

信息安全课程设计说明书

题 目： 可信赖的安全电子商务平台

学 院： 计算机与信息安全学院

专 业： 信息安全

姓 名： 李博研

学 号： 1900301418

指导教师： 王勇

2022年 11 月 20 日

# 摘要

互联网自出现之日起，就一直在改变人们的生活，它不断发展，不断的渗透进我们的生活。传统的行业也不断的与互联网发生碰撞，同时也产生了一些独特的行业。电子商务就是传统行业与互联网结合的常务，它将传统的线下交易转移到了线上交易，极大的方便了人们的生活，不过由于其存在于网络当中，是虚拟的，可能存在各种不安全的因素，慢慢的就成为了电子商务安全方面的挑战。

本篇文章在前半部分主要介绍有关安全电子商务的一些内容，即安全对电子商务发展的重要性介绍，让人们对于电子商务安全方面有个简单的了解与认识，后半部分则是对于系统的设计与实现的内容，侧重于实践。

**关键词**：电子商务交易平台、python、Django、MySQL

目 录

[1、绪论 1](#_Toc121409726)

[1.1、研究背景 1](#_Toc121409727)

[1.2、研究意义 2](#_Toc121409728)

[1.3、国内外研究现状 2](#_Toc121409729)

[2、系统设计分析 3](#_Toc121409730)

[2.1、系统需求分析 3](#_Toc121409731)

[2.2、总体流程的介绍 4](#_Toc121409732)

[2.3、主要模块的介绍 4](#_Toc121409733)

[2.3.1、身份认证 4](#_Toc121409734)

[2.3.2、加密模块 4](#_Toc121409735)

[2.3.3、商品发布模块 4](#_Toc121409736)

[3、实现方案 5](#_Toc121409737)

[3.1、用户注册和登录 5](#_Toc121409738)

[3.2、商品发布 9](#_Toc121409739)

[3.3、用户中心 10](#_Toc121409740)

[3.4、网站主页面 11](#_Toc121409741)

[3.5、其他页面操作的实现 11](#_Toc121409742)

[4、功能测试 13](#_Toc121409743)

[5、结论 16](#_Toc121409744)

[参考文献 18](#_Toc121409745)

# 1、绪论

1999年，第一家C2C电子商务平台—8848诞生，这意味着C2C模式诞生，象征着电子商务平台的起点。三个月后，易趣诞生；紧接着，四年之后，阿里巴巴旗下的C2C网站淘宝网直到2003年才成立。2008年是电子商务飞黄腾达的一年，天猫、京东等电子商务企业慢慢浮现。随着淘宝、京东等电子商务平台的涌现与遍布，电子商务已经经理了二十多个岁月。

电子商务在互联网的遍及于急速繁荣下变得更加强大。《2019年第44次中国互联网络发展状况统计报告》显示：6.39亿网络用户选择网络购物，比2018年尾增加0.29亿，占网络用户整体的74.8%；而手机网络用户中的6.22亿选择网络购物，较前一年年底增长0.30亿，占全部网民的73.4%；并且根据统计可得：网络购物用户平均每天打开淘宝七次，可见电子商务平台在消费者的心中拥有着重要的地位[1]。

电子商务是在虚拟的网络上开展的，进行贸易的双方因而摆脱了传统商品交易那种面对面的买卖销售方式，降低了用户双方交易的成本并且节约了交易时间。然而在拥有便捷优势的同时，缺乏双方对面交易的审核认证及相应的沟通协商，这种信息交换的不确定性及商品信息的不对称性，尤其是对交易双方所应具备的相应的商家经营资质、买家交易资质、售后服务体系等相关的审查或审核几乎为零[2]，这无形中增加了电子商务交易的风险，因而缺乏传统商务交易中的信任感和安全感。同时电子商务平台每天都要处理大量的用户数据，这其中就涉及到用户个人的隐私问题，而如何保护用户个人隐私问题成为了建立电子商务平台的重中之重，这也向人们展示了随着电子商务平台的推陈出新，信息安全问题也慢慢露出破绽。电子商务平台信息安全问题的出现严重制约了电子商务平台的发展，解决电子商务平台中的信息安全问题也变得越来越重要。

