**软件工程课程设计**

**——结构化设计**

**题 目:校园二手物品交易平台**

**班 级：20计科2班**

**2022 年 4 月 12 日**

目录

[一. 引言 3](#_Toc70282438)

[二. 体系结构设计 4](#_Toc70282439)

[2.1.系统功能模块划分 4](#_Toc70282440)

[2.1.1用户信息管理系统功能描述 4](#_Toc70282441)

2.1.2商品管理系统功能描述……………………………………………………………………………………..5

2.2系统架构图……………………………………………………………………………………………………………………8

[2.3精化后的数据流图 9](#_Toc70282443)

2.4系统结构图…………………………………………………………………………………………………………………….9

[2.4.1 个人信息管理系统的系统结构图 10](#_Toc70282444)

[2.4.2 信息查看系统的系统结构图 10](#_Toc70282445)

[三. 接口设计 12](#_Toc70282451)

[3.1人机交互接口设计 12](#_Toc70282452)

[①登陆界面设计 13](#_Toc70282453)

[②找回密码界面设计 13](#_Toc70282454)

[③注册界面设计 13](#_Toc70282455)

[④ 首页界面设计 14](#_Toc70282456)

[3.2 模块间接口设计 1](#_Toc70282459)5

[3.2.1查看和修改信息个人信息管理系统的接口 1](#_Toc70282460)5

[3.2.2登录页面与主界面之间的接口 1](#_Toc70282461)6

[3.2.3删除修改个人发布商品界面的接口 1](#_Toc70282462)6

[四. 数据设计 1](#_Toc70282466)7

[4.1 调整后的实体-关系图 1](#_Toc70282467)7

[4.2 关系模式的确立 1](#_Toc70282468)8

[4.3 数据库中表的实现 1](#_Toc70282469)8

[4.4 建表语句(MySQL) 2](#_Toc70282470)1

[五．过程设计 2](#_Toc70282471)4

[5.1 主要功能模块程序流程图 2](#_Toc70282472)4

[5.2 重要算法的伪代码描述 3](#_Toc70282473)3

# 一. 引言

“面向结构”的程序设计方法即[**结构化程序设计方法**](https://baike.baidu.com/item/%E7%BB%93%E6%9E%84%E5%8C%96%E7%A8%8B%E5%BA%8F%E8%AE%BE%E8%AE%A1%E6%96%B9%E6%B3%95/4707472)， 结构上将软件系统划分为若干功能模块， 各模块按要求单独编程， 再由各模块连接， 组合构成相应的软件系统。 该方法强调程序的结构性， 所以容易做到易读，易懂。该方法思路清晰，做法规范，深受设计者青睐。结构化程序设计思想使程序执行效率提高 ，程序的出错率和维护费用大大减少。

结构程序设计就是一种进行程序设计的原则和方法，按照这种原则和方法可设计出**结构清晰、容易理解、容易修改、容易验证**的程序结构化程序设计的目标在于使程序具有一个合理结构，以保证和验证程序的正确性，从而开发出正确、合理的程序。

由于[模块](https://baike.baidu.com/item/%E6%A8%A1%E5%9D%97)相互独立，因此在设计其中一个模块时，不会受到其它模块的牵连，因而可将原来较为复杂的问题化简为一系列简单模块的设计。模块的独立性还为扩充已有的系统、建立新系统带来了不少的方便，因为我们可以充分利用现有的模块作积木式的扩展。

结构化程序设计采用自顶向下、逐步求精的设计方法，各个模块通过“**顺序、选择、循环**”的控制结构进行连接，并且只有一个入口、一个出口。

# 二. 体系结构设计

## 2.1.系统功能模块划分

### 2.1.1个人信息管理系统功能描述

#### ① 注册和验证功能

用户在使用本软件时应该首先进行注册，学生应首先进行身份验证，经对应企业端验证回复后方可完善基本信息，并在邮箱验证后分配id并将基本信息和登陆信息录入到信息表中，企业方在后台通过管理员进行注册。

#### ② 登陆和找回密码功能

用户可以通过id或注册邮箱和密码进行登陆活动，如果密码忘记或者欲改变密码可以通过注册时保存的邮箱进行接受验证码信息，或者通过密保问题进行验证个人信息。通验证身份后，输入新的密码，系统将新密码进行替换。

#### ③ 用户信息更改功能

用户可以在系统内进行信息修改操作，修改信息之后将修改候的信息保存至信息表并在个人信息界面显示修改后的数据。

#### ④用户建议功能

用户可以通过用户提出建议功能将个人的建议反馈给管理员，管理员可以通过该功能将反馈后的答复反馈给该用户。并将建议加入输出建议表

### 2.1.2商品管理系统功能需求

#### ①浏览商品界面内容

用户登录之后，进入商品界面，浏览商品，评论内容，系统会从数据库中查找出相关内容，并将信息呈现到用户界面上

#### ②发布商品

用户提交商品信息，管理员在进行内容审核之后将合格的商品信息发布，若商品的内容不合格，则将修改信息发送给用户，用户根据内容修改之后再次提交商品发布信息，重新进行审核

#### ③举报内容

用户选中想举报的内容，点击举报，选择举报类别，输入举报理由，举报信息将发送到管理员处。管理员将在数据库中搜索被举报的内容，进行审核。

#### ④发布评论内容

用户在查看商品时，或者购买商品之后，可以选择发布评论，填写评论内容，并发送到相关帖子下，数据会存入数据库中。

#### ⑤举报评论

#### 用户可以在看到不当的评论之后将该评论信息举报，经由管理员核实之后，若发现该评论存在不当的事实，则会删除该评论，并给予该评论发布者警告，多次被警告的用户将会被给予禁言或者封号的惩罚措施。

#### ⑥举报商品

用户购买商品或者收到商品之后，若发现商品存在欺诈现象或者商品页存在不良情况，可向管理员提出举报，经管理员核实之后，若情况属实，则会撤销该商品并且对商品发布者进行警告，多次违反的用户将会被给予封号或者注销账号的惩罚。

