**【校园跳蚤市场系统】**

**详细设计说明书**

**[版本号V\_1.0]**

**2022年11月20日**

**组长：杜康睿**

**副组长：李佳骏**

**组员：吴羽腾 卢晓雅 湛馥榕**

**目录**

[**【校园跳蚤市场系统】** 1](#_Toc121857626)

[**1.引言** 3](#_Toc121857627)

[**1.1编写目的** 3](#_Toc121857628)

[**1.2背景** 3](#_Toc121857629)

[**1.3定义** 3](#_Toc121857630)

[**1.4参考资料** 3](#_Toc121857631)

[**2. 系统结构** 3](#_Toc121857632)

[**2.1 模块划分** 3](#_Toc121857633)

[**2.2数据库接口说明** 4](#_Toc121857634)

[**3．用户账户模块详细设计** 5](#_Toc121857635)

[**3.1 注册和登录** 5](#_Toc121857636)

[**3.2 账户管理** 8](#_Toc121857637)

[**3.3 商品管理** 9](#_Toc121857638)

[**3.4 心愿清单** 10](#_Toc121857639)

[**3.5 订单记录** 12](#_Toc121857640)

[4.  **商品展示模块详细设计** 12](#_Toc121857641)

[**5. 支付模块设计详细设计** 13](#_Toc121857642)

[**5.1 运输方式选择模块** 14](#_Toc121857643)

[**5.2 运输地址选择模块** 15](#_Toc121857644)

[**5.3 结账完成模块** 16](#_Toc121857645)

[**6．评论模块详细设计** 17](#_Toc121857646)

[**7．购物车模块设计详细设计** 18](#_Toc121857647)

[**8. 管理模块** 20](#_Toc121857648)

**1.引言**

**1.1编写目的**

本文档的目的是描述校园跳蚤市场系统的详细设计，主要内容包括网校园跳蚤市场系统的总体设计和模块详细。

本文档的预期读者为软件工程师和编码人员，供其具体实现时使用。

**1.2背景**

软件名称：校园跳蚤市场系统

数据库名称：SQLite

任务提出者：杜康睿 李佳骏

开发者：吴羽腾 卢晓雅 湛馥榕

用户：有商品交易需求的大学生

**1.3定义**

[列出本文件中用到的专门术语的定义和外文首字母组词的原词组。]

ID：identification缩写，指数据库中对每个记录的唯一性标识。

购物车：用户即将购买的商品列表。

心愿清单：用户感兴趣并想要持续关注的商品对象列表。

**1.4参考资料**

[1]田文洪.软件工程——共同演进的方法与实践. 机械工业出版社.2013

**2. 系统结构**

**2.1 模块划分**

根据功能分解，校园跳蚤市场系统被划分为用户端子系统和管理端子系统。其中用户端包含的顶层模块有用户账户模块、商品展示模块、支付模块、评论模块、购物车模块；管理端包含的顶层模块有商品管理和用户管理模块。

模块之间的关系如下图2-1-1所示。

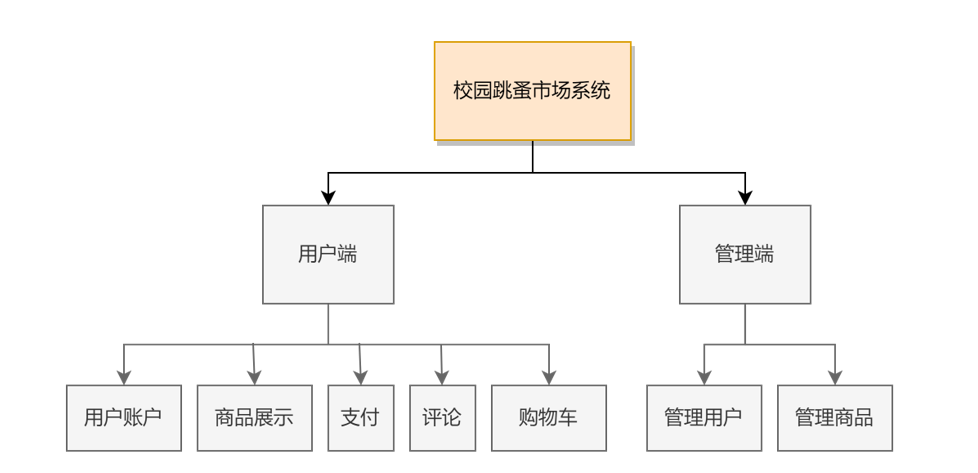


图2-1-1 系统结构

**2.2数据库接口说明**

1、连接数据库

输出：连接成功或失败

2、添加记录

向数据库中添加记录，如用户成功注册时向数据库添加用户信息。

输入：相应的记录信息，如添加用户信息时输入为用户名、密码、邮箱。

输出：添加成功或失败

3、记录筛选

用户登录或进行某些操作时需要核对用户名和密码是否匹配来进行认证。

输入：筛选条件（某属性包含）

输出：筛选成功或失败，筛选得到的记录

响应函数：Model.objects.filter(condition)

4、记录获取

根据记录的ID（主键）获取对应的记录

输入：该记录的ID

输出：获取成功或失败，获取的记录

响应函数：Model.objects.gets(pk=id)

5、记录更新

输入：更新条件

输出：更新成功或失败

响应函数：Model.objects.update(condition)

6、记录删除

输入：该记录的ID

输出：删除成功或失败

响应函数：Model.objects.delete()

**3．用户账户模块详细设计**

用户账户模块主要由五个部分组成，即注册和登录、账户管理、商品管理、心愿清单和订单记录部分。

**3.1 注册和登录**

|  |
| --- |
| 用户注册模块设计 |
| 1.1 模块描述：用户使用本人邮箱进行账号注册。 |
| 1.2 功能：使用邮箱注册账号，需要验证邮箱才能成功注册。要求邮箱有效且未被注册过，两次密码输入需要一致才能成功注册。 |
| 1.3 性能：在从外界输入错误邮箱的时候程序不受影响并成功返回错误信息。 |
| 1.4 输入项：用户的邮箱、账户名、两次输入一致的密码。 |
| 1.5 输出项：未注册成功向用户输出未注册成功的原因，注册成功则可用刚注册账号进行成功登录。 |
| 1.6 接口：连接数据库中用户表的接口。 |
| 1.7 设计模式：用户点击注册按钮后，判断用户提交的表单是否符合字段约束，如果符合则接下来进行邮箱验证阶段，不符合则提示用户相应的错误，邮箱验证成功后，注册成功并自动登录。 |
| 1.8 流程逻辑：实现用户注册的简要逻辑如下图3-1-1。 |

|  |
| --- |
| 用户登录模块设计 |
| 1.1 模块描述：用户使用自己的邮箱进行账号登录。 |
| 1.2 功能：使用邮箱与密码进行登录，需要成功匹配才能登录成功。 |
| 1.3 性能：在从外界输入其他无关操作的时候程序不受影响。能够及时响应。 |
| 1.4 输入项：用户的邮箱、密码。 |
| 1.5 输出项：登录成功返回商城首页，登录失败返回登录界面并提示信息。 |
| 1.6 接口：连接数据库中用户表的接口。 |
| 1.7 设计模式：用户点击登录按钮后，判断邮箱密码是否匹配，匹配成功则成功登录，进入商城首页；否则返回登录界面并提示密码或信息错误。 |
| 1.8 流程逻辑：实现用户登录的简要逻辑如下图3-1-5。 |

