[网上虚拟银行开发技术文档]

（密码学原理与实践-实践部分）

[项目名称：虚拟银行 Virtual Bank]

[姓名：　高博为　　]

[学号：　1160300103　　]

目录

1. 背景与意义

1.1项目开发意义

在现实生活中，随着移动互联网的高速发展，传统金融服务已经不能够满足日益增长的人们的个人金融服务对便利性的需求，现在，很多人出门已经习惯不带现金或只带少量现金，相对以往也更少去现实中的银行办理业务。移动支付也正是在这样的行业大发展背景下,得到了迅速发展。如今,全球互联网在快速发展,以安全、高效、便捷为优势的网上金融服务在电子支付领域备受推崇。

但与此同时伴随而来的问题也很突出，众所周知在金融方面最重要的就是安全问题，即使是电子商务领域也是如此，一旦网上金融交易平台面临层出不穷的网络攻击无法抵御的话，轻则造成信息泄露，重则用户资金被窃取，无论哪种后果都是要尽量避免的。

因此这个项目的重点将放在预防网络攻击与窃听，保护用户隐私及资金安全的方面。一个安全高效的网上虚拟银行永远是有广阔前景的。

1.2 国内外现状及技术综述

目前，国内众多的商业银行纷纷设立网上银行业务，外资银行也竞相介入，国内网上银行的市场竞争与日趋激烈。

我国网上银行主要包括两种：一种是完全依赖于因特网发展起来的全新的电子银行，它所有的业务都依靠因特网进行；另一种是在现有的商业银行基础上发展起家来的，把银行传统业务捆绑到因特网上，开设新的电子服务窗口，即所谓传统业务的外挂电子银行系统。目前我国大多数网上银行交易系统都属于第二种。

网上银行最早起源于美国，其后迅速蔓延到 Internet 所覆盖的各个国。

目前，国外网上银行的类型主要有三种：

第一种是虚拟银行，即没有传统的营业网点，直接建立在 Internet 上的网络银行。

第二种是实体与虚拟结合的网络银行，即将已存在的传统银行和既有的封闭型专用网络系统与 Internet 联网，提供真正的互联网服务，允许客户直接通过

Internet 直接查询账户余额、转移资金、付款等。

第三种是发布消息型网络银行，即将业已存在的传统银行在 Internet 上设立网址，通过主页介绍银行自身情况、发布有关金融的各种新闻和消息等，并未在网上开展传统的银行业务，这尚不能成为真正的网络银行。

现在发展起来的 Martbase,Google Checkout,Paypal,MoneyBookers,2Checkout 是国外最主要的网络支付。

2. 需求分析

2.1 总体需求

本实验基于 Web 实现了一个安全的网上虚拟银行,前后端之间采用加密通信，防重放攻击等手段实现较为安全的资金流通。

本虚拟平台共有三类用户。

一类是普通用户，普通用户在银行平台可以使用充值，提现，转账，查看与修改个人信息等业务，同时，每次在新设备上登录时，若需要进行充值等资金相关的操作，都需要登陆一次 CA 中心，以将自己的私钥保存在自己的设备上，之后在这台设备上登录并存取资金时则不需要进行验证。

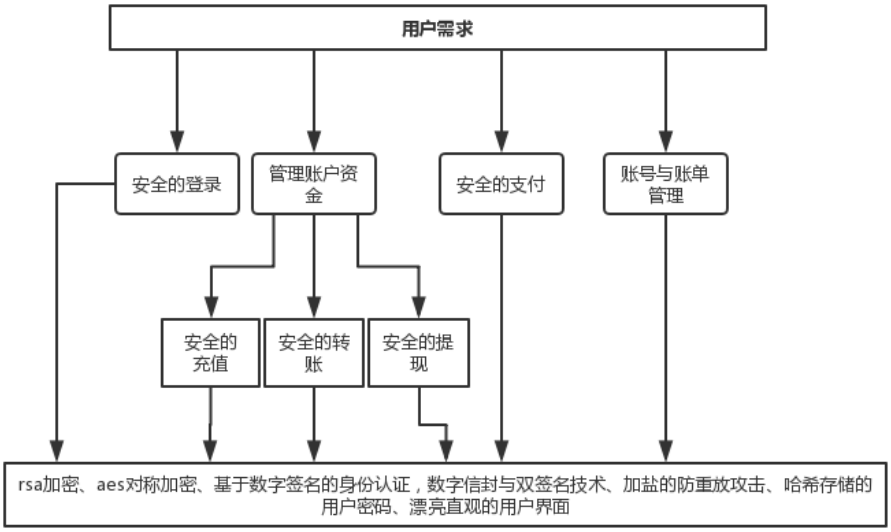
第二类是与虚拟银行协作的电子商务平台，这类用户使用银行为其提供的支付接口完成用户到商家的资金转移，在交互过程中，有证书验证，加密处理，以保证安全性。