#### ⑦买家与卖家交流

用户在查看商品时可以通过该平台与卖家进行实时交流沟通，交流的信息会存储在数据库中

#### ⑧加入购物车功能

用户可以在发现自己喜欢的商品之后先将商品加入购物车。用户可以在购物车内进行付款。对于加入购物车的商品，商品信息会加入购物车表。

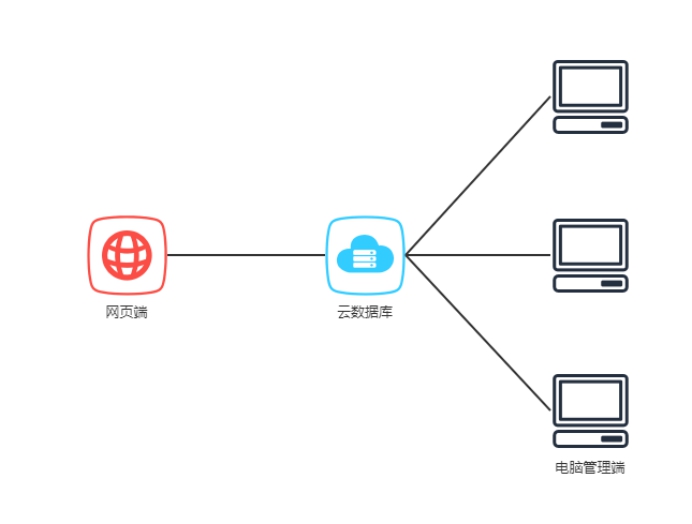
#### ⑨商品交易功能

用户在发现自己喜欢的商品之后，可以选择购买该商品。如果交易成功，则会在交易后留下交易记录并记录在数据库当中，用户可以查看过往的交易记录

#### ⑩商品搜索功能

用户可以在商品展示页的搜索框中通过关键字对自己需求的商品进行搜索，也可以通过在商品展示页的分栏里进行初步的查看。同时，搜索的历史记录会保留在数据库内。

