_saledata.vue`
- **涉及的数据表：** `UserInfo`，`OrderInfo`，`StockInfo`
- **流程分析：**
  - 商家访问销售页面时，前端向后端传递商家信息。
  - 后端根据商家信息查询相关订单（包括销售和采购），按时间排序并返回给前端。
  - 前端利用动态图表展示销售数据，支持按周次和不同粒度（如商品、地区）进行数据分析。
- **代码实现：**

```
def show_business_orderProduct(request):
    response = {}
    try:
        # 所有订单
        o1 = UserInfo.objects.get(id=request.GET.get("business_id"))
        orderIds = OrderInfo.objects.filter(business_id=o1).order_by(
            "-order_createtime"
        )

        # 商家的所有订单 涉及到的用户头像
        userAvatar = []
        for i in orderIds:
            userAvatar.append(i.customer_id)

        # 所有订单涉及到的商品
        orderProduct = []
        nowProduct = []
```

```
        for i in orderIds:
            t = OrderProducts.objects.filter(order_id=i)
            orderProduct.extend(t)
            for j in t:
                nowProduct.append(j.product_id)

        # 去重
        nowProduct = list(set(nowProduct))

        response["orders"] = json.loads(serializers.serialize("json", orderIds))
        response["orderProducts"] = json.loads(
            serializers.serialize("json", orderProduct)
        )
        response["nowProduct"] = json.loads(serializers.serialize("json",
nowProduct))
        response["userAvatar"] = json.loads(serializers.serialize("json",
userAvatar))
        response["msg"] = "success"
        response["error_num"] = 0
    except Exception as e:
        response["msg"] = str(e)
        response["error_num"] = 1
    return JsonResponse(response)
```

17、导出商家营业数据分析

- **前端文件：** `business_saledata.vue`
- **涉及的数据表：** `Product`, `StockInfo`
- **功能介绍：**
  - 商家可以通过前端界面导出进货、销售数据到 Excel 文件。
  - 提供可视化的销售数据导出功能，使商家能够更方便地进行数据分析和报告生成。
- **代码实现：**

```
<template>
              <download-excel
              class="export-excel-wrapper"
              :data="DetailsForm"
              :fields="json_fields"
              :header="Title"
              name="导出销量.xls"
            >
              <!-- 上面可以自定义自己的样式，还可以引用其他组件button -->
              <el-button type="success">导出excel</el-button>
              </download-excel>
              </template>
```

```
readExcel(e) {
      // 读取表格文件
```

```
    let that = this;
    const files = e.target.files;


    if (files.length <= 0) {
      return false;
    } else if (!/\.(xls|xlsx)$/.test(files[0].name.toLowerCase())) {
      this.$message({
        message: "上传格式不正确，请上传xls或者xlsx格式",
        type: "warning"
      });
      return false;
    } else {
      // 更新获取文件名
      that.upload_file = files[0].name;
    }



    const fileReader = new FileReader();
    fileReader.onload = ev => {
      try {
        const data = ev.target.result;

        const workbook = read(data, {
          type: "binary"
        });

        const wsname = workbook.SheetNames[0]; //取第一张表
        const ws = utils.sheet_to_json(workbook.Sheets[wsname]); //生成json表格内容
        //ws就是读取的数据（不包含标题行即表头，表头会作为对象的下标）

        //this.submit_form();

                  console.log(ws);

      } catch (e) {

        console.log(e);
        return false;
      }
    };
    console.log("rrrttt")
    fileReader.readAsBinaryString(files[0]);
  },
```