## 1.1、研究背景

互联网成为当今时下的主流，也促进了电子商务平台的盛行，电子商务逐渐成为首要的生意形式，并且成为一个非常流行的概念。它是利用计算机与互联网进行商务活动的。从内容来看，是买卖双方通过互联网发布与商务有关的各类信息，开展商务谈判、签约、资金支付和结算，实现商品的配送以及提供售后服务等。从环节来看，是买卖双方通过互联网实现有关商务资金流、信息流和物流的交互。

由于电子商务平台主要处理的是用户操作后产生的信息，相当于数据，这也就是说明电子商务主要处理的便是数据，而对于数据主要集中在四个方便：数据的隐私性、数据的完整性、交易对象可认证行、数据的不可抵赖性[3]。

数据的隐私性指的是在进行交易的各个环节中，数据都必须是安全的，数据都必须保证对外是不可见的，另外，电子商务交易数据往往包含着许多交易双方的私密信息，不能出现在信息传输的过程中被窃取或者被窃取信息不能被窃取者发现其其中秘密，也必须保障其不被非法使用，不能在当事人不知情的情况被使用；数据的完整性是指在进行电子商务交易活动中，所有交易的数据必须是整套的、完整的，不能出现在传输的过程中被篡改的情况，或者在数据不完整的情况下必须能够准确识别，及时丢弃或者做相关处理；交易对象可认证行指的是进行电子商务交易的两方的身份必须可以进行认证，不能出现交易双方不能互相认证身份的情况；数据的不可抵赖性指每一份发送出去的数据，不论是卖方，还是买方，都是没有争议的，不可以抵赖，必须保障交易双方对自己做过的行为不可抵赖。正是由于各方面的原因，导致电子商务平台信息安全问题所受到的关注不断增加，而用什么方法来解决该问题也变得越来越重要。

## 1.2、研究意义

随着网络时代的到来，越来越多的人通过Internet进行商务活动。电子商务的发展前景良好，而其安全问题也变的越来越突出。今年来，网络安全事件不断攀升，电子商务方面的行业都成为了攻击目标，以信息泄露、病毒威胁和身份仿冒为主的网络安全问题正在不断的增多[4]，这些网络安全问题不仅会对企业的形象与信誉造成严重的损害，还会使企业和客户的财产遭到非法用户的窃取。任何个人、企业或商业机构以及银行若通过一个不安全的网络进行商务交易，会导致商业机密信息或个人隐私的泄露，从而导致巨大的利益损失。所以，设计一个安全的电子商务平台，是非常重要的一部分。

电子商务安全是电子商务的得以发展的前提和保障，在互联网上开展电子商务，参与电子商务的用户，都会多少的担忧自己交易的安全问题，只有在安全得到保障的前提下，电子商务才能健康快速的发展，才能吸引更多的参与者;电子商务安全是保证电子市场秩序的前提，虽然网上交易有各种各样的优势，如果电子商务的安全的不到保障，给交易的双方带来重大损失，这将极大地扰乱虚拟市场的秩序，而且用户可能会摒弃网上交易的方式而选择其他更为稳妥的交易形式；电子商务作为国家经济体的重要组成部分，其自身的安全与整个市场经济的安全息息相关，电子商务市场若变得混乱，将导致而人们生活的不安和不便捷。国家经济市场的安全，金融政策以及货币政策的稳定，对外贸易与投资安全就不能在国家经济运作支柱的数字化、网络化环境中得到有效保障[5]。电子商务极其重要，而且目前电子商务安全现状不容乐观，许多问题亟待解决，而对于安全电子商务平台的研究，显的非常有现实意义和价值。

## 1.3、国内外研究现状

电子商务安全涉及到的内容非常的丰富，其中包括数据加密、安全机构、身份认证安全协议等。国内外的大学、相关研究机构以及市场上的各大厂商，已经推出了比较完整的电子商务安全安全技术框架，研究出了一些比较完善的电子商务安全方案，有的已经初步试用。

针对电子商务的系统安全层面，电子商务相对而言是一个繁琐的理论系统体制，在双方的交易过程中虽然保护客户端的安全、传输过程中信息的机密、为止提供服务的服务器端的安全很关键，但是要确保电子商务在交易过程中一个完整体系的安全，所采用的安全措施还需要不断的完善，例如交换信息过程中对信息的保密性、对交易双方各自身份的安全认证、交易能安全顺利完成的交易协议和在交易达成一致后的支付安全，在这整个交易过程中，每一个环节都限制着电子商务交易过程的安全性。