## 2.2系统架构图



## 2.3 精化后的数据流图

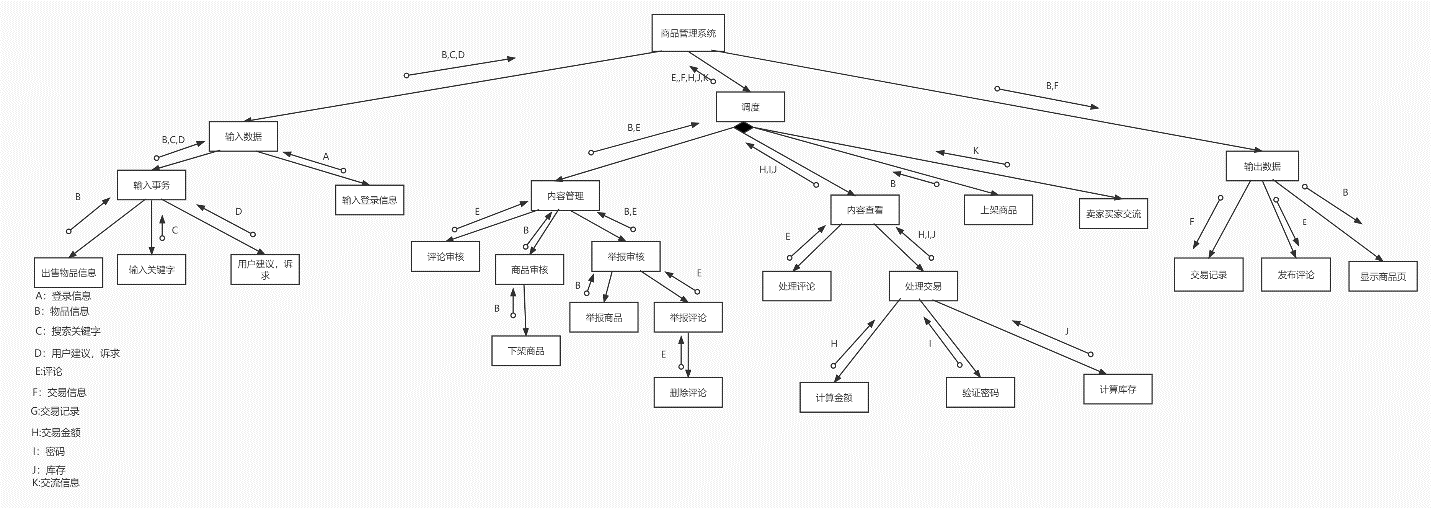
## C:\Users\lenovo\Desktop\数据流图改.jpg

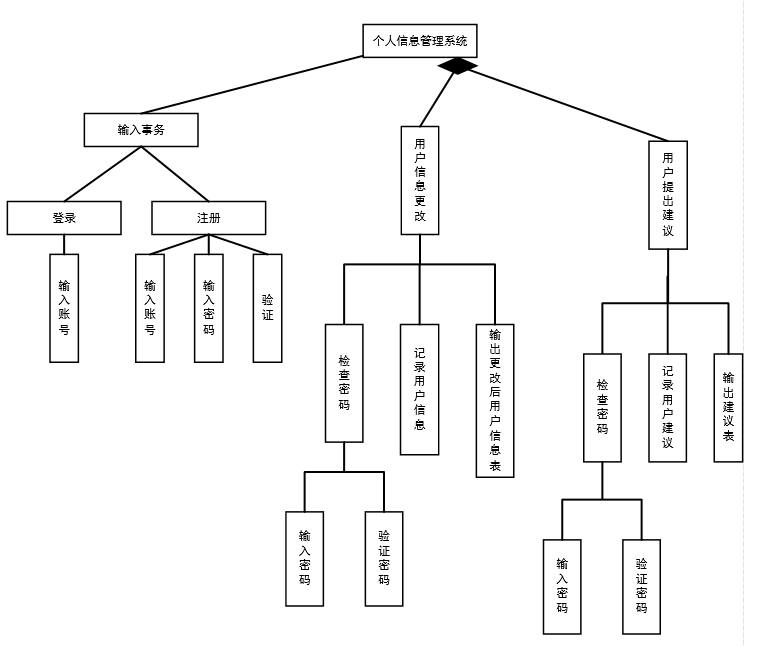
## 2.4系统结构图

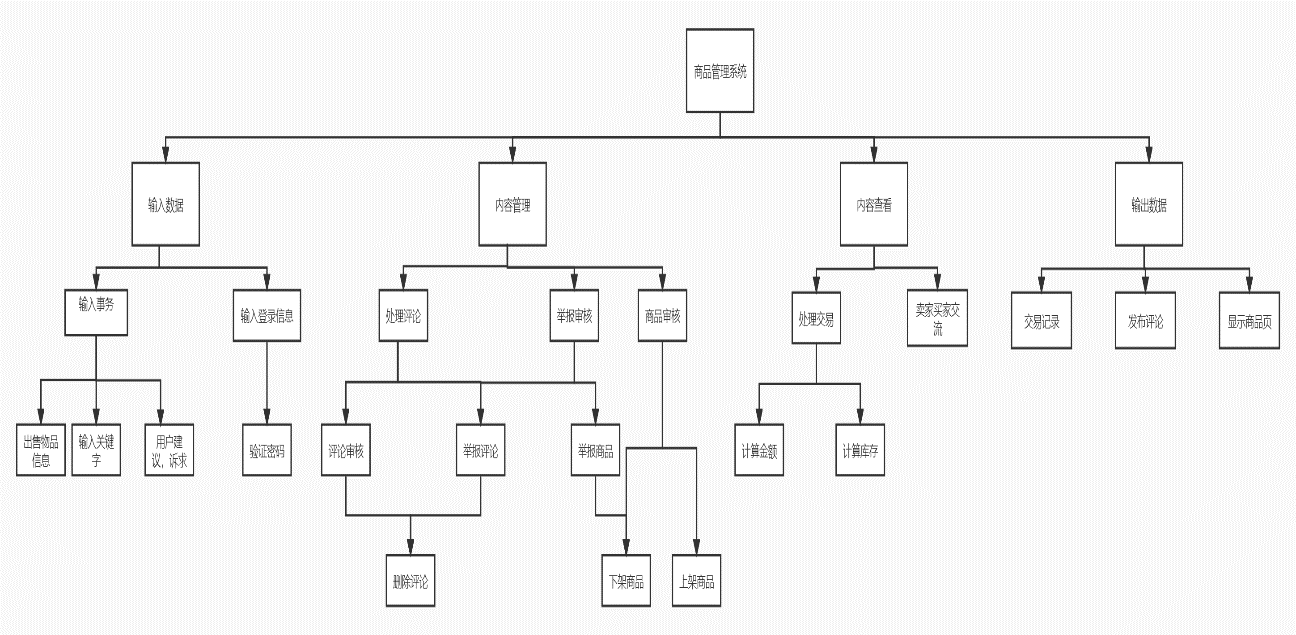
### 2.4.1 个人信息管理系统的系统结构图

### C:\Users\lenovo\Documents\Tencent Files\153642895\FileRecv\jg1.png

### 2.4.2 商品管理系统的系统结构图



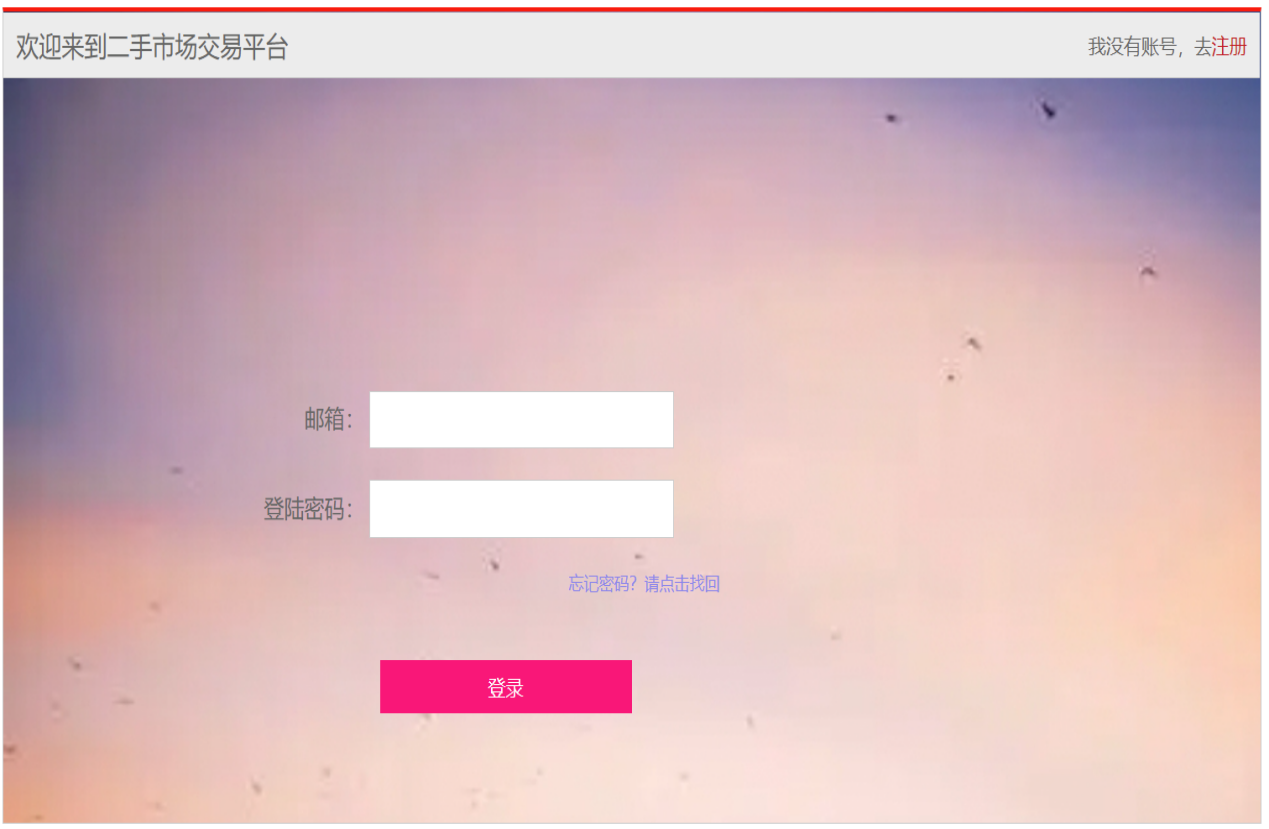