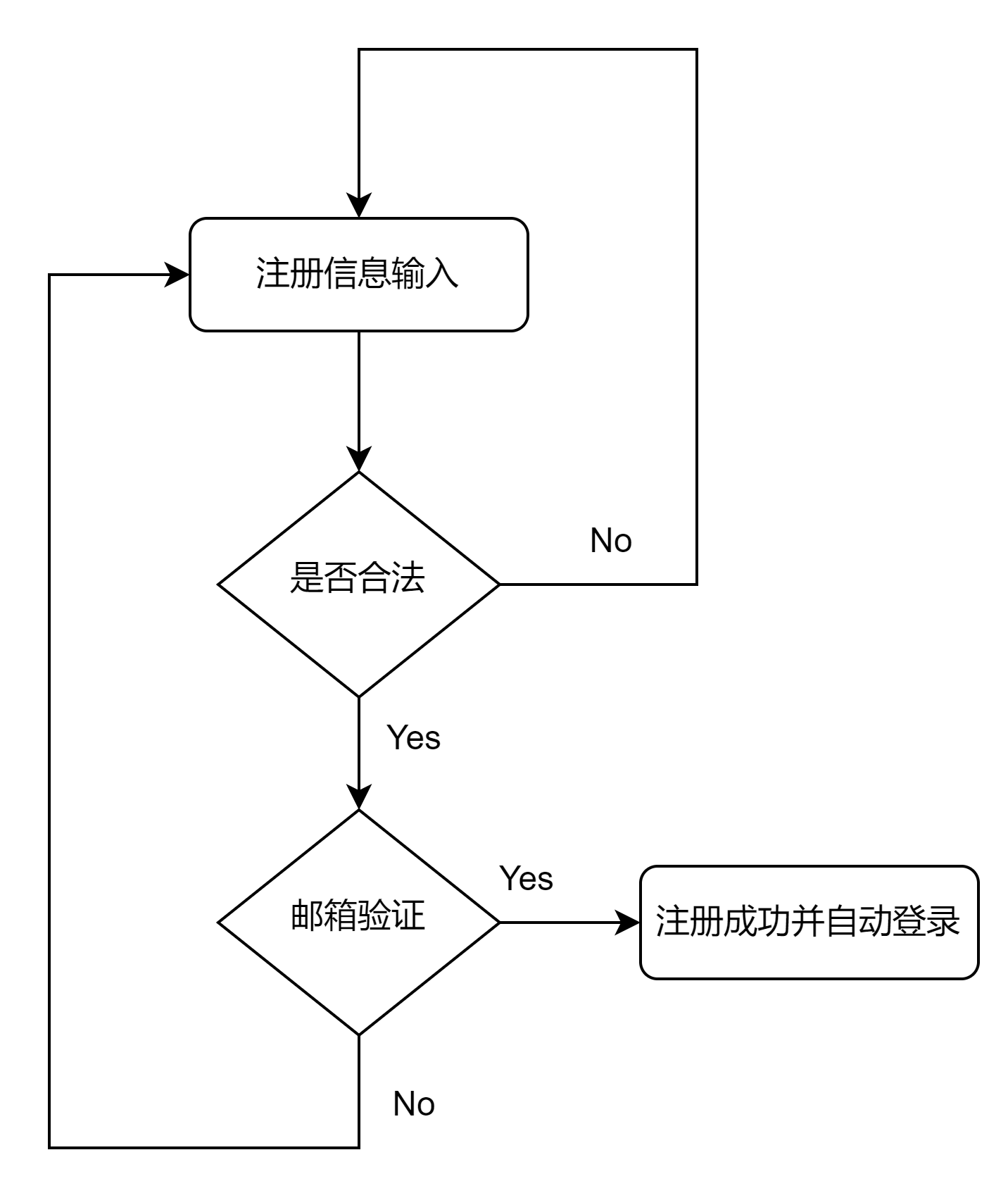


图3-1-2 注册流程

注册模块的代码如下图3-1-3所示：

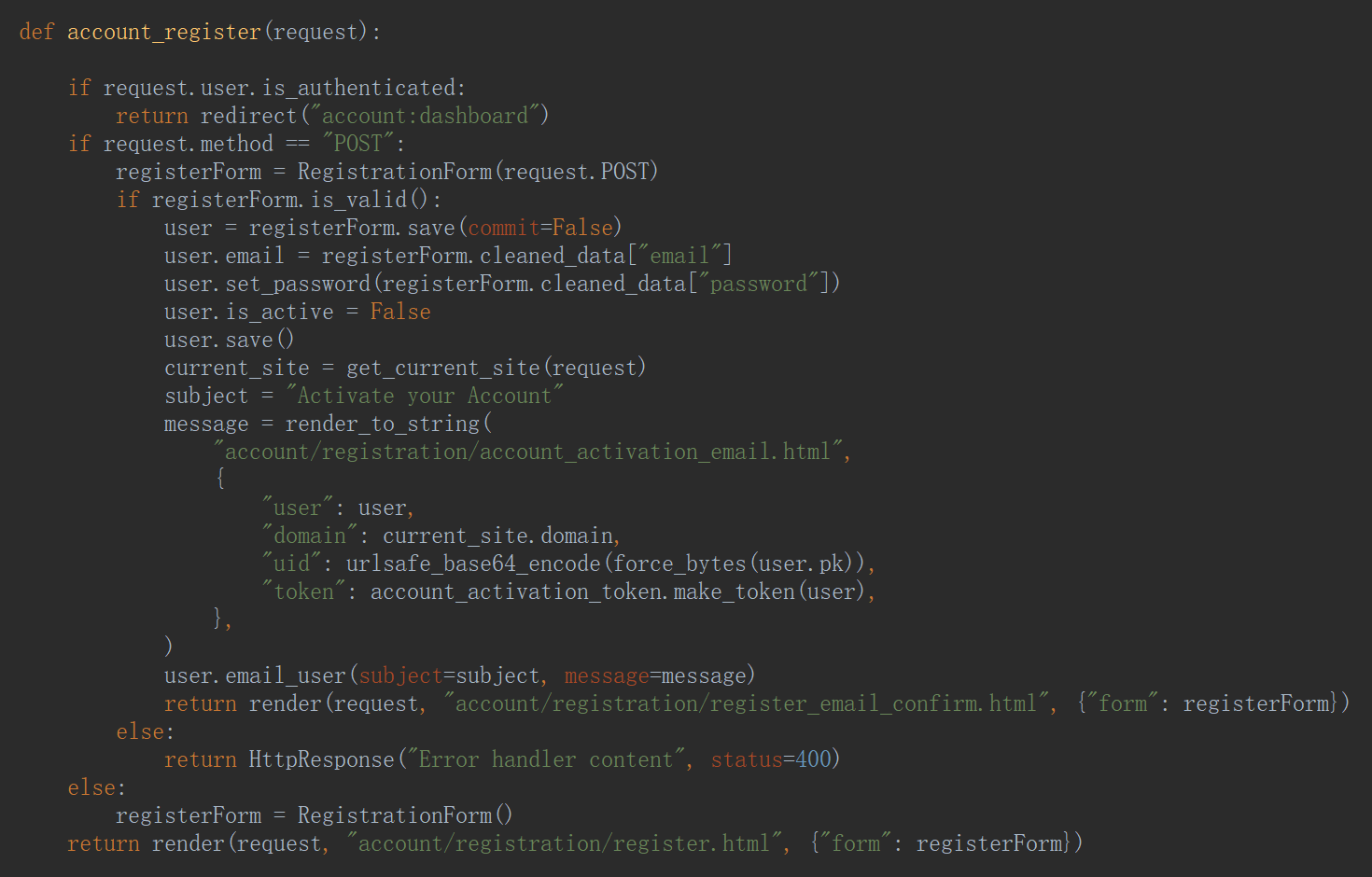


图3-1-3 注册代码

邮箱验证成功后，激活注册的账号，如图3-1-4。

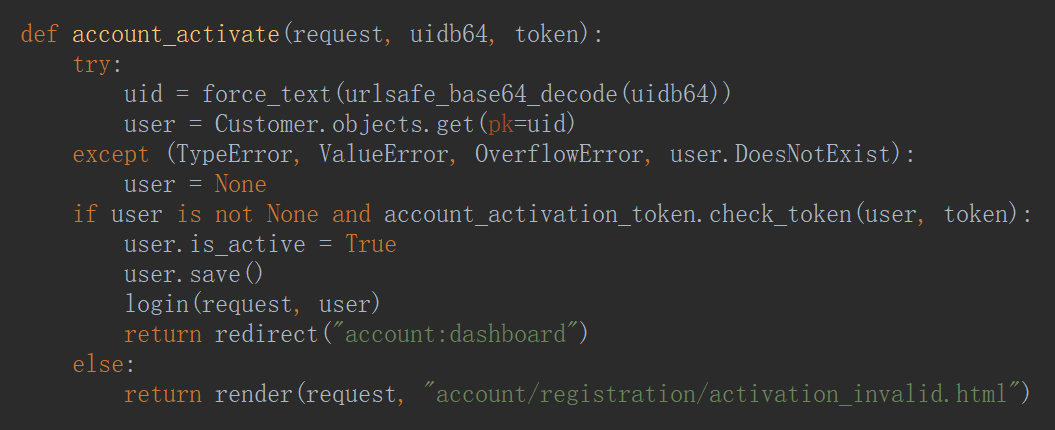


图3-1-4 邮箱激活

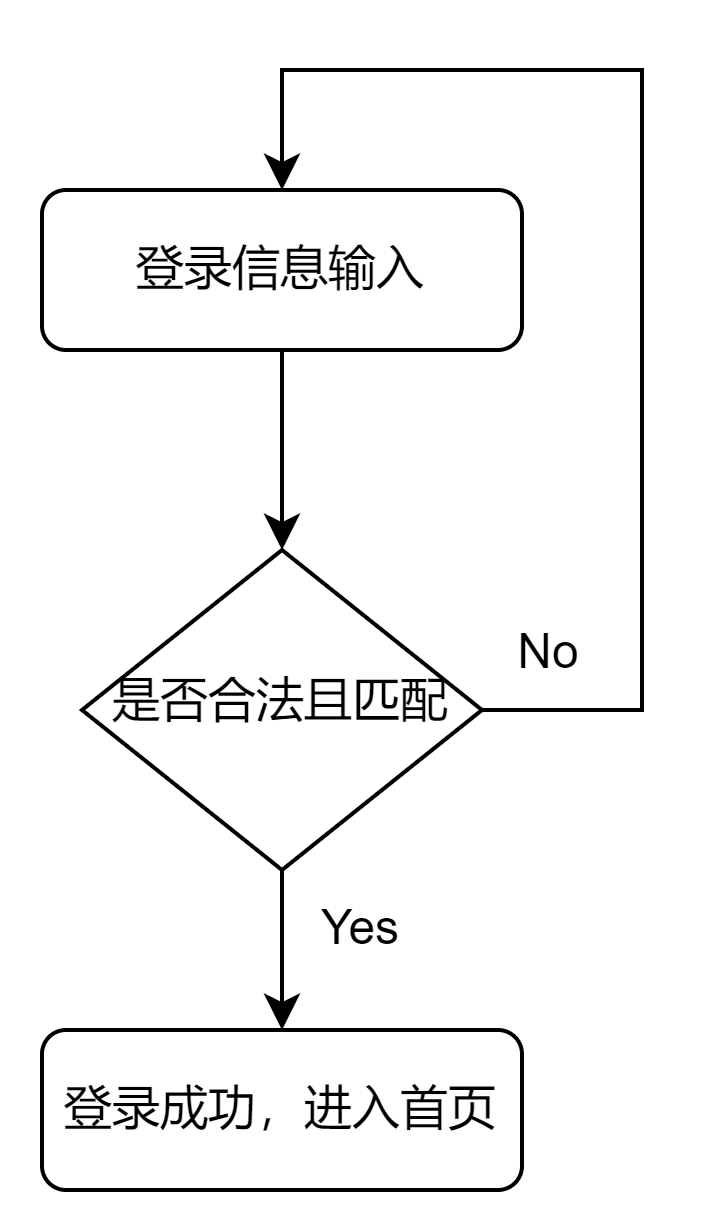


图3-1-5 登录流程

登录的前端网址转换如图3-1-6所示，当网址为login时，返回名为account/login.html的前端界面。

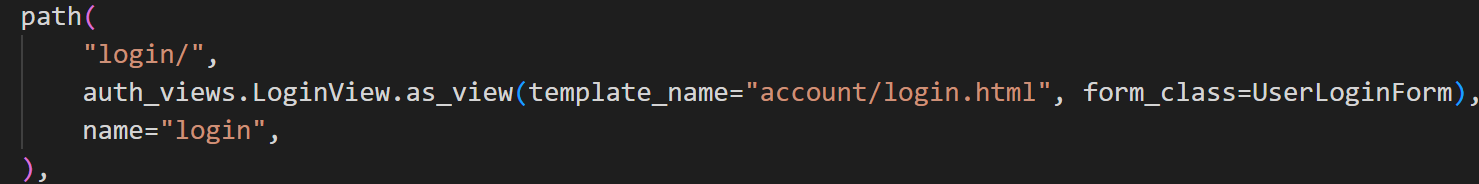


图3-1-6 登录url转换

**3.2 账户管理**

该部分分为个人信息更新和账户注销两个底层模块。

其中，个人信息更新模块的展开设计如下：

|  |
| --- |
| 个人信息更新模块设计 |
| 1.1 模块描述：实现用户账户管理的模块。 |
| 1.2 功能：更改用户邮箱、电话等个人信息。 |
| 1.3 性能：在从外界输入其他无关操作的时候程序不受影响。 |
| 1.4 输入项：用户个人信息表单。 |
| 1.5 输出项：响应更改操作是否成功，展示新的个人信息。 |
| 1.6 接口：连接数据库中用户数据表的接口，返回首页的跳转按键。 |
| 1.7 设计模式：用户点击提交按钮后，判断用户提交的表单是否符合字段约束，如果符合则更新记录用户信息的数据表，不符合则提示用户相应的错误。 |
| 1.8 流程逻辑：实现个人信息更改的简要逻辑如下图：图3-2-1. 个人信息更改流程逻辑图 |