18、用户、商家查看订单（以查看“买家未到货”状态为例）

- **前端文件：** `order_inDelivery.vue`
- **涉及的数据表：** `UserInfo`，`OrderInfo`
- **流程分析：**
  - 用户可以根据订单状态筛选查看订单，例如查看“买家未到货”的订单。
  - 前端传递用户信息和订单状态至后端，后端查询并返回符合条件的订单信息给前端。
  - 前端显示订单信息和相关商品信息供用户查看。
- **代码实现：**

```
def show_business_orderProduct_status(request):
    response = {}
    try:
        # 所有订单
        o1 = UserInfo.objects.get(id=request.GET.get("business_id"))
        orderIds = (
            OrderInfo.objects.filter(business_id=o1)
            .filter(order_status=request.GET.get("status"))
            .order_by("-order_createtime")
        )

        # 商家的所有订单 涉及到的用户头像
        userAvatar = []
        for i in orderIds:
            userAvatar.append(i.customer_id)

        # 所有订单涉及到的商品
        orderProduct = []
        nowProduct = []
        for i in orderIds:
            t = OrderProducts.objects.filter(order_id=i)
            orderProduct.extend(t)
            for j in t:
                nowProduct.append(j.product_id)

        # 去重
        nowProduct = list(set(nowProduct))

        response["orders"] = json.loads(serializers.serialize("json", orderIds))
        response["orderProducts"] = json.loads(
            serializers.serialize("json", orderProduct)
        )
        response["nowProduct"] = json.loads(serializers.serialize("json",
nowProduct))
        response["userAvatar"] = json.loads(serializers.serialize("json",
userAvatar))
        response["msg"] = "success"
        response["error_num"] = 0
    except Exception as e:
```

```
        response["msg"] = str(e)
        response["error_num"] = 1
    return JsonResponse(response)
```

19、修改订单状态（以商家修改订单信息为例）

- **前端文件：** `order_inDelivery.vue`
- **涉及的数据表：** `OrderInfo`
- **流程分析：**
  - 商家通过前端界面选择订单并修改其状态，例如将订单标记为“已发货”。
  - 前端将订单ID和修改后的状态传递给后端，后端更新订单状态并返回处理结果给前端。
  - 前端刷新页面显示更新后的订单状态，如“订单已发货”。
- **代码实现：**

```
def alterOrderStatus(request):
    response = {}
    try:
        order = OrderInfo.objects.get(id=request.GET.get("order_id"))
        order.order_status = request.GET.get("status")
        order.save()

        response["msg"] = "succes111s"
        response["error_num"] = 0
    except Exception as e:
        response["msg"] = str(e)
        response["error_num"] = 1

    return JsonResponse(response)
```

20、用户点赞商品，用户、商家查看商品点赞信息

- **前端文件：** `business_product_view.vue`, `customer_product_view.vue`
- **涉及的数据表：** `UserInfo`, `Product`, `LikeList`
- **流程分析：**
  - 用户或商家可以通过前端对商品进行点赞操作，后端根据用户和商品ID记录点赞信息到 `LikeList` 表。
  - 在商品查看页面，前端展示商品的点赞信息给用户或商家。
- **代码实现：**

```
def toggle_user_like_to_product(request):
    response = {}
    try:
        o1 = UserInfo.objects.get(id=request.GET.get("user_id"))
        product = Product.objects.get(id=request.GET.get("product_id"))
        a = LikeList.objects.filter(user_id=o1, product_id=product)
```

```
        if a.count() == 1:
            a[0].delete()
        else:
            newitem = LikeList(user_id=o1, product_id=product)
            newitem.save()

        response["msg"] = "success"
        response["error_num"] = 0
    except Exception as e:
        response["msg"] = str(e)
        response["error_num"] = 1
    return JsonResponse(response)
```