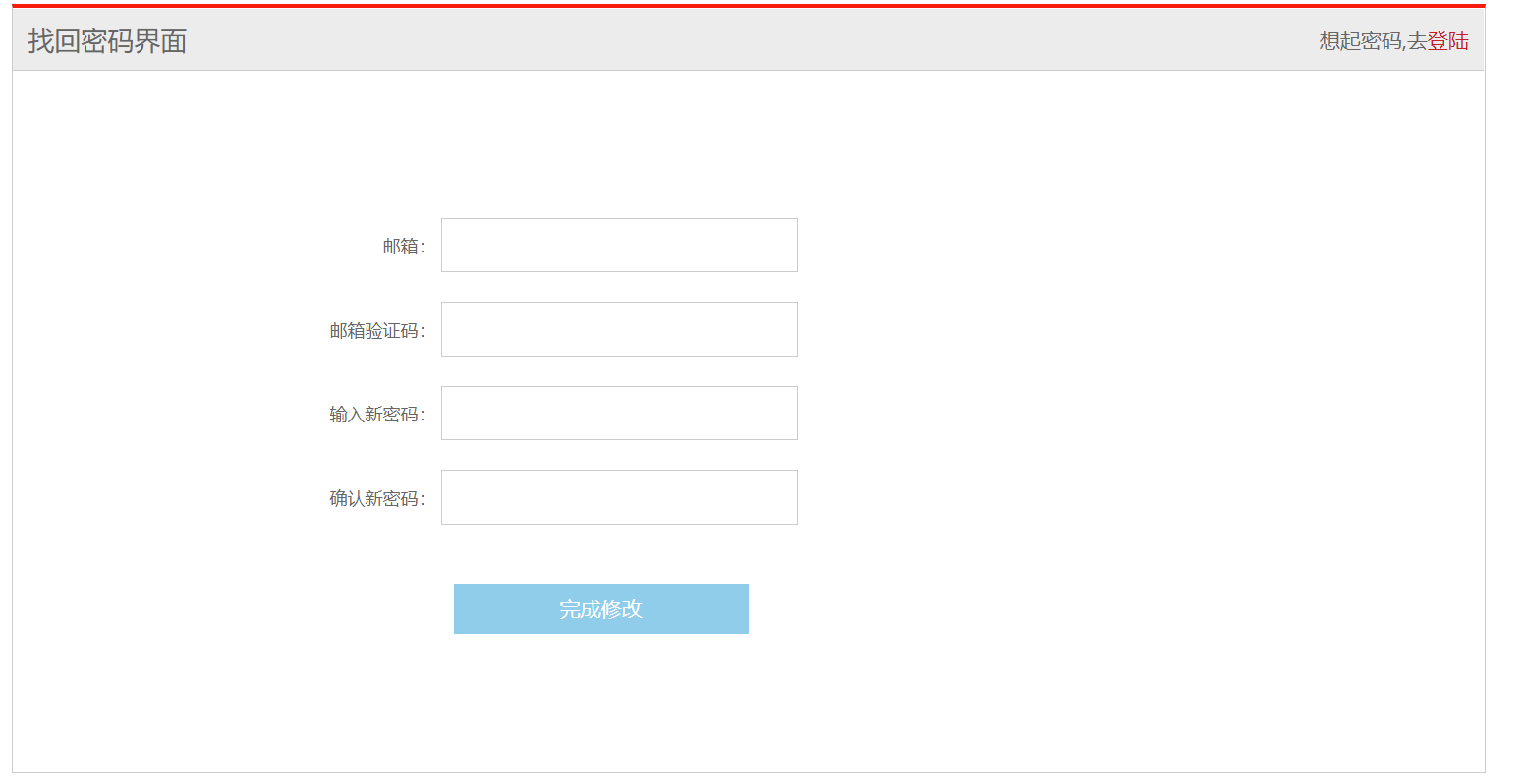
# 三. 接口设计

## 3.1人机交互接口设计

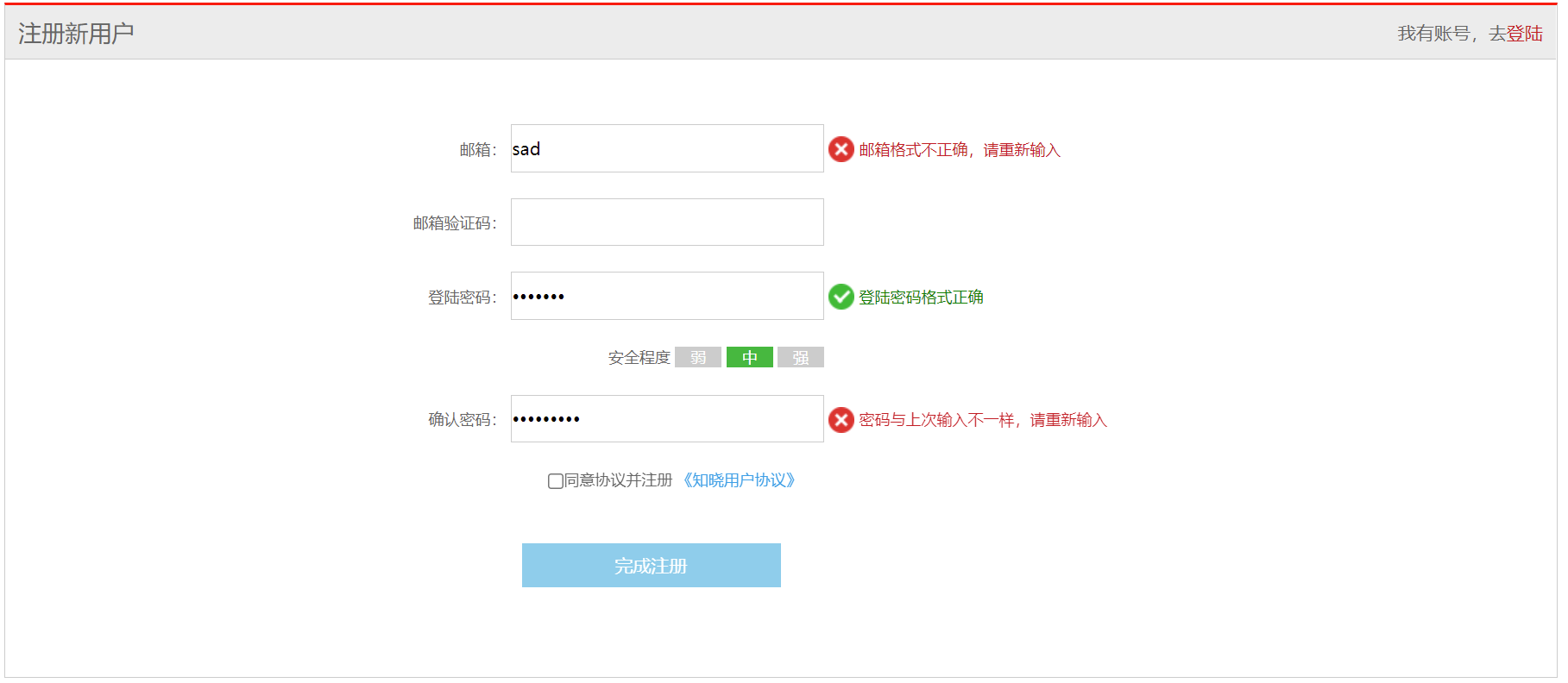
### ①登录界面设计



### ②找回密码界面设计



### ③注册界面设计



### ④ 首页界面设计



## 3.2 模块间接口设计

### 3.2.1查看和修改个人信息管理系统的接口

描述：在进入查看和修改信息模块时，系统需要根据操作对象的身份信息进行显示和表查询

实现：构造方法的参数传递

例： def ReadAndExitinfo(request,ID):

Self.ID=ID;