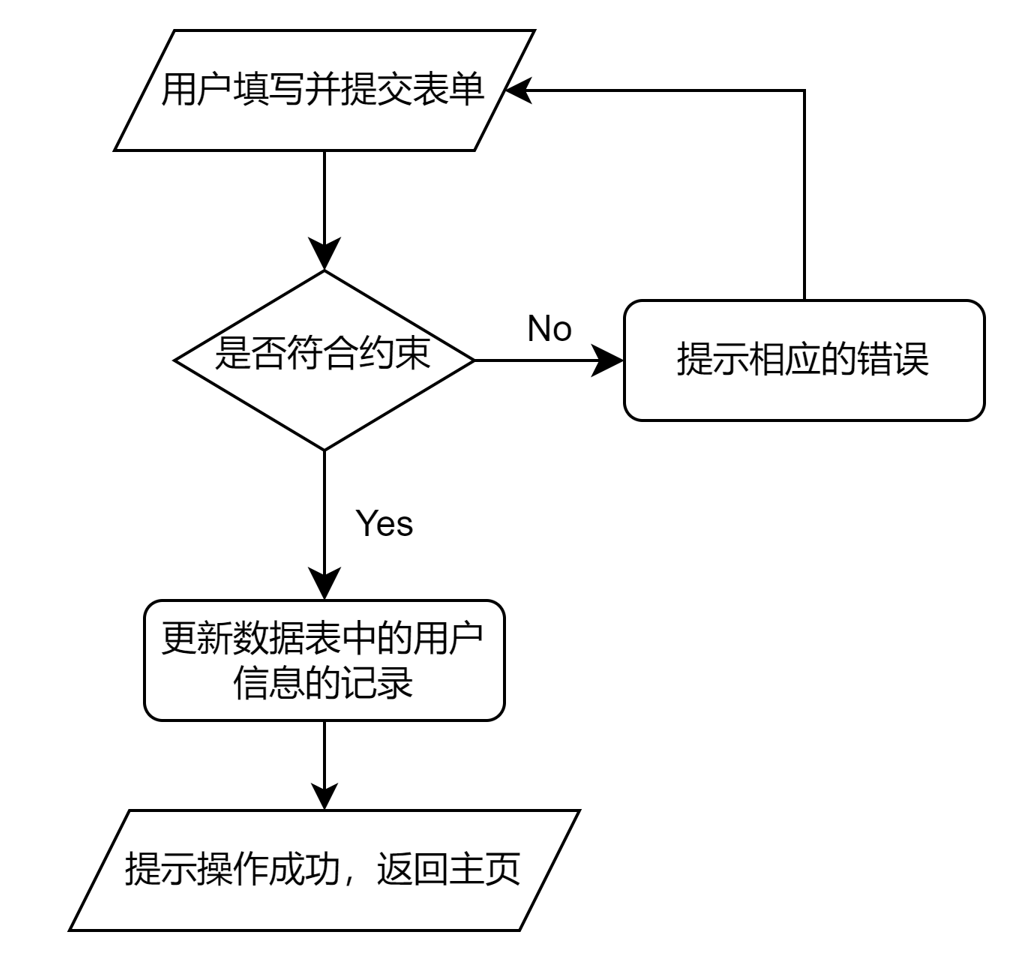


图3-2-1. 个人信息更改流程逻辑图

**3.3 商品管理**

该部分分为商品添加、商品更新、商品删除、查看该商品的订单四个底层模块。

其中，商品添加模块的展开设计如下：

|  |
| --- |
| 1.3 设计方法： |
| 1.1 模块描述：实现商品添加的模块。 |
| 1.2 功能：用户添加自己要售卖的商品。 |
| 1.3 性能：在从外界输入其他无关操作的时候程序不受影响。 |
| 1.4 输入项：待售商品信息表单。 |
| 1.5 输出项：提示表单的填写错误，响应上传操作是否成功。 |
| 1.6 接口：连接数据库中商品数据表的接口。 |
| 1.7 设计模式：用户点击提交按钮后，判断用户提交的表单是否符合字段约束，如果符合则在商品数据表中增加一行，填入该商品的信息，不符合则提示用户相应的错误。 |
| 1.8 流程逻辑：实现添加商品的简要逻辑如下图：图3-3-1. 添加商品流程逻辑。 |

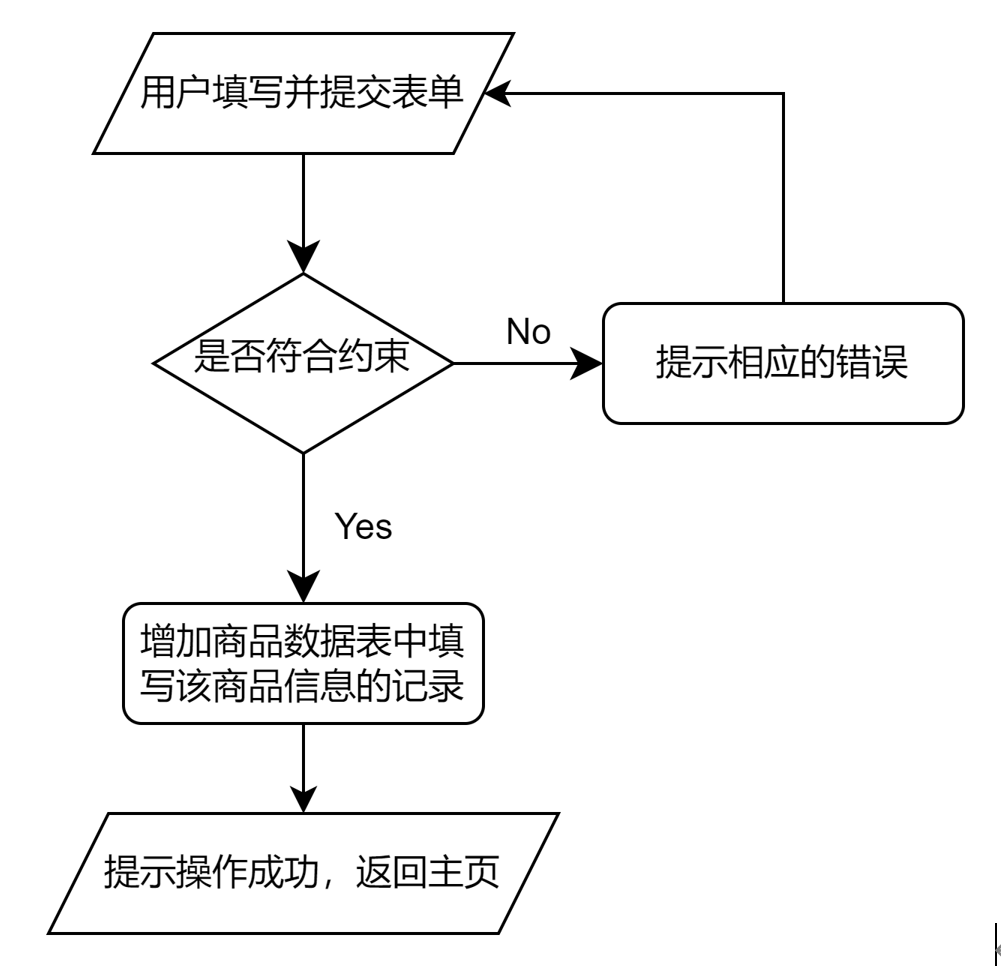


图3-3-1. 添加商品流程逻辑

**3.4 心愿清单**

心愿清单存放用户心愿商品列表，包括3个组件：路由转换组件url、视图组件view以及设计界面user\_wish\_list.html。其中，路由组件url包含对两个网址的转换，分别是wishlist和wishlist/add\_to\_wishlist/<int:id>。视图组件view包含两个函数，即wishlist(request)和add\_to\_wishlist(request, id)。

|  |
| --- |
| 1.1模块描述：用户心愿的商品列表 |
| 1.2 功能：用户用此模块来存放暂时还不想加入购物车，但以后可能购买的商品。也可以添加用户想要购买，但已经售罄的商品。 |
| 1.3 设计方法：通过判断商品是否加在 |
| 1.4 流程逻辑：简要逻辑用流程图表示，包括图3-4-1 心愿清单展示流程图，图3-4-2 添加、移除商品流程图。 |
| 1.5 接口设计： wishlist(request)：用户在account界面点击心愿清单，会通过dashboard模块接口发送GET消息，请求心愿清单网址，通过url路由组件调用该函数。 add\_to\_wishlist(request, id)：用户在单个商品界面点击加入心愿清单，会通过商品展示模块接口发送GET请求，并传递商品id。请求wishlist/add\_to\_wishlist/<int:id>网址，通过url路由组件调用这个函数。 {% url 'account:user\_wishlist' product.id %}：用户在心愿清单的user\_wish\_list.html点击移除心愿清单按钮会触发这个超链接，通过url路由组件调用add\_to\_wishlist(request, id)函数。 数据库接口。 |