21、用户收藏店铺（即商家）

- **前端文件：** `business_product_view.vue`, `customer_product_view.vue`, `product_to_business.vue`
- **涉及的数据表：** `UserInfo`, `StarList`
- **流程分析：**
  - 用户可以通过前端收藏商家，前端传递用户和商家ID至后端记录到 `StarList` 表中。
  - 在商品查看页面，前端展示用户收藏的商家信息。
- **代码实现：**

```
def toggle_user_star_to_shop(request):
    response = {}
    try:
        o1 = UserInfo.objects.get(id=request.GET.get("user_id"))
        shop = UserInfo.objects.get(id=request.GET.get("business_id"))
        a = StarList.objects.filter(star_customer_id=o1, star_business_id=shop)
        if a.count() == 1:
            a[0].delete()
        else:
            newitem = StarList(star_customer_id=o1, star_business_id=shop)
            newitem.save()

        response["msg"] = "success"
        response["error_num"] = 0
    except Exception as e:
        response["msg"] = str(e)
        response["error_num"] = 1
    return JsonResponse(response)

def get_user_star_to_shop(request):
    response = {}
    try:
        o1 = UserInfo.objects.get(id=request.GET.get("user_id"))
        shop = UserInfo.objects.get(id=request.GET.get("business_id"))
        a = StarList.objects.filter(star_customer_id=o1, star_business_id=shop)
        if a.count() == 1:
```

```
            response["star"] = "star"
        else:
            response["star"] = "not"

        response["msg"] = "success"
        response["error_num"] = 0
    except Exception as e:
        response["msg"] = str(e)
        response["error_num"] = 1
    return JsonResponse(response)

def get_shop_stars(request):
    response = {}
    try:
        shop = UserInfo.objects.get(id=request.GET.get("business_id"))

        a = StarList.objects.filter(star_business_id=shop)
        p = a.count()
        response["stars"] = p
        response["msg"] = "success"
        response["error_num"] = 0
    except Exception as e:
        response["msg"] = str(e)
        response["error_num"] = 1
    return JsonResponse(response)
```

22、安全性检查

- **前端文件：** 多个前端文件
- **涉及的数据表：** `UserInfo`
- **功能介绍：**
  - 在商家上架商品和用户下单时，系统检查商家和用户的个人信息是否完善，若不完善则提醒补充。
- **代码实现：**

```
async addCommodity() {
  let m = [];
  try {
    const response = await this.axios.get('fetch_userInfo', {
      params: {
        user_id: this.$store.state.userId
      }
    });
    console.log(response);
    m = response.data;
  } catch (error) {
    console.log(error);
  }
```

```
  if (m.user_mobile === "") {
    console.log("null");
    window.alert("请完善联系方式");
    this.$router.push({ path: "/business/business_information" });
    return;
  }

  if ((m.user_province === "") || (m.user_city === "") || (m.user_area === "")) {
    window.alert("请完善地址");
    this.$router.push({ path: "/business/business_information" });
    return;
  }

  if (m.user_address === "") {
    window.alert("请完善详细地址");
  }
}
```

23、管理员查看用户登录记录

- **前端文件：** `user_log.vue`
- **涉及的数据表：** `UserInfo`,`UserLog`
- **功能介绍：**
  - 管理员可以查看用户和商家的登录记录，记录包括用户名、登录时间、登录地点和用户类型。
- **代码实现：**

```
def add_log(request):
    response = {}
    try:
        iid = request.GET.get("user_id")
        o1 = UserInfo.objects.get(id=iid)
        l = UserLog(
            log_id=o1,
            log_name=o1.user_name,
            log_type=o1.user_type,
            log_province=o1.user_province,
            log_city=o1.user_city,
            log_area=o1.user_area,
        )
        l.save()

        response["msg"] = "succes111s"
        response["error_num"] = 0
    except Exception as e:
        response["msg"] = str(e)
        response["error_num"] = 1

    return JsonResponse(response)
```

```
def get_user_log(request):
    response = {}
    try:
        o = UserLog.objects.order_by("-log_time")
        response["list"] = json.loads(serializers.serialize("json", o))
        response["msg"] = "succes111s"
        response["error_num"] = 0
    except Exception as e:
        response["msg"] = str(e)
        response["error_num"] = 1

    return JsonResponse(response)
```