}

### 3.2.2登陆验证与主界面之间的接口

描述：由于每种类型用户登陆后进入的主界面右上角显示信息不同，需要在主界面启动时传递管理员信息。

实现：方法的参数传递

例：用户的登陆显示图像

def mainwindow(request,ID):

photo\_owner = request.POST['ID']

result=Users.objects.filter(ID=photo\_owner)

if result:

return render(request,'demo/demo.html', {'result'

:1})

3.2.3 删除/修改个人发布物品之间的接口

描述：在我们删除/修改特定的商品时，需要查询或点击一个特定的商品。此处需要接口传递商品的编号。

实现：

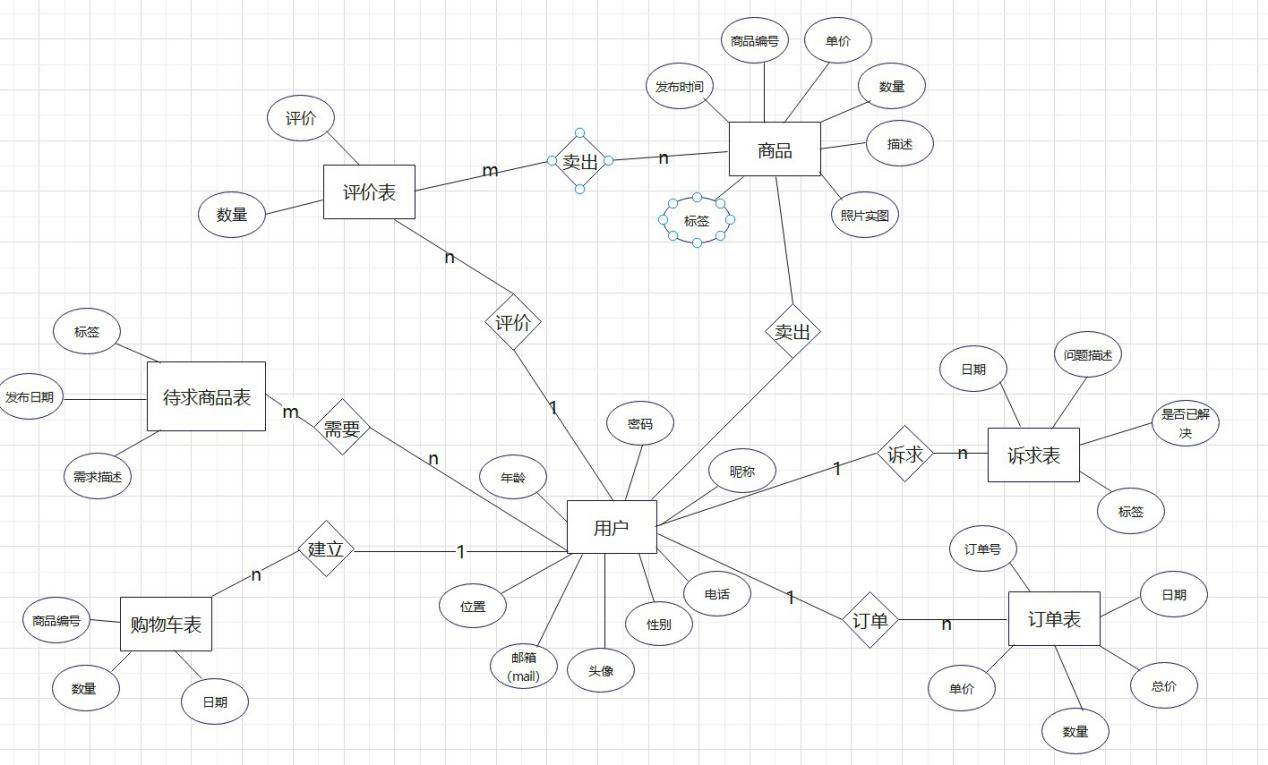
def DeleteAndEdit(request,ID):

Self.ID=ID;

}

# 四. 数据设计

## 4.1 调整后的实体-关系图



## 4.2 关系模式的确立

用户表（ID，email，telephone，密码，昵称，位置，性别，年龄，姓名）

商品标签表（num, tag）

商品表（ID，商品ID，单价，数量，发布日期，商品描述，商品照片，tag）

待求商品表（需求编号，ID，tag，物品描述，发布日期）

购物车表（ID，商品编号，单价，数量，加入日期）

评价表（cusID，ID，商品ID，评价日期，评价内容）

订单表（cusID，ID，成交日期，商品ID，数量，单价，总价）

建议表（ID，日期，建议描述，是否解决）

## 4.3 数据库中表的实现

用户表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 属性名 | 说明 | 字符类型 | 长度 | 约束 |
| ID | 系统自动为用户生成的ID | Varchar | 8 | 主键 |
| Email | 绑定用户的email，可以用于找回密码， | varchar | 30 | 唯一 |
| telephone | 用户的电话 | Varchar | 11 | 唯一 |
| 密码 | 用户用于登陆的密码 | varchar | 20 | 长度>6并且长度<20 |
| 昵称 | 用户自己设置的昵称 | nvarchar | 30 |  |
| 位置 | 卖家买家所在的位置 | nVarchar | 30 |  |
| 性别 | 用户的性别 | Char | 2 | ‘男’，‘女’ |
| 年龄 | 用户的年龄 | int | 1 | >0 and <100 |
| 姓名 | 用户的姓名 | nvarchar | 8 |  |

标签表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 属性名 | 说明 | 字符类型 | 长度 | 约束 |
| num | 标签所属于的编号 | int |  | >0 |
| tag | 商品的标签种类 | varchar | 20 | 主键 |

商品表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 属性名 | 说明 | 字符类型 | 长度 | 约束 |
| ID | 商品所属的卖家 | varchar | 8 |  |
| 商品ID | 商品的编号 | varchar | 8 | 主键 |
| 单价 | 商品的单价 | float |  | >0 |
| 数量 | 商品的数量 | int |  | >0 |
| 发布日期 | 商品发布的日期 | datetime |  |  |
| 商品描述 | 卖家有关要卖的物品的描述 | nvarhar | 255 |  |
| 商品照片 | 卖家对要卖的商品的图片在某一文件中的相对路径的记录 | Nvarchar | 40 |  |
| tag | 商品所属的种类 | Nvarchar | 10 |  |

待求商品表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 属性名 | 说明 | 字符类型 | 长度 | 约束 |
| 需求编号 | 需求的编号 | varchar | 4 | 主键 |
| ID | 发布需求的用户ID | Varchar | 8 |  |
| Tag | 发布需求的商品所属的种类 | nvarchar | 30 |  |
| 物品描述 | 发布需求的商品的相关描述 | nvarchar | 100 |  |
| 发布日期 | 发布需求的当天日期 | Datetime | 20 | Not null |