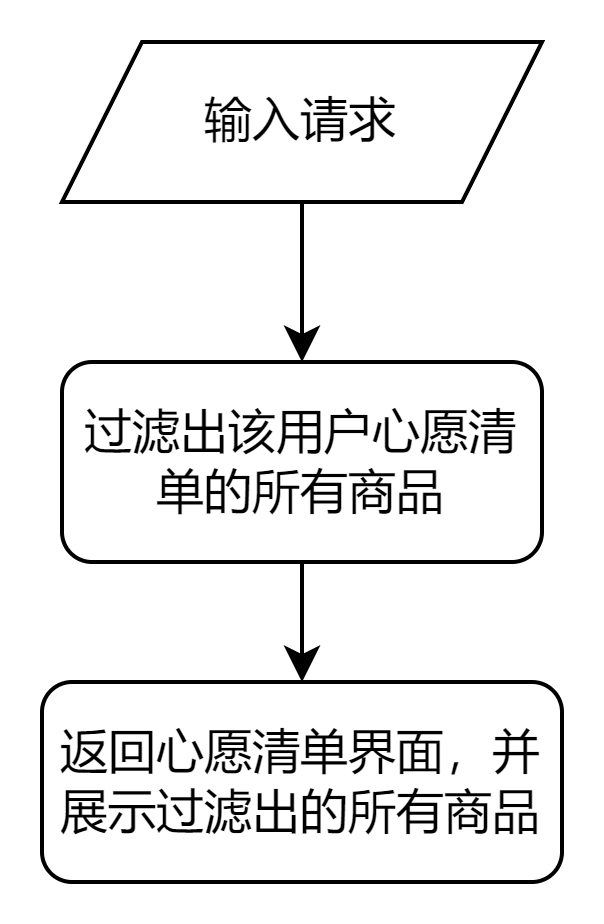


图3-4-1 心愿清单展示流程图

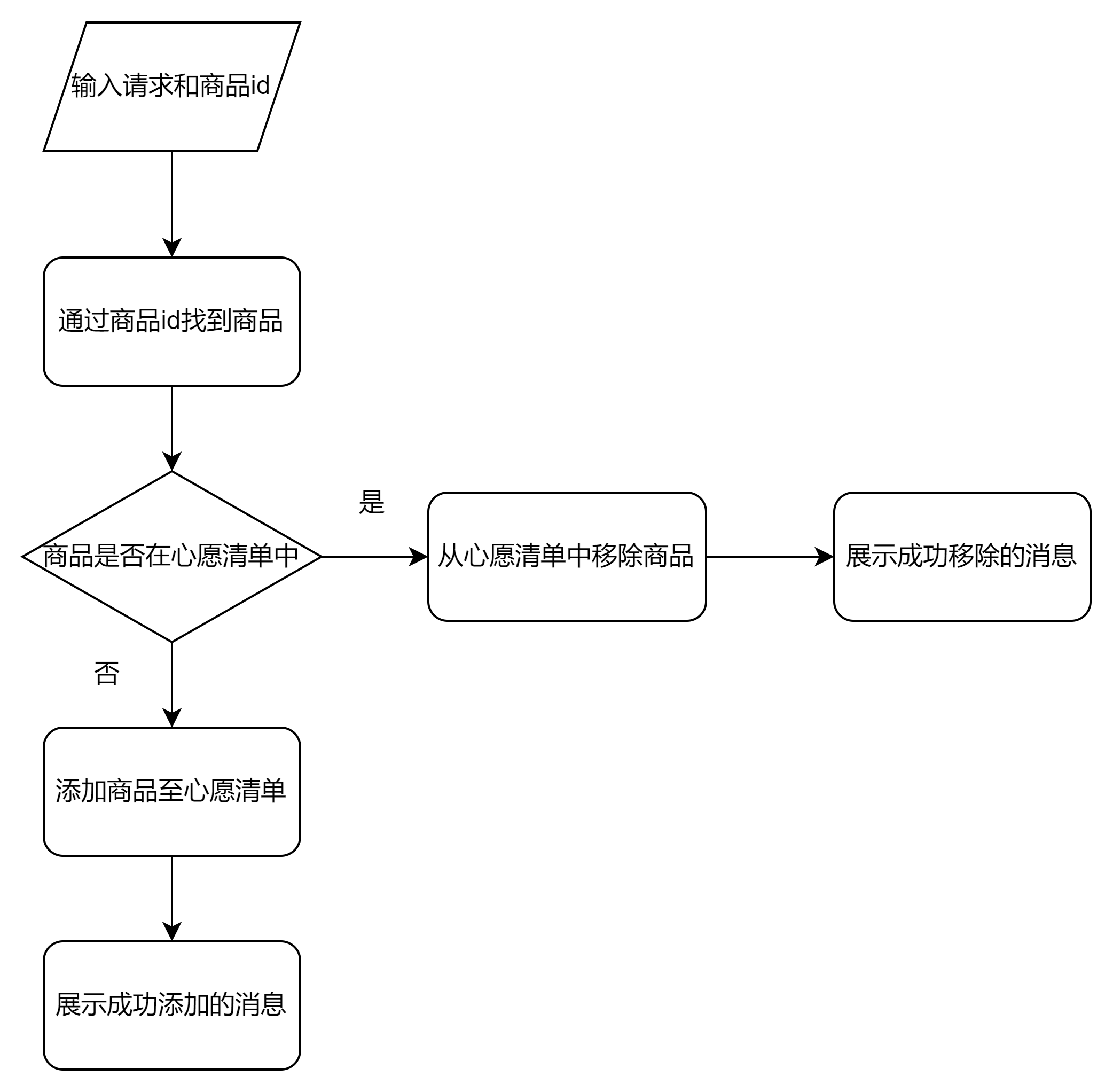


图3-4-2 添加、移除商品流程图

**3.5 订单记录**

该部分只包含订单记录展示一个底层模块，其展开设计与商品展示模块类似，不再赘述，请参考图4。

4.  **商品展示模块详细设计**

该部分分为商品展示、商品详细信息两个底层模块。

其中，商品展示模块的展开设计如下：

|  |
| --- |
| 商品展示模块 |
| 1.1 模块概述：实现商品展示的模块。 |
| 1.2 功能：展示所有商品，分类展示商品，展示名称包含搜索关键字的商品。 |
| 1.3 性能：可以在正常HTTP响应时间内完成。 |
| 1.4 输入项：商品类别，搜索关键字。 |
| 1.5 输出项：展示符合条件的商品。 |
| 1.6 设计模式：根据用户的输入筛选商品的类别或名称，展示符合条件的商品。 |
| 1.7 流程逻辑：实现展示商品的简要逻辑如下图：图4-1.商品展示流程逻辑。 |
| 1.8 接口：连接数据库中商品数据表的接口；商品展示页面是本系统的主页，所以包含以下跳转到其他模块的接口：product\_detail（本模块跳转商品详细信息模块）、account\_dashboard（本模块跳转用户账户模块）、basket\_summary（本模块跳转购物车模块） |