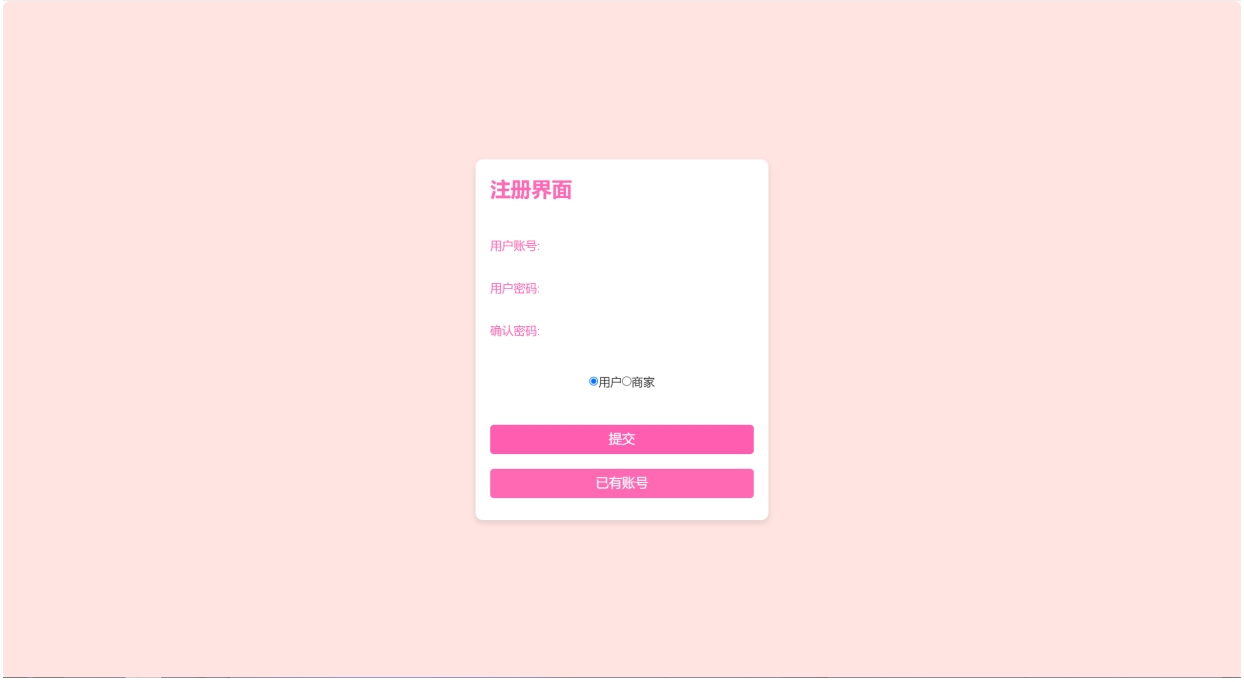
24、管理员审核商品上架、修改申请

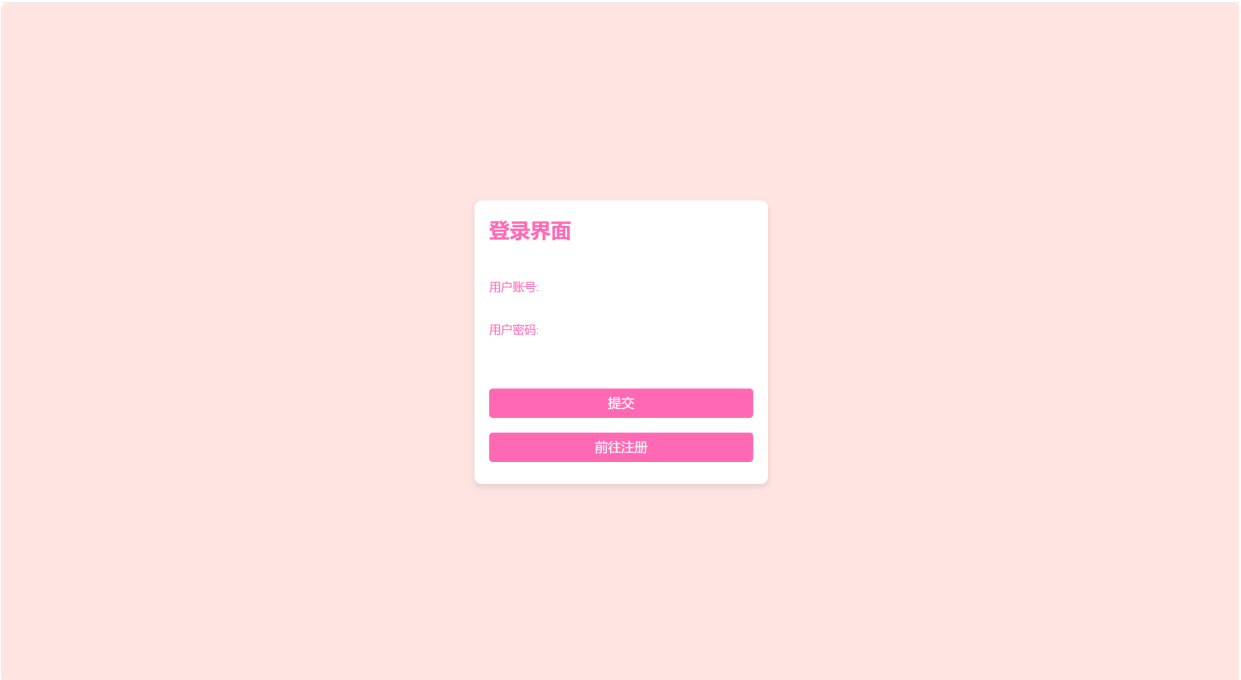
- **前端文件：** `editCheck.vue`, `hasCheck.vue`, `rejectCheck.vue`, `toCheck.vue`
- **涉及的数据表：** `CheckProduct`, `Product`
- **功能介绍：**
  - 商家提交商品上架或修改申请需要经过管理员审核，审核通过后才能进行相应操作。
- **代码实现：**

```
def alterCheckStatus(request):
    response = {}
    try:
        result = request.GET.get("status")
        iid = int(request.GET.get("order_id"))

        o1 = CheckProduct.objects.get(id=iid)
        o1.check_status = result
        o1.save()

        pid = o1.product_id.id

        o2 = Product.objects.get(id=pid)
        if result == "通过审核":
            o2.product_status = "上架"
            o2.save()
        if result == "回退申请":
            o2.product_status = "下架"
            o2.save()

        response["msg"] = "succes111s"
        response["error_num"] = 0
    except Exception as e:
        response["msg"] = str(e)
        response["error_num"] = 1

    return JsonResponse(response)
```