购物车表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 属性名 | 说明 | 字符类型 | 长度 | 约束 |
| Id | 购物车所属的主人 | nvarchar | 8 | 主键 |
| 商品编号 | 表示商品的数字字符串 | Varchar | 8 | 主键 |
| 单价 | 商品的单价 | float |  | >0 |
| 数量 | 购物车中商品的数量 | int |  | >0 |
| 加入日期 | 买家把商品加入购物车的日期 | datetime |  | Not null |

评价表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 属性名 | 说明 | 字符类型 | 长度 | 约束 |
| cusId | 买家ID | varchar | 8 | 主键 |
| ID | 卖家ID | Varchar | 8 |  |
| 商品ID | 购买的物品的ID | Varchar | 8 | 主键 |
| 评价 | 用户对购买的物品和卖家的评价 | nvarchar | 100 |  |
| 评价日期 | 用户对购买的物品和卖家的评价的日期 | datetime |  | Not null |

订单表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 属性名 | 说明 | 字符类型 | 长度 | 约束 |
| cusID | 买家id | varchar | 8 | 主键 |
| ID | 卖家 | varchar | 8 |  |
| 成交日期 | 交易完成的日期 | datetime |  | Not null |
| 商品ID | 商品的编号 | varchar | 8 | 主键 |
| 数量 | 购买的商品的数量 | Int |  | >0 |
| 单价 | 商品的单价 | float |  | >0 |
| 总价 | 商品的总价 | float |  | >0 |

建议表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 属性名 | 说明 | 字符类型 | 长度 | 约束 |
| Id | 提出建议用户的id | varchar | 8 | 主键 |
| 日期 | 提交建议的日期 | datetime |  | Not null |
| 建议描述 | 用户提出的具体建议 | nvarchar | 40 | not null |
| 是否解决 | 问题是否已经解决 | boolean |  | True false |

## 4.4 建表语句(SQL server)

##标签表

create table 标签

(

num int ,

tag nvarchar(20),

primary key (tag)

)

##用户表

create table 用户(

ID varchar(8),

Email varchar (30) unique,

Telephone varchar(11) unique,

密码 varchar(20) check(len(密码)>6 and len(密码)<20),

昵称 nvarchar(30) ,

位置 nvarchar(30),

性别 char(2) check(性别 in ('男','女')),

年龄 int check(年龄 >0 and 年龄 <150),

姓名 nvarchar(8) ,

PRIMARY KEY(ID)

);

##商品表

create table 商品表

(

ID varchar(8),

商品ID varchar(8) ,

单价 float check (单价>0),

数量 int check(数量>0),

发布日期 datetime default (getdate()),

商品描述 nvarchar(255) ,

商品照片 nvarchar(40),

tag nvarchar(20),

primary key(商品ID),

foreign key (ID) references 用户(ID),

foreign key (tag) references 标签(tag),

)

##待求商品表

create table 待求商品

(

需求编号 int primary key ,

ID varchar(8),

tag nvarchar(20),

物品描述 nvarchar(100),

发布日期 datetime default(getdate()),

foreign key (ID) references 用户(ID),

foreign key (tag) references 标签(tag),

)

##购物车表

create table 购物车

(

ID VARCHAR(8),

商品ID varchar(8) ,

单价 float check (单价>0),

数量 int check(数量>0),

加入日期 datetime default (getdate()),

primary key(ID,商品ID),

foreign key (ID) references 用户(ID),

foreign key (商品ID) references 商品表(商品ID),

)

##评价表

create table 评价

(

cusID varchar(8),

ID VARCHAR(8),

商品ID varchar(8) ,

评价 nvarchar(100) ,

评价日期 datetime default (getdate()),

primary key(cusID,ID),

foreign key (ID) references 用户(ID),

foreign key (cusID) references 用户(ID),

foreign key (商品ID) references 商品表(商品ID),

)

##订单表

create table 订单表

(

cusID varchar(8),

ID VARCHAR(8),

单价 float check (单价>0),

数量 int check(数量>0),

总价 float check (总价>0),

商品ID varchar(8) ,

成交日期 datetime default (getdate()),

primary key( cusID,商品ID),

foreign key (ID) references 用户(ID),

foreign key (cusID) references 用户(ID),

foreign key (商品ID) references 商品表(商品ID),

)

##建议表

create table 建议

(

ID VARCHAR(8),

建议日期 datetime default (getdate()),

建议描述 nvarchar(40) ,

是否解决 bit,

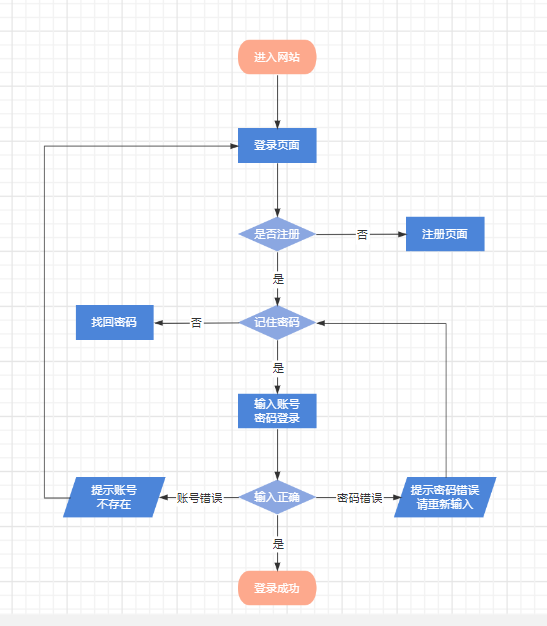
foreign key (ID) references 用户(ID),

)