第三类是系统的管理员用户，同时采取一定的措施防止有权限的管理员得到银行用户的密码等私密信息，以防信息泄露。

整个系统尽可能的保证了信息交互的安全性，能够抵御身份冒充，重放攻击，中间人，数据伪造等攻击，也能有效防止信息泄露，保护用户隐私。

2.2 功能需求

以下所有功能都实现了最基本的安全要求，即与服务器交互过程的加密，以及登录与资金存取转账时的防重放攻击处理。资金转账存取时隐式的经过了用户私钥签名，后台会通过 CA 认证中心得到用户公钥进行身份验证，有较强的安全性。



需求分析图如上

需求详述：

2.2.1 用户注册：

用户通过注册成为系统用户并登录验证身份后，可以使用该系统提供的服务。注册时包括用户名、密码、银行卡号等基本信息。用户提供的密码将以哈希值的形式存储于数据库中，以保证不会造成信息泄露。

2.2.2 用户登录：

用户通过系统的相关验证设计，登陆系统验证身份。在用户登录时为了保障用户账户资金的安全性，要求用户首次登陆时需要设定支付密码，在之后进行所有的资金相关的操作时，都要求输入支付密码，支付密码必须严格等于 8 位数字。支付密码也是以哈希值的形式存储。

2.2.3 充值：

用户可以从银行卡中向虚拟银行中自己的账户充值资金。

2.2.4 提现：

用户可以将自己账户中的资金提取到自己的银行卡中，这个过程也是安全的。

2.2.5 转账：

用户可以将自己账户中的资金转至别的用户账户中，这个过程要求接受转账的用户必须已经在系统中注册，同时这个过程也经过加密处理。

2.2.6 付款：

在商家平台将商品加入购物车后，点击生成订单后将跳转至本系统提供的支付页面，用户输入注册时绑定的预留手机号与支付密码即可确认，若验证无误则将跳转至 CA 认证中心进行签名操作，系统收到双签名后将进行转账。
付款操作。

2.2.7 账户管理：

用户在这里可以查看与修改自己的个人信息，这一部分在系统页面上是分为两个页面的，一个页面可以查看个人信息并上传头像，另一个页面则用来修改个人信息，包括登录密码和付款密码，以及预留手机号与卡号等。

2.2.8 账单管理：

用户可以查看自己的消费记录，消费记录分为转账、付款、提现和充值四种，以不同颜色的标签显示，并分为收入与支出两大部分显示，十分清晰直观。

2.2.9 用户主页：

主要信息展示页面，会显示用户账户的余额与支出总额，以及会根据最近的收支情况绘制一幅图表，以为用户提供更为直观的信息。

2.2.10 管理员功能需求：

管理员登陆：管理员通过网页端通过验证后完成登陆，登陆系统后可以完成相应的管理操作，这一部分有 Django 默认提供。

数据库管理：管理员可以查看系统的所有注册用户的信息以及所有生成的账单信息，和对应的数据库表单，方便系统管理与修改。

2.3 性能需求

1）网站的页面载入速度较快，正常网速下每个页面载入时间小于 1s。
2）满足多人同时进行存取款及交易操作。
3）用户的身份认证过程需要安全可靠，速度快，认证登录时间不能过久。
4）对不同浏览器支持性都要较好，能够兼容大多数现代浏览器。
5）系统可以长时间持续运行。

3.概要设计

3.1实验环境

网络带宽不低于 50Mbps
内存 4G，硬盘 500G，双核 CPU 处理器

3.2软件环境

操作系统：
Windows10 系统
开发环境：
Chrome 浏览器
Pycharm IDE 2018.3.1
Python3.6
Web 服务器：
Django 2.2
数据库：
Sqlite
开发语言：
Python,JavaScript,Html,Css

3.3用户环境

主流新版本浏览器

如：chrome，Firefox, Edge 等

3.4 模块划分

将主要功能从逻辑上抽象划分后如下图所示：

3.5模块定义

3.5.1 用户模块：

主要功能：

包含用户管理（即登陆与注册）功能，账户资金管理功能，账单功能，信息查看与修改功能，用户界面在内的一个较大的模块，也是显示给用户的外壳。

关联关系：

是其他所有模块所服务的高层对象，使用其他模块所提供的功能，直接与用户交互。

3.5.2 安全模块：

主要功能：

基于加密算法（包括 RSA，AES），认证技术（数字签名，数字证书），以及最基本的密码比对为用户提供账户信息安全、资金安全，另外还做了安全存储（所有密码哈希存储）与防重放攻击。