6 系统运行实例

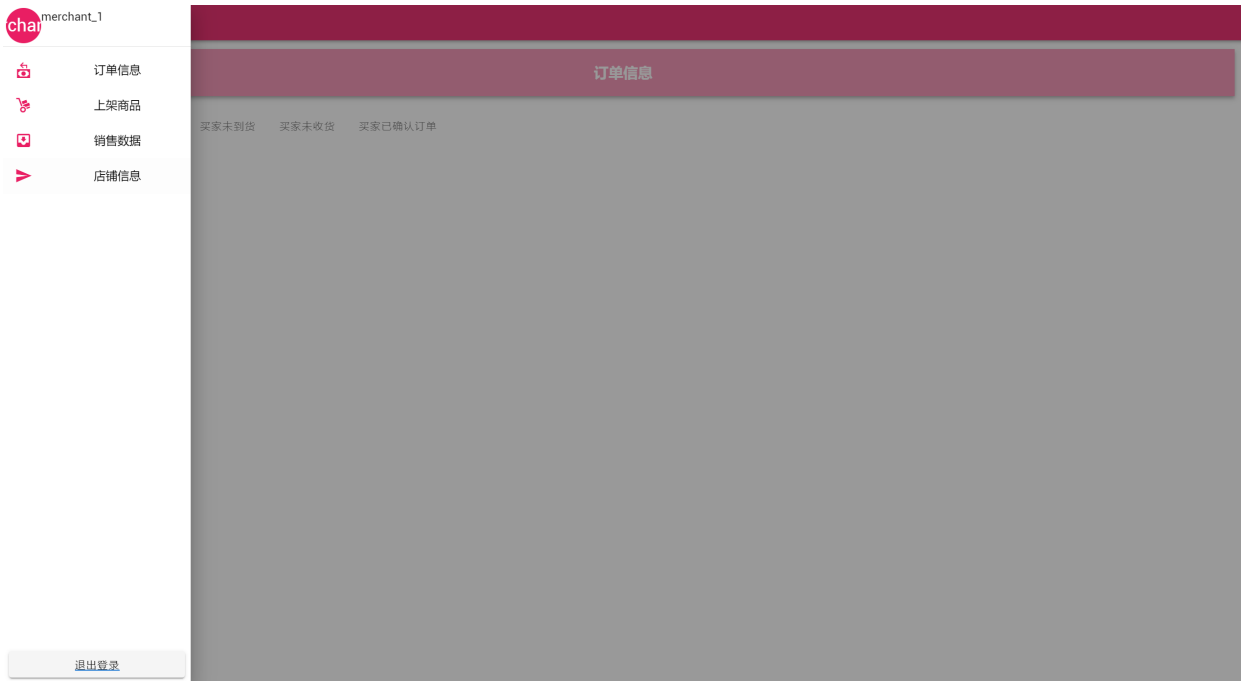
6.1 登录与注册界面





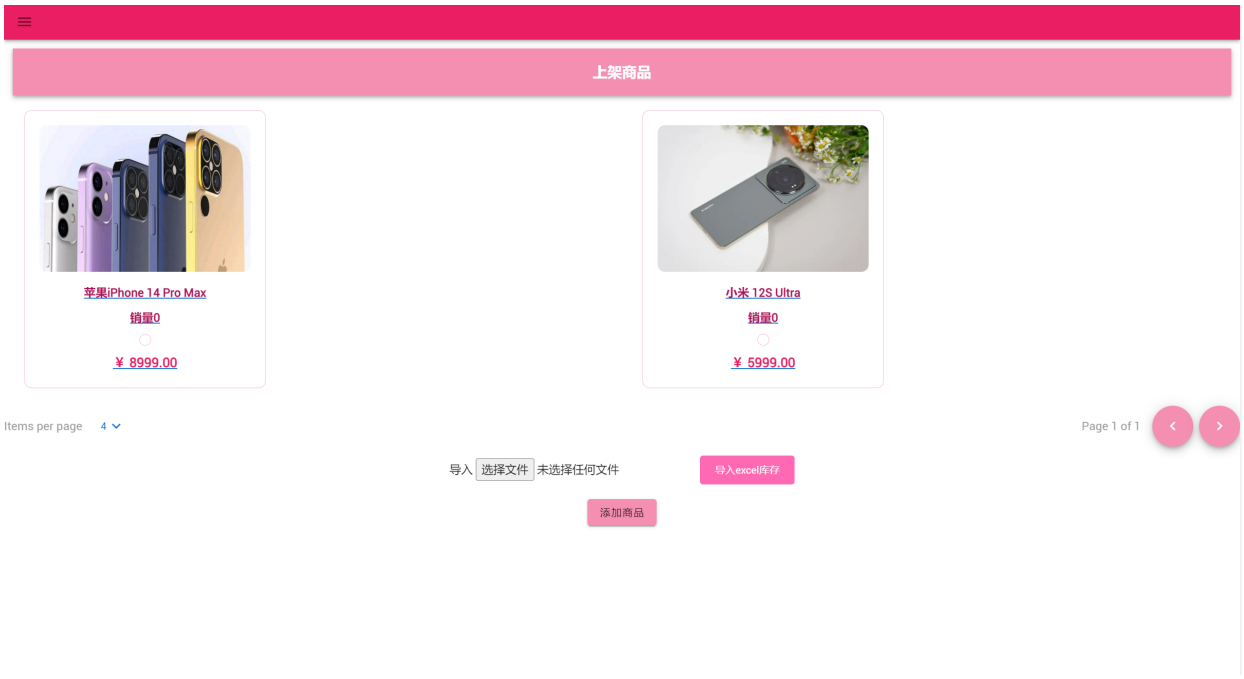
6.2 商家界面

6.2.1 商家界面侧边索引栏



6.2.2 商家个人信息页与修改功能

6.2.3 上架商品展示页



6.2.4 商品详情页