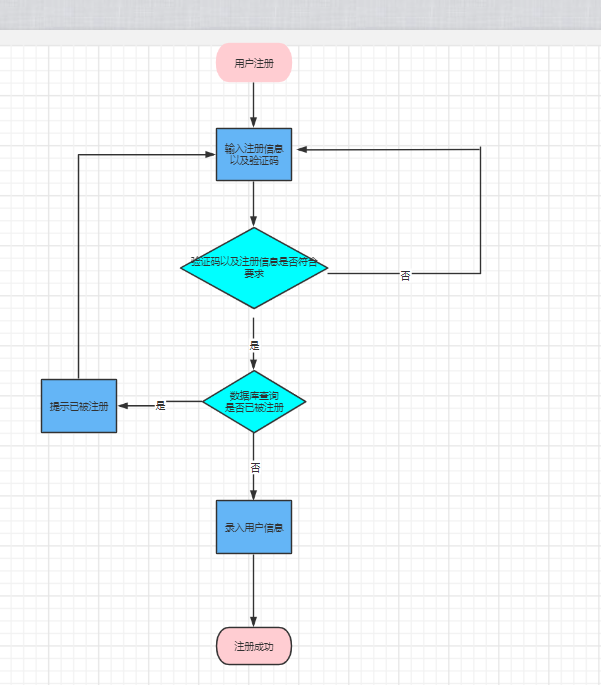
# 五．过程设计

## 5.1 主要功能模块程序流程图

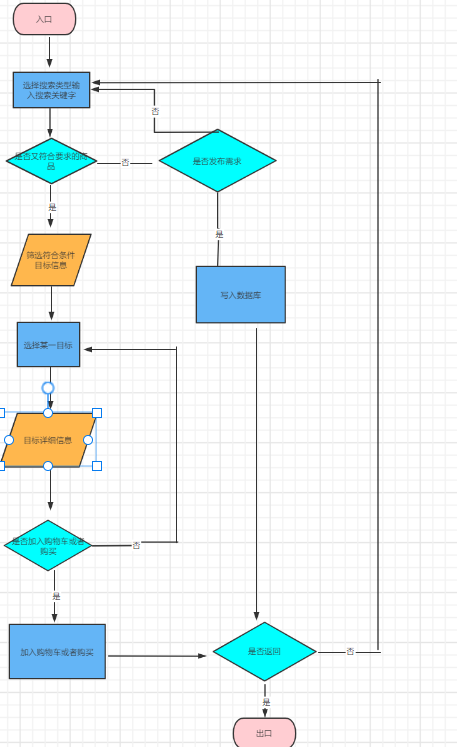
①用户登录模块



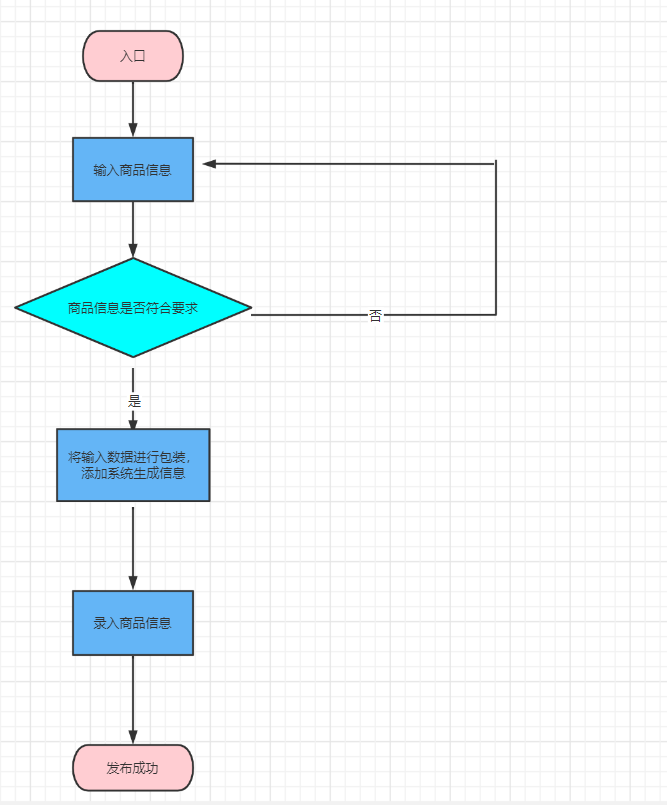
②用户注册模块



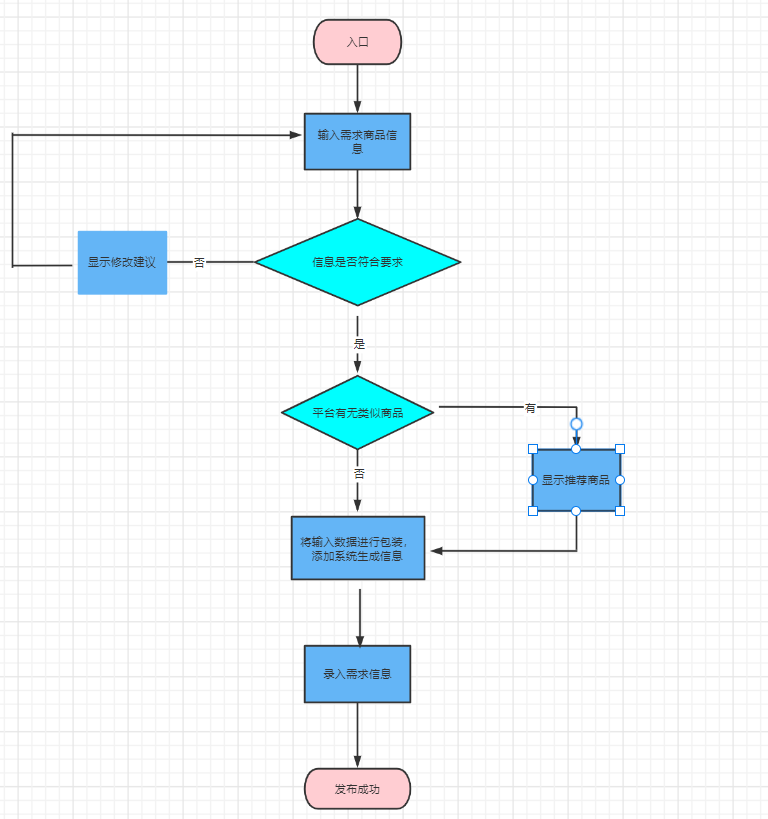
③搜索查看信息模块



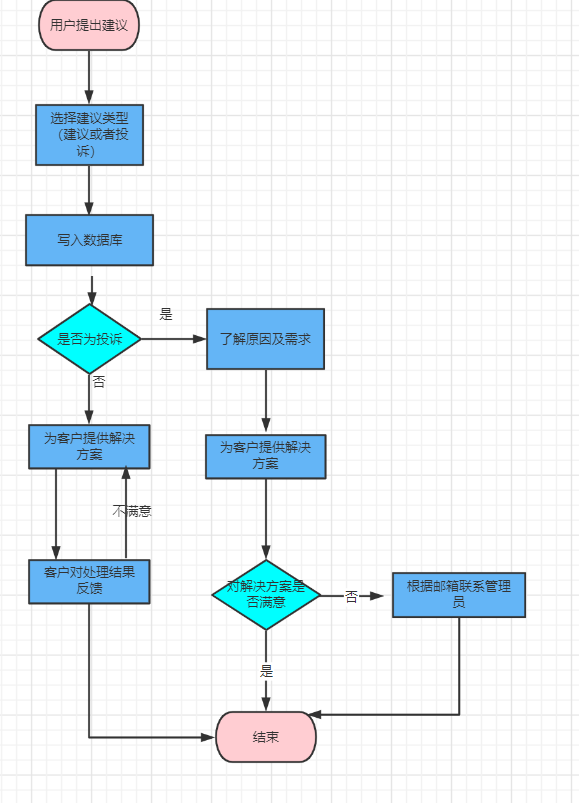
④商品发布模块



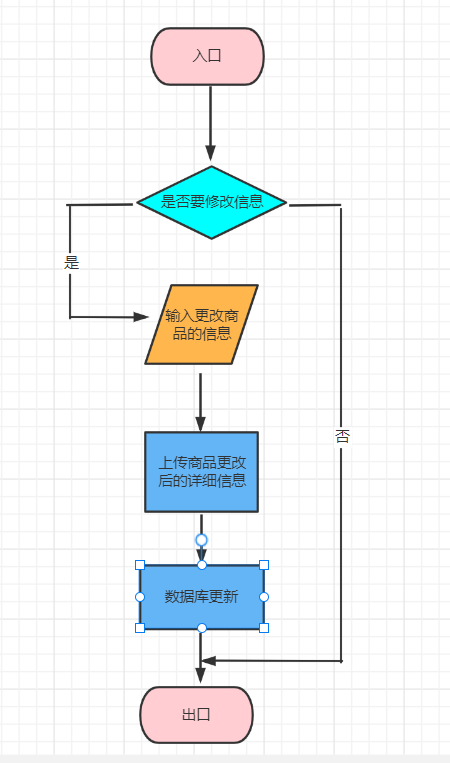
⑤需求发布模块



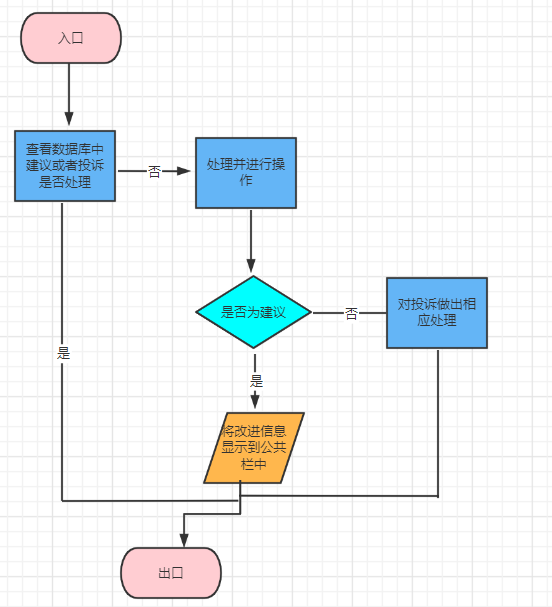
⑥用户建议模块



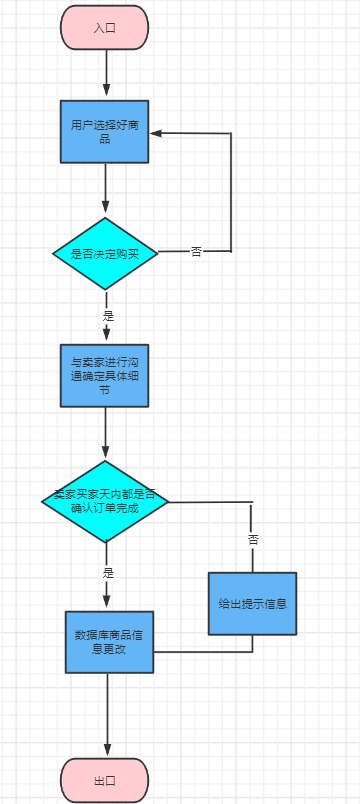
⑦管理商品信息模块



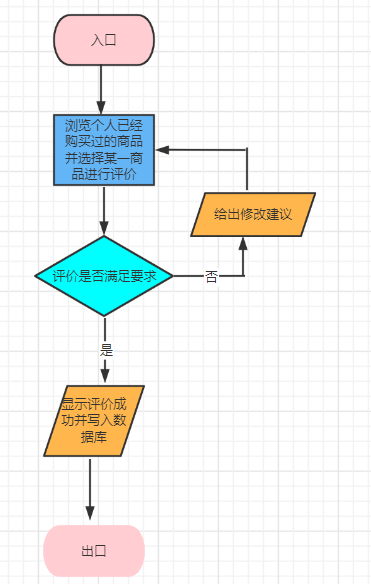
⑧建议投诉处理模块



⑨商品交易模块：



⑩评价过程模块



## 5.2 重要算法的伪代码描述

重要算法

给用户发送注册和找回密码验证码和生成的8位ID：

（为描述清楚，给出部分前端以及后端代码）：

HTML：创建一个表单实例：

login.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>test</title>

</head>

<body>

<form method="post" action="{% url 'check' %}">

<input type="text" name="name" placeholder="your username"><br>

<input type="password" name="pwd" placeholder="your password"><br>

<input type="submit" value="提交"><br>

</form>

</body>

</html>

后端python算法的伪代码实现：

import yagmail

import pandas as pd

import random