在网络安全方面，互联网上95%的数据包都是通过OSI模型中的传输层和网络层配合来传输的，而传输层的协议主要是TCP协议，网络层的主要协议为IP协议，而TCP/IP协议簇是完全开放的，它们在设计之初是没有考虑安全方面的问题[6]，在互联网的一次数据交换过程中，其最小的交换单位是IP数据包，无论是线上交易还是线上服务，其最终不可避免都要通过互联网来交换数据报文包来达到目的，不发分子通过攻击脆弱的网络可以很容易的获得许多信息，并更换甚至伪造这些信息。现阶段能够提供安全服务的技术和协议很多，但都不是完美的，都在不同层面存在漏洞。并且在不安全的网络上，由于本身安全基础薄弱，建立其上的网络服务也不安全，这些都极大的限制了电子商务的发展。

电子商务最早是出现在美国，随着时间的推移，电子商务在美国发展的最完善，因此他们电子商务安全方面的法律法规也领先其他国家，1987年，《计算机安全法》诞生，1998年，《使用电子媒介作传递用途的声明》发布，确立了电子数据的效用；2000年，通过了《全球和全国商务电子签名法案》，确立起电子签名的法律效力。

而我国的发展还是处于初级阶段，对随着我国电子商务的不断发展，我们不断的认识到电子商务伴随的环境健康安全的重要性。电子商务在国内的发展，各大高校，公司或是相关的研究机构都在不懈的研发各类相关的安全产品，并取得了不错的效果，出现了一些不错的产品。

# **2、系统设计分析**

一个合格的电子商务平台必然是以安全为前提的，可以说安全是电子商务系统的骨架，一个完整的电子商务平台必须拥有对数据进行加密操作的部分，以保障数据的安全性、隐私性。

## 2.1、系统需求分析

该系统设计主要是基于大众用户来开发的一个平台，也就是说用户都拥有相同的权限，不过对于未授权的用户是不具有操作该平台的权限的，因此这就需要对用户是否被授权进行检测；由于该系统是基于web客户端来进行开发的，就需要用户进行账号和密码的输入，而为了保护用户的账号安全，就需要对用户的密码在传输过程中进行加密的操作，也就是说需要一个加密的模块；该系统是一个相当于商品交易的网站，即需要发布商品的界面和存储商品信息的数据表，既然是一个商品交易网站就应该有商品购买和商品加入购物车的需求；同时系统是面向用户的，系统应该拥有用户相关信息，购买的商品的管理等相关操作以及存放用户数据的数据表；该系统考虑到用户可能会有一些错误的操作，导致数据存储到数据库当中，而这时就可以通过数据恢复的模块，来将数据恢复到错误操作之前，但是由于设计的问题，该数据恢复的模块并不能恢复到任意的时间点，只限定在一些具体的操作之前的数据库，如发布商品之前、删除商品、购买商品等相关操作之前的数据库，非常的局限性。通过以上的分析，该系统的主要模块有以下几个模块：包括：身份认证、注册模块、登录模块、加密模块、发布商品、购买商品、数据恢复等模块。

## 2.2、总体流程的介绍

用户访问可以直接访问该网站，而对于其他操作的话则需要用户进行注册的操作，只有注册且登录完成后才可以进行购买、发布商品的操作。

其中对于用户自身信息的也会有相关的展示，同时用户也可以对自己信息数据进行修改、保存等操作。由于用户自身拥有发布商品的权限，因此可以对自己发布商品的进行删除的操作，不过由于实力有限没有一个完整的购买商品的流程，只是一个简单的购买的操作，流程为用户只能先将商品加入购物车之后，才可以对商品进行购买，同时将加入购物车的商品和购买的商品展示在用户中心的界面当中。

## **2.3、主要模块的介绍**

### 2.3.1、身份认证

身份认证又称为“验证”，是指通过一定的手段，完成对用户身份的确认。而该系统的身份认证主要是对相关操作的一种认证，来确保非授权的用户没有对商品进行操作的权限，以此来保护页面内容的有用性，而不是一些不太友好的信息。同时身份认证也是确保用户的唯一性，这样方便于正常的管理操作。

### 2.3.2、加密模块

对于电子商务来说，信息加密技术是最基本的安全措施，它是一种主动的安全防范措施。实质是对以符号为基础的数据进行置换和移位的交换算法，受密钥的关键符号串控制。分为对称加密和非对称加密。而该系统采用的是非对称加密的方式，对于用户来说并非完全是透明的，一部分的数据用户是可以通过一些手段是可以查看的。