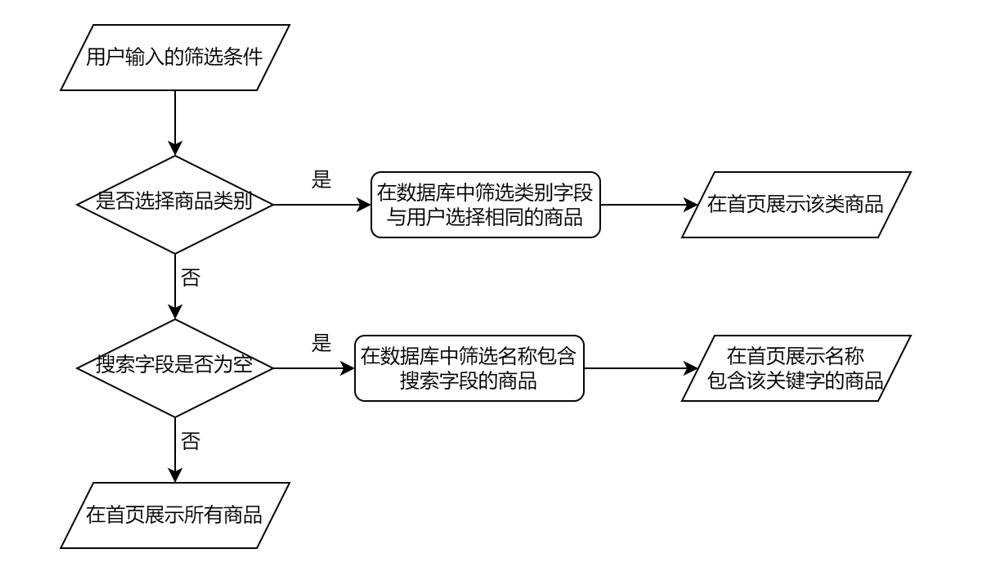


图4-1.商品展示流程逻辑

商品搜索框的前端代码，如图4-2所示：



图4-2 搜索框前端代码

展示所有商品（product\_all）和分类展示商品（category\_list）的函数，如图4-3。

搜索功能在product\_all函数内使用Django中的数据模型对象的filter方法实现，使用名称包含搜索字段条件（title\_\_icontains = request.GET['search\_text']）过滤。



图4-3 展示商品函数代码

**5. 支付模块设计详细设计**

支付模块在用户选择商品加入购物车后调用。

**5.1 运输方式选择模块**

|  |
| --- |
| 运输方式选择模块设计 |
| 1.1 模块概述：用户可以在该模块选择快递运输方式 |
| 1.2 功能：显示所有可选的运输方式 |
| 1.3 性能：可以在正常HTTP响应时间内完成 |
| 1.4 输入项：用户选择运输方式、用户确认 |
| 1.5 输出项：展示可选的运输方式、可操作UI模块 |
| 1.6 设计模式：根据用户在选单中选定的运输方式决定运费，并更改basket模块中的运输方式信息、计算新的总价 |
| 1.7 流程逻辑：当basket\_summary购物车页面确认结账后，跳转到运输方式选择界面。运输方式选择界面显示有更改运输方式的选单和用户确认选择的按钮。之后开始获取用户输入。若用户更改输入方式，则在basket模块中修改运输方式并重新计算总价、更新页面。当用户确认选择时，跳转运输地址选择页面。流程图见图5-1-1 |
| 1.8 其他模块接口：basket\_summary（购物车模块跳转本模块）、delivery\_address（本模块跳转运输地址模块）、basket\_update\_delivery（ajax调用购物车模块更新运输方式） |

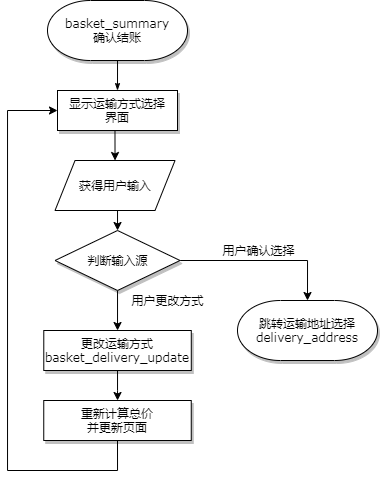


图5-1-1 运输方式选择模块流程图

**5.2 运输地址选择模块**

|  |
| --- |
| 运输地址选择模块设计 |
| 1.1 模块概述：用户可以在该模块选择要寄送的地址 |
| 1.2 功能：显示用户现有的所有地址；若用户没有设置任何地址，则提醒用户无可用地址并给出添加地址的跳转链接 |
| 1.3 性能：可以在正常HTTP响应时间内完成 |
| 1.4 输入项：用户选择默认地址，用户选择修改地址 |
| 1.5 输出项：给出添加地址的跳转链接、显示用户所有可用地址、可操作UI模块 |
| 1.6 设计模式：用户通过点击相应地址的确认按钮来进行地址的选择，选择后将立刻更新默认地址；用户可以修改已有地址；用户点击确认按钮后跳转到结账完成界面 |
| 1.7 流程逻辑：当收到运输方式选择完成的跳转时。1、若用户有地址，根据已有信息显示运输地址选择界面，包括选择默认地址按钮、选择修改地址按钮和用户确认按钮。接下来获得用户输入。若用户更改默认地址，则调用account中set\_default来设置该用户的默认地址；若用户修改地址则跳转account中地址修改模块；若用户确认选择，则转交跳转给。2、若用户无地址，提示用户无地址，并显示前往专用的地址添加模块add\_address\_tmp的链接控件。 |
| 1.8 其他模块接口：delivery\_edit\_address（跳转到account地址编辑界面）、set\_default（调用account设置默认地址）、payment\_complete（确认跳转到结账完成界面）、add\_address\_tmp（account为本模块提供的专用添加地址模块） |

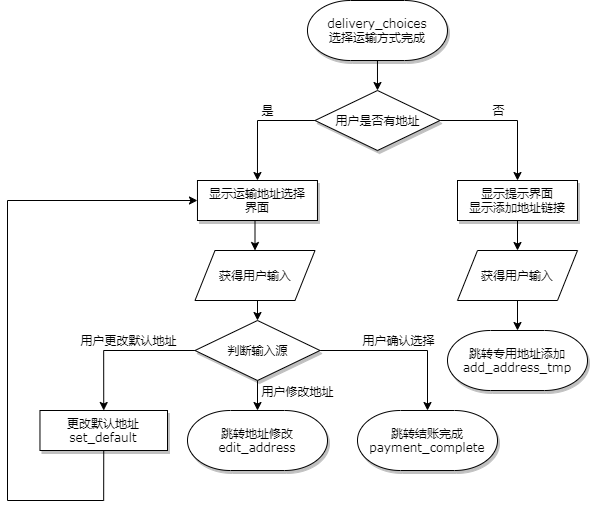


图5-2-1 运输地址选择界面流程图

**5.3 结账完成模块**

|  |
| --- |
| 结账完成模块设计 |
| 1.1 模块概述：生成订单，并提示用户订单已生成 |
| 1.2 功能：根据现有数据生成用户的订单，并且提示用户支付完成的信息并为用户提供查看订单的链接 |
| 1.3 性能：可以在正常HTTP响应时间内完成 |
| 1.4 输入项：用户选择跳转至查看订单 |
| 1.5 输出项：展示提示信息、展示跳转至查看订单的链接 |
| 1.6 设计模式：首先根据现有已收集的数据生成订单，然后提示用户订单已生成，用户通过点击相应地址的链接来选择跳转至查看订单 |
| 1.7 流程逻辑：获得当前用户，显示支付完成的信息，并提示用户可以选择跳转至订单查看界面。若用户选择查看订单，则跳转。伪代码如图5-3-1 |
| 1.8 其他模块接口：delivery\_address（运输地址选择界面调用本模块），user\_orders（调用account中的查看所有订单界面显示所有订单），product（数据库接口），basket模块接口，用户模块接口 |