import string

import time ##先引入python必要的包

class send\_mail:

## 这个类实现自动发送邮件的功能,下面会用到

#目前适用于163邮箱

def init\_\_(self,user='\*\*\*\*\*\*\*\*@163.com',passwd='\*\*\*\*\*\*\*\*\*'):

##这里到时候会填充我的163邮箱账号密码·

self.yag = None

self.mail\_user = user

#密码(部分邮箱为授权码)

self.mail\_pass = passwd

#初始化用户名字和密码

self.yag = yagmail.SMTP(user=self.mail\_user, password=self.mail\_pass, host='smtp.163.com')

def messsage(self,body,file=None,who='2948561911@qq.com',title="验证码"):

subject = title

body = '你的验证码是'+body+',两分钟内有效，请及时查收并且不要泄露给任何人，感谢你的注册。'

self.yag.send(to=who,subject=subject, contents=[body])

def reg(request):

if request.method == 'POST':

user\_email = request.POST.get('email')

##从前端得到必要的数据

time\_start = time.time()

yzm = ''.join(random.sample(string.digits, 6))

email = send\_mail()

email.messsage(yam,who=user\_email)

user\_yzm = request.POST.get('yzm')

while time.time()-time\_start<=1000\*120 :

##用于实现两分钟内验证码有效的功能

if user\_yzm==yzm:

{成功将用户写入数据库并且告知用户}

{告知用户验证码错误}

return render(request, 'login.html')