采用的方式是RSA的加密方式，在后端自动生成公私钥，然后将公钥发送到前端，然后将用户发送的数据通过公钥加密，后发送到后端，这样可以稍微有效的保护数据在发送过程中的安全性，可以适当地减少一些非法用户截获数据后发现原始数据的风险，而后端则通过私钥解密来得到用户所发送的正常数据，然后来对正常数据进行一系列的操作。

### 2.3.3、商品发布模块

该模块主要是用户正常处理操作的模块，也是该系统的主要构成部分，通过用户来发布商品，再将商品给展示给用户，然后用户购买等一系列操作，因此对于发布商品模块需要特别的对待，该模块主要包含商品的名称、价格、类型、地址以及该商品的图片等内容，同时将数据存储到商品的数据表当中。对于商品中的一些重要信息是必须用户填写的，否则就会导致商品发布失败的现象，就比如商品的名称、价格、类型这些信息对于商品是必不可少的。

# **3、实现方案**

该系统的实现主要是通过python中的Django框架来实现的，对于数据库的实现同样也是利用Django框架中的orm模块来实现的，主要是在使用的过程中发现，通过该模块实现比较简单方便一些，同时也可以避免一些不必要的逻辑方面的错误，就比如sql注入，如果是利用常规的数据库的操作，是有可能被用户利用，构成sql拼接导致非法执行不希望用户执行的操作，而使用该模块则可以适当的避免这一错误的发生。

## 3.1、用户注册和登录

对于该功能的实现，主要包含用户密码数据在传输过程中的加密问题，以及一些数据的验证等问题，还有就是对于相关数据的存储操作。

（1）登录操作的代码实现：

def login(request):

if request.method == "GET":

# username = request.session.get('username', None)

# if username is None:

# 伪随机数生成RSA公私钥对

random\_generator = Random.new().read

rsa = RSA.generate(1024, random\_generator)

rsa\_private\_key = rsa.export\_key()

rsa\_public\_key = rsa.public\_key().export\_key()

# 用session方式存储公私钥，PKCS1格式，方便与后续的验证操作

request.session['privkey'] = rsa\_private\_key.decode()

request.session['pubkey'] = rsa\_private\_key.decode()

pub\_key = request.session.get('pubkey')

return render(request, 'login.html', {'pub\_key': pub\_key})

if request.method == "POST":

pub\_key = request.session.get('pubkey')

user\_name = request.POST.get('username')

password = request.POST.get('passwd')

# 对用户输入的用户名和密码进行检查

username\_db = User.objects.values('username')

if user\_name and password:

privkeystr = request.session.get('privkey').encode()

privkey = RSA.importKey(privkeystr)

cipher = PKCS1\_v1\_5.new(privkey)

# 将base64编码格式password进行解码，后解密

password = cipher.decrypt(base64.b64decode(password.encode()), 'error').decode()

# 对用户的身份进行验证，是否存在于数据库当中

for user in username\_db:

if user\_name == user['username']:

user\_current = User.objects.get(username=user\_name)

# 由于用户密码是以MD5的形式存储在数据库当中的，因此需要进行MD5值的比较

password\_md5 = hash\_md5(password)

if password\_md5 == user\_current.password:

request.session['user\_name'] = user\_current.username

request.session['user\_id'] = user\_current.id

return HttpResponseRedirect('/index/')

else:

# 进行相应的提示操作，便于用户的使用体验，对于提示的内容也需要注意不能过于具体，应具有一定的朦胧感，防止有心人进行利用。

error = "用户名或密码错误"

else:

error = "用户名或密码错误"

else:

error = "用户名或密码错误"

return render(request, 'login.html', locals())

else:

error = "用户名或密码不能为空"

return render(request, 'login.html', locals())

（2）注册界面的代码实现：

def register(request):

if request.method == "GET":

random\_generator = Random.new().read

rsa = RSA.generate(1024, random\_generator)

rsa\_private\_key = rsa.export\_key()

rsa\_public\_key = rsa.public\_key().export\_key()

# 用session方式存储公私钥，PKCS1格式

request.session['privkey'] = rsa\_private\_key.decode()

request.session['pubkey'] = rsa\_public\_key.decode()

pub\_key = request.session['pubkey']

return render(request, 'register.html', locals())

if request.method == "POST":