6.2.5 商品上架界面

6.2.6 订单信息

6.2.7 营业情况

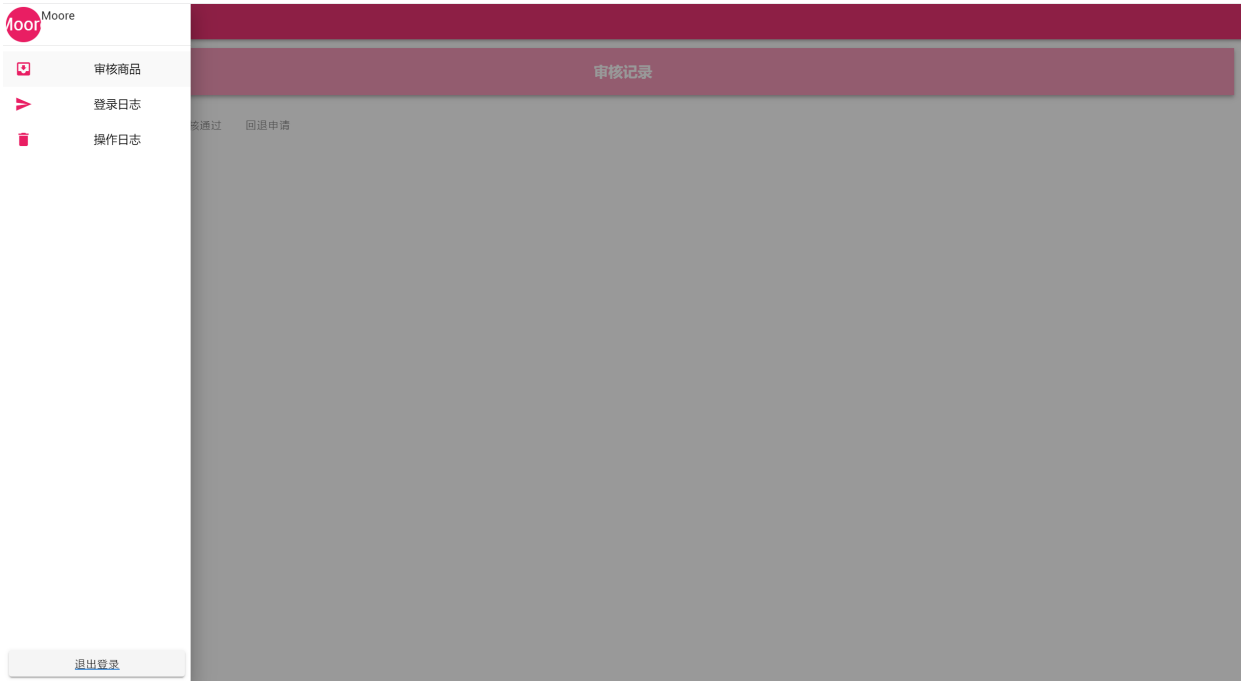
6.3 买家界面

6.3.1 买家界面侧边索引栏

6.3.2 商品展示页面与搜索功能

6.3.3 购物车界面

6.4 管理员界面


Moore
审核商品
登录日志
操作日志
退出登录
审核记录
通过
回退申请