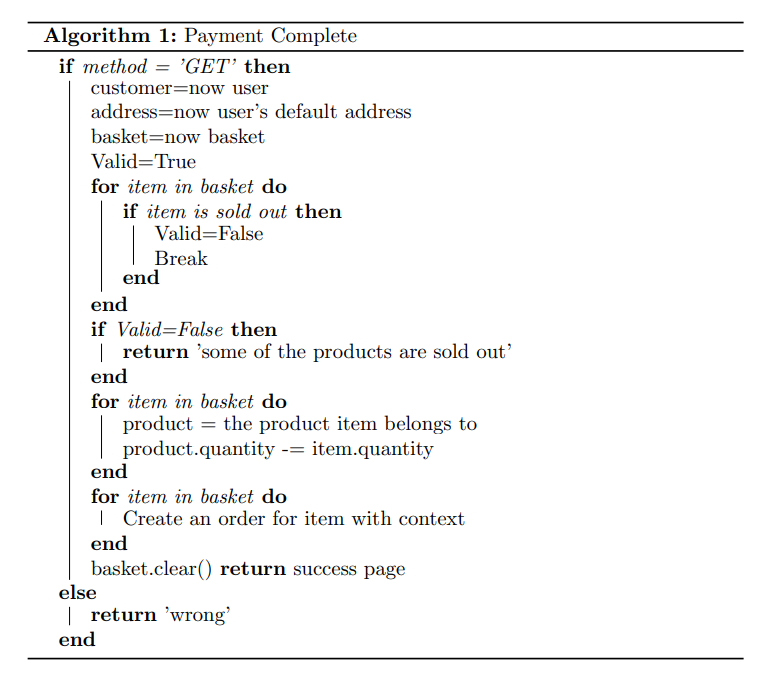


图5-3-1 结账完成模块伪代码

**6．评论模块详细设计**

用户可以使用评论模块对已提交的订单进行评论，它包括3个组件：路由转换组件url、视图组件view、设计界面组件post\_comment.html。评论模块的设计表和设计流程图如下。

|  |  |
| --- | --- |
| |  | | --- | | 评论模块设计 |   1.1模块描述：评论订单 |
| 1.2 功能：用户可以通过此模块，就订单列表中的各个订单发表评论。 |
| 1.3 设计方法：用try except方法设计，防止重复评论问题；使用if else判断评论内容是否有效，以及是否采用GET或POST方法访问。 |
| 1.4 流程逻辑：简要逻辑如图6-1. 评论模块流程逻辑图 |
| 1.5 接口设计： post\_comment(request, order\_id)函数：用户在订单记录界面点击发表评论按钮，会通过订单记录模块的接口发送GET消息，并传入相应的订单号order\_id。请求访问评论订单网址post-comment/<int:order\_id>/，经过路由转换组件url会调用post\_comment函数。 redirect(reverse("account:user\_orders"))语句：点击Submit按钮，将使用POST方法的发表评论，并重定向回到订单列表。 save()：保存评论表单至数据库。 |

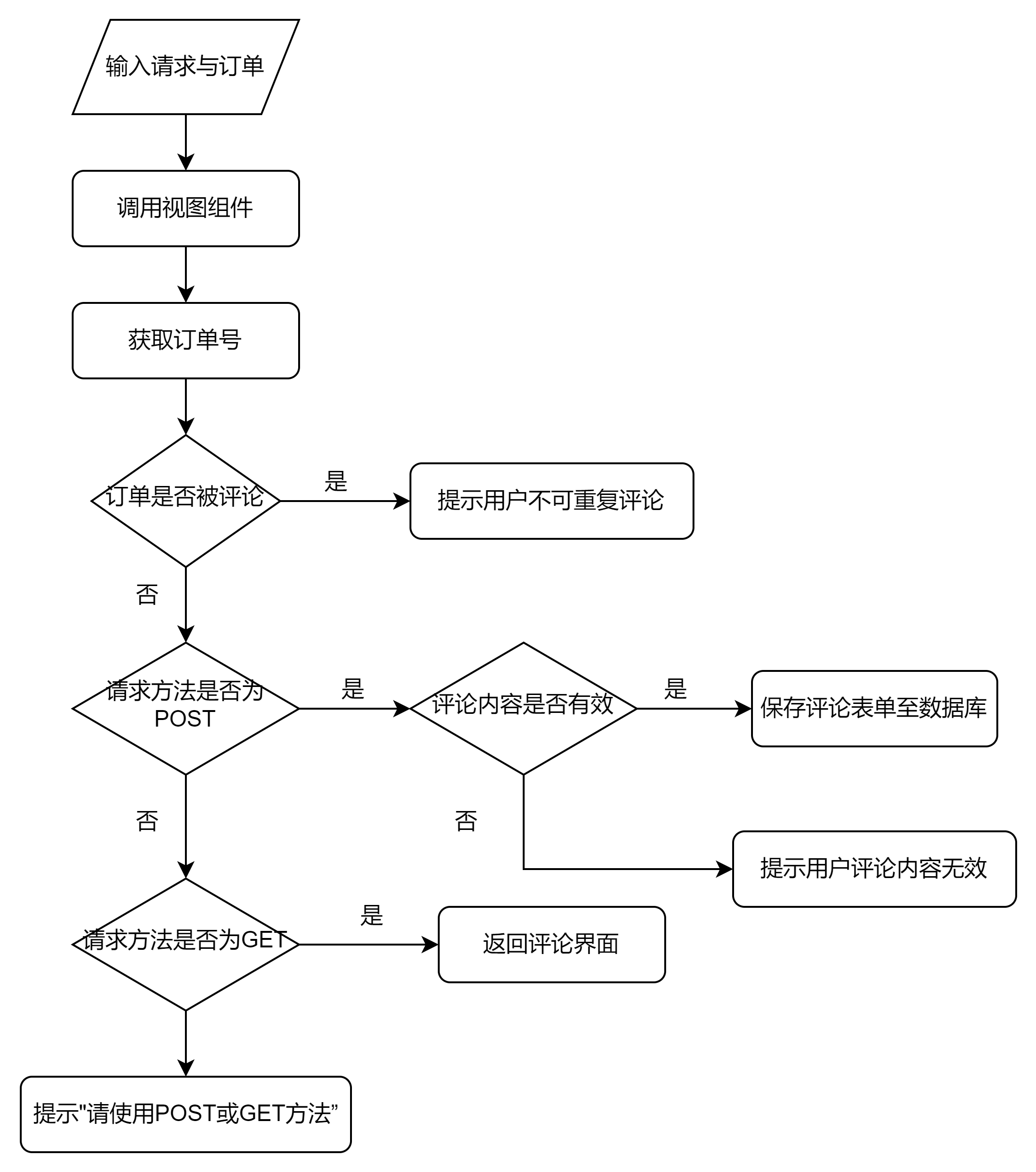


图6-1 评论模块流程图

在接收到请求后调用视图组件view，即调用post\_comment(request, order\_id)函数，使用函数get\_object\_or\_404根据订单号取得相应的订单。然后利用try except方法，首先使用try，尝试获得订单的评论order.comments，若成功返回响应，提示用户不能重复评论。若异常，即订单还没有评论，进入except。