# 获取注册传入的参数

sex = request.POST.get("sex")

email = request.POST.get("email")

phone = request.POST.get("phone")

username = request.POST.get("username")

pub\_key = request.session['pubkey']

if not username:

username\_error = "请输入正确的用户名"

return render(request.GET, 'register.html', locals())

# password\_1 和 password\_2加密

password\_1 = request.POST.get("passwd1")

password\_2 = request.POST.get("passwd2")

if password\_1 and password\_1:

# 利用存储在session中的密钥进行解密

private\_str = request.session.get('privkey').encode()

private = RSA.importKey(private\_str)

cipher = PKCS1\_v1\_5.new(private)

# 解密

password\_1=cipher.decrypt(base64.b64decode(password\_1.encode()),'error').decode()

password\_2=cipher.decrypt(base64.b64decode(password\_2.encode()), 'error').decode()

# 对密码强度进行判断

flag = 1

flag = check\_password(password\_1)

if flag == 0:

password\_1\_error = "密码强度不够，请重新设置密码"

return render(request, 'register.html', locals())

else:

password\_1\_error = "密码不能为空"

return render(request, 'register.html', locals())

# 利用hash函数对password\_1密码进行加密处理

password\_m1 = hash\_md5(password\_1)

# 对password\_2密码进行hash加密处理

password\_m2 = hash\_md5(password\_2)

if password\_m1 != password\_m2:

password\_2\_error = "两次输入密码不一致"

return render(request, 'register.html', locals())

# 查询用户是否注册

try:

old\_user = User.objects.get(username=username)

username\_error = "当前用户已被注册"

return render(request, 'register.html', locals())

# 若当前用户没有找到，则会报错，说明当前用户名可以使用

except Exception as e:

try:

user = User.objects.create(username=username, password=password\_m1, sex=sex, email=email,phone\_number=phone)

# 注册成功

success = "注册成功！请前往登录"

# request.session['username'] = username

return render(request, 'register.html', locals())

except Exception as e:

# 若创建不成功，报错

username\_error = "用户名已被占用"

return render(request, 'register.html', locals())

（3）MD5加密

MD5在这里主要用于对用户密码进行加密，加密用于存储用户密码，同时用于用户密码的验证。

# hash加密存储与验证

def hash\_md5(ciphertext):

m = hashlib.md5()

# 对待加密的明文进行update方法处理，输入的形式为字符串

m.update(ciphertext.encode())

# 以十六进制存储

return m.hexdigest()

（4）对用户密码强度的判定

# 密码强度的判断，通过对是否含有数字，字母，大小写，特殊字符来确定密码的强度，将含有每一个该特征记为1，对大于3的判定为合格的密码强度，否则为不合格。

def check\_password(password):

d = al = au = pu = 0

for ch in password:

if ch in string.digits: # 判断是否含有数字

d = 1

elif ch in string.ascii\_lowercase: # 判断是否含有字母

al = 1

elif ch in string.ascii\_uppercase: # 判断是否含有大小写

au = 1

elif ch in string.punctuation: # 判断是否含有特殊字符

pu = 1

if len(password) < 8: # 判断密码的长度

strong = 0

else:

strong = d + al + au + pu

if strong < 3:

return 0

elif strong >= 3:

return 1

## 3.2、商品发布

商品发布的代码实现：

def issue\_form(request):

flag = 0

if request.method == "POST":

goods\_name = request.POST.get('goods\_name')

goods\_detal = request.POST.get('goods\_detal')

goods\_price = request.POST.get('goods\_price')

categorys = request.POST.get('categorys')

address = request.POST.get('address')

pho\_num = request.POST.get('pho\_num')

file\_img = request.FILES.get('file\_img') # 获取文件

file\_chunks = file\_img.chunks()

# 文件保存路径

file\_name = os.path.join("image", do\_file\_name(file\_img.name)).replace('\\', '/')

# 文件完整的保存路径

file\_path = os.path.join(settings.MEDIA\_ROOT, file\_name).replace('\\', '/')

with open(file\_path, "wb") as file:

for chunk in file\_chunks:

file.write(chunk)

new\_issue = Goods()

new\_issue.name = goods\_name

new\_issue.detal = goods\_detal

new\_issue.price = goods\_price

new\_issue.master\_pho = pho\_num

userid = request.session.get('user\_id')

new\_issue.user\_id = userid

sort = Sort.objects.get(id=categorys)

new\_issue.sort\_id = sort.id

new\_issue.img = file\_name

try:

# 对操作之前的数据库的进行备份

os.system(

'mysqldump -uroot -proot web1db> database/web1db.sql')

new\_issue.save()

flag = 1

except Exception as e:

print(e)

return HttpResponse(flag)

## 3.3、用户中心

用户中心主要包含用户的信息、加入购物车的商品记录、发布商品的记录、购买商品的记录，其中主要以用户个人信息为主，且用户可以对自己的个人信息进行更改保存。

用户中心相关数据操作的代码实现：

def user\_center(request):

user\_name = request.session.get('user\_name')

user\_id = request.session.get('user\_id')

if request.method == "GET":

# 该用户发布的商品

issue\_list = Goods.objects.filter(user\_id=user\_id)

# 该用户购物车中商品

goods\_list = Cart.objects.filter(cart\_user\_id=user\_id)

# 用户购物记录

user\_buy = Buy.objects.filter(user\_id=user\_id)

cart\_list = [] # 存储用户购物车的商品

for goods in goods\_list:

cart\_id\_dict = {'cart\_id': goods.id, 'cart\_create': goods.cart\_create\_time,

'good': Goods.objects.get(id=goods.goods\_id)}

cart\_list.append(cart\_id\_dict)

buy\_list = [] # 存储用户购物记录

for buy in user\_buy:

user\_buy\_dict={'buy\_id':buy.id,'buy\_create':buy.create\_time,'good':Goods.objects.get(id=buy.good\_id)}

buy\_list.append(user\_buy\_dict)

user = User.objects.get(id=user\_id)

if user.birthday is not None:

user\_birthday = user.birthday.strftime('%Y-%m-%d')

else:

user\_birthday = user.birthday

return render(request, 'user\_center.html', locals())

## 3.4、网站主页面

@csrf\_exempt # 保障网站与后端数据的正常接受，也是为了防止csrf跨站请求伪造的发生。

def index(request):

user\_name = request.session.get('user\_name')

user\_id = request.session.get('user\_id')

if user\_id is not None: # 判断当前是有用户操作还是无用户操作，若有用户操作则将非该用户发布的商品进行展示

user = User.objects.get(id=user\_id)

goods\_list = Goods.objects.filter().exclude(user\_id=user\_id).order\_by('create\_time')

return render(request, 'index.html', locals())

else:

goods\_list = Goods.objects.filter().order\_by('create\_time')

return render(request, 'index.html', locals())

## 3.5、其他页面操作的实现

（1）商品加入购物车实现

主要是通过获取该商品的id值以及用户自己的id值来实现对应关系的，并将数据存储到Cart购物车表当中。

def join\_cart(request):  
 result = 1  
 good\_id = request.GET.get("id")  
 user\_id = request.session.get("user\_id")  
 cart = Cart()  
 cart.cart\_user\_id = user\_id  
 cart.goods\_id = good\_id  
 try:  
 os.system(  
 'mysqldump -uroot -proot web1db> database/web1db.sql')  
 cart.save()  
 except Exception:  
 print(Exception)  
 result = 0  
 return HttpResponse(json.dumps(result))

（2）发布商品的删除

与加入购物车是类似的，通过获取商品的id，来对其进行删除的操作。

def del\_goods(request):  
 result = 1  
 good\_id = request.GET.get("id")  
 try:  
 os.system(  
 'mysqldump -uroot -proot web1db> database/web1db.sql')  
 Goods.objects.filter(id=good\_id).delete()  
 except Exception:  
 result = 0  
 return HttpResponse(json.dumps(result))

（3）购物车商品的删除

购物车商品的删除，相对来说比较复杂一些，不过并不是通过获取商品的id来进行删除的，因为一个用户可能多次将该商品加入到购物车当中，因此可能会造成删除多个的事件的发生，这时就需要获取购物车表中该删除商品的id来进行删除，而解决这个问题，则是通过在传递数据的时候将该数据一同传输到前端来实现的。

def del\_carts(request):  
 result = 1  
 user\_id = request.session.get("user\_id")  
 good\_cart\_id = request.GET.get("id")  
 del\_cart\_dict = {}  
 try:  
 os.system(  
 'mysqldump -uroot -proot web1db> database/web1db.sql') #实现对数据库备份的操作  
 del\_cart\_dict["cart\_user\_id"] = user\_id  
 del\_cart\_dict["id"] = good\_cart\_id  
 Cart.objects.filter(\*\*del\_cart\_dict).delete()  
 except Exception as e:  
 result = 0  
 print(e)  
 return HttpResponse(json.dumps(result))

# 4、功能测试

（1）注册、登录测试

注册成功时页面的显示，右下方会提示注册成功的显示，点击可以跳转到登录界面。



图4-1

登录界面，用户名或密码输入不正确，会出现相应的提示



图4-2

（2）主页面则是由轮播图，发布商品和分类栏构成，点击对应的元素，可以跳转到与之相对应的界面，不过由于技术的不足，对于分类栏的设计并不是非常的方便，而是通过页面的跳转来实现分类功能的。

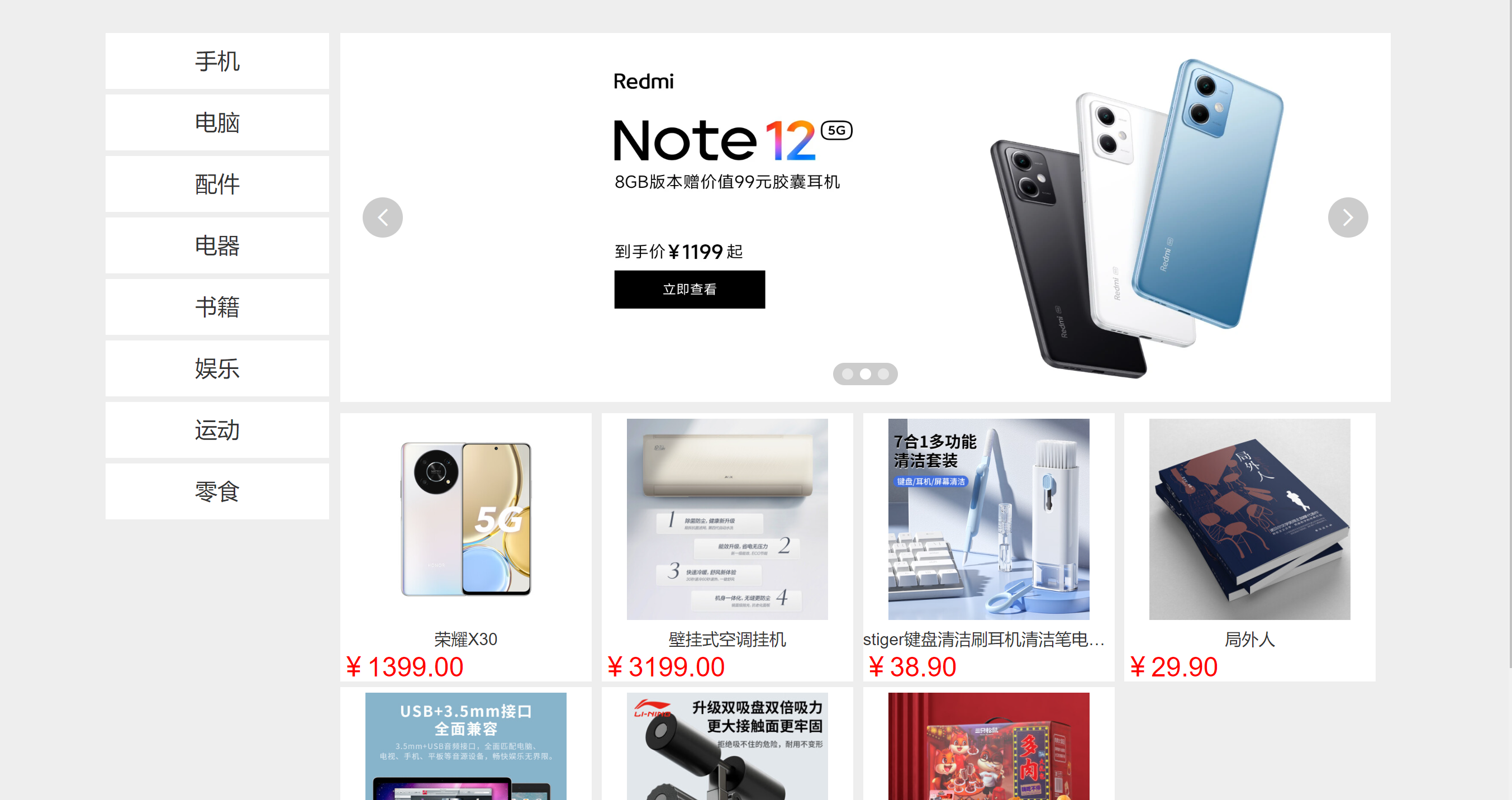


图4-3

（3）商品页面的展示，设计的比较简单，涉及到的内容也比较少，对于没有登录时，点击加入购物车是无法正常操作的，即点击没有任何的反应，而当用户登录后，点击则会有弹框用以提示用户。