在except中，首先通过if语句判断请求方法是否为请求是否为POST方法，若为POST方法，利用CommentForm函数获取评论表单。再检查表单内容是否有效，同样采用if else方法。若有效保存表单至数据库，否则返回HTTP响应，提示用户表单内容有误。若为GET方法，调用函数render(request, 'comment/post\_comment.html')，返回post\_comment.html界面。若不是POST也不是GET方法，返回消息提示用户请使用POST或GET方法。

**7．购物车模块设计详细设计**

购物车在未登录时仍可使用。

|  |
| --- |
| Basket模块设计 |
| 1.1 模块描述：购物车模块。 |
| 1.2 功能：实现购物车的常见功能，查看购物车的商品总结并可以点击结算以进行下一步操作。 |
| 1.3 性能：能够正确并流畅地运行。 |
| 1.4 输入项：在商品详情页面点击“加入购物车”的商品信息 |
| 1.5 输出项：渲染basket/summary.html 模板，展示出目前添加到购物车的所有商品信息与总价。 |
| 1.6 设计方法（算法）：使用面向对象方法，在basket.py中实现Basket类。Basket类初始化如图所示，使用session。Basket中有成员函数：add()，update()，get\_price()，delete()，clear()，save()。 |
| 1.7 流程逻辑：在某一商品详情页面点击“加入购物车”后，ajax访问{% url "basket:basket\_add" %}链接，经过basket/urls.py分发后，调用basket/views.py的add\_basket函数，如图7-2所示。在basket/views.py的add\_basket函数中，调用Basket类的成员函数add()，这一流程逻辑在图7-3中展示。另一主要流程逻辑是：点击购物车，进入购物车界面，返回渲染好的templates/basket/summary.html模板。 |
| 1.8 接口：在 Basket类中，如图7-1所示，向结账模块提供basket\_update\_delivery()接口。 |

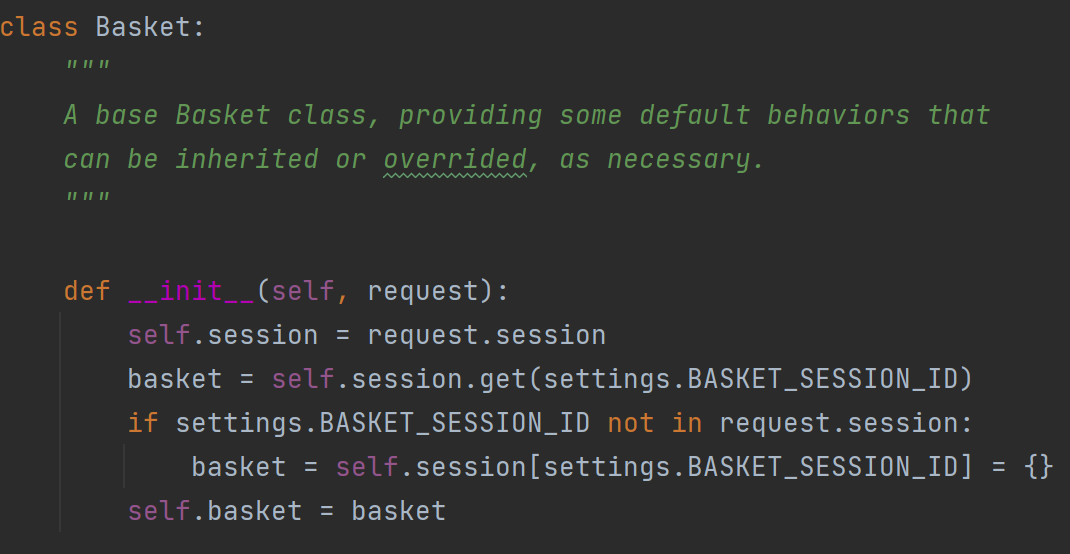


图7-1 Basket类

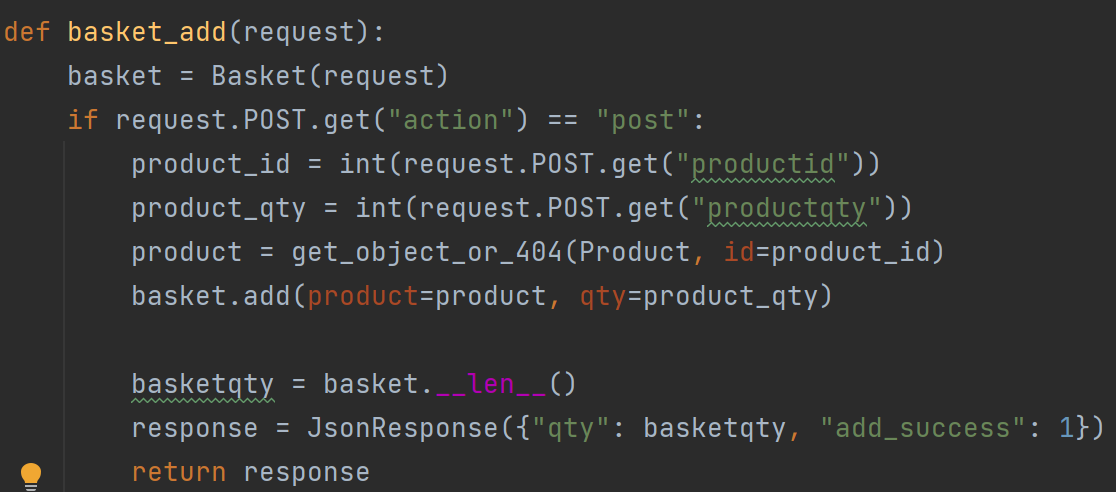


图7-2 basket\_add函数

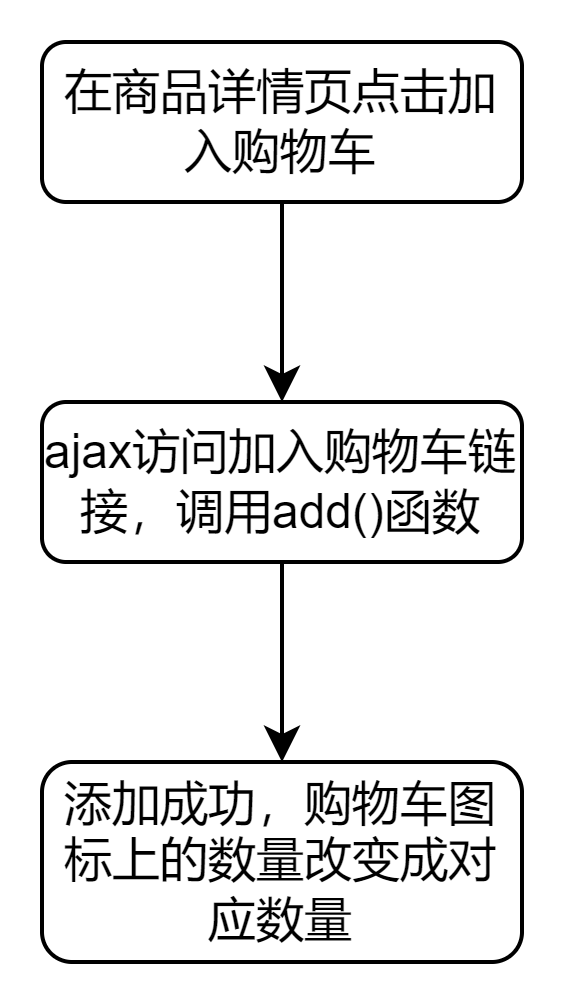


图7-3添加商品流程图

**8. 管理模块**

管理模块包含用户管理和商品管理两个子模块，这两个模块的详细设计与用户账户模块下的账户管理模块和商品管理模块相似，请参考3.2和3.3节。