图4-4

（4）发布商品页面，对于一些必须要填写的项目没有输入，则是会出现相应的弹窗提示；其中的图片显示初始时默认的图片，当用户发布完一次之后，该图片就会变为上一次发布时的图片，这也是在设计时的不足，不能将其调整为初始的图片。



图4-5

（5）用户中心包含发布的商品、购物车、个人信息、购物记录四个模块，对于个人信息，用户可以进行修改，同时也会将修改后的内容展示给用户，其中生日是一个统一设定的一个值条件是当前用户的生日为空时。



图4-6

# **5、结论**

电子商务伴随着互联网的产生而出现，世界上著名电子商务公司有很多，如亚马逊等，电子商务经历了十多年的发展，现阶段已经渐渐的趋向稳定，国内的电子商务起步较国外起步稍晚，淘宝时最早的电子商务公司之一，经过了十多年的发展，淘宝已经成为了国内最大最成熟的电子商务平台，目前国内的电子商务发展之势呈现良好的态势，大的电子商务平台有淘宝、天猫、京东、其他还有许多领域的电子商务平台。未来电子商务的发展将会更加的繁荣。

由于电子商务的特殊性，安全对于电子商务来说时非常重要的，可以理解为时必不可少的一部分，而本文则抓住这一部分，从安全开始着手考虑，设计了此次安全电子商务系统。

本篇文章在前半部分主要介绍有关安全电子商务的一些内容，即相关背景知识等内容，让人们对于电子商务安全方面有个简单的了解与认识，后半部分则是对于系统的设计与实现的内容，侧重于实践。为了完成有关安全电子商务的背景知识，进行了大量的文献的查找，在最初的时候由于自己出发点的错误，导致浪费了大量的时间去查找电子商务相关的内容，后在写总结的时候意识到出发点偏差，及时纠正对安全的电子商务方面开始着手，才刚好符合此次的课题内容。其实在最初的系统设计时并没有考虑到商品发布的问题，后来在设计页面实现完成是，发现没有商品的发布，主界面就没有内容可以填充，后来参考其他购物网站添加了商品发布的内容。

该系统在数据传输的过程中，其中一部分的数据是存储在session当中的，而由于未对其设置时间的限制，因此只要用户没有退出账号，则session中的数据会一直存在，对于这些可能会产生一些不必要的麻烦。还有就是在设置url路径和用户页面时的变量设计的并不是非常的严谨，若是在一个比较大的项目当中，可能会为后期的维护造成较大的成本，比较麻烦。最重要的是由于掌握的知识不是非常的充分，对于安全方面的设计并没有涉及多少，主要在于传输过程中的加密问题，以及数据库的备份方面有一点的涉及。

通过本次的课程设计，使自己了解到了利用python来设计网站的方法，也是自己初次使用python来设计网站，过程并不是非常的顺利，不过当网站设计完成后，感觉还是非常良好的，同时也让自己对这一方面的内容有了一定程度的了解，以及一个小型网站在设计时的大致流程还是非常重要的这些知识。尤其是对于安全方面，一个安全的网站设计时非常的困难的，毕竟只要是人，就一定会在设计的过程中出现一些错误，从而造成一些不必要的漏洞的产生。

# 参考文献

1. 刘阳. 电子商务平台信息安全机制的研究[D].黑龙江大学,2020.DOI:10.27123/d.cnki.ghlju.2020.000489.
2. 张庆丽. 电子商务安全策略研究[D].河南大学,2012.
3. 胡正刚. 安全电子商务系统的设计与实现[D].北京邮电大学,2016.
4. 飞多高. 电子商务中的网络安全问题研究[EB/OL].中原工学院信息商务学院.2018.
5. 张炯明. 安全电子商务使用技术.清华大学出版社,2002,4(I):27~34.
6. Mo J, La R J,Anantharam V, et al. Analysis and comparison of TCP Reno and Vegas[J].Proceedings - IEEE INFOCOM, 1999,3:1556--1563.
7. 刘明华.杨蜜.LIU Ming-hua.YANG Mi 国外电子商务安全技术研究现状及发展趋势[J].-信息网络安全 2009（8）.