关联关系：

是整个虚拟银行系统的安全保证，为用户保证一切操作的安全性。

3.5.3 管理员模块：

主要功能：

使系统认证的管理员账户可以对数据库进行细致的管理工作，可以更好地维护整个银行系统

关联关系：

系统管理与维护

3.5.4 存储模块：

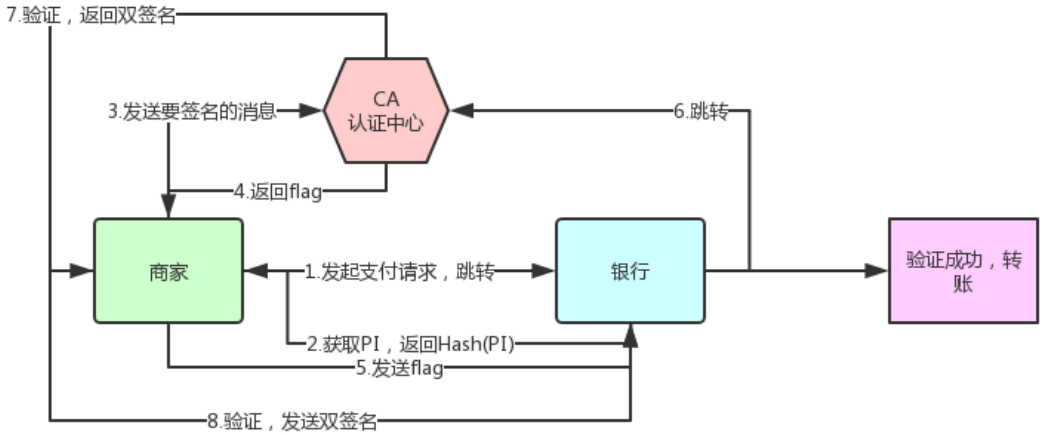
主要功能：

存储所有用户的个人信息，维护用户数据，是系统运行的基础。

关联关系：

其中一部分参与到安全模块中

3.5.5 支付模块



主要功能：

与其他模块交互，帮助完成支付功能。

关联关系：

使用许多模块提供的服务。

3.6 主要技术

3.6.1 加密技术：

机密性：

RSA 非对称加密：

服务器保存有自身的一对 Rsa 公私密钥对，用户可以获取服务器的公钥对要传递的信息进行加密，服务器收到后用自身私钥解密。

信息传递时，使用用户私钥对密文摘要生成数字签名，服务器收到后用用户公钥进行验证，以进行身份验证。

AES 对称加密：

与电商平台交互时使用，避免了 RSA 加解密的速度慢，对要加密消息长度有限制等多种弊端。

完整性：

MD5 哈希算法：

服务器存储用户登陆密码与支付密码时使用，以保证用户密码不被恶意管理员得知，保护用户隐私与信息资金安全。MD5 将任意长度的“字节串”映射为一个 128bit 的大整数，并且是通过该 128bit 反推原始字符串是困难的，换句话说就是，即使你看到源程序和算法描述，也无法将一个 MD5 的值变换回原始的字符串

SHA 哈希算法：

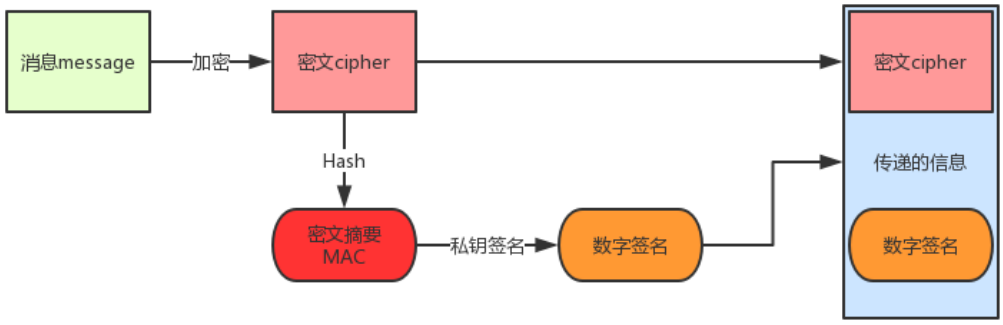
主要在通信过程中使用，对消息加密后的密文连接后做哈希，生成 MAC 标签，用于保证信息未被篡改。

3.6.2 认证技术：

Hash 对比：

将用户密码以 MD5 哈希的形式存储于数据库中，用户登录或支付时，将收到的密码做一次 MD5 哈希，与数据库中存储的密码进行比对，如相同则允许操作，否则拒绝。系统在并不知道用户密码的明码的情况下就可以确定用户登录系统的合法性。这可以避免用户的密码被具有管理员权限的用户知道。

RSA 数字签名：



RSA 数字签名流程图

通过进行资金操作的用户的私钥，对卡号，密码等私密信息的密文形成的消息摘要做签名，数字签名具有不可抵赖性和不可伪造性，用户私钥存储于用户本地，所以一旦签名后，配合数字证书技术进行验证签名，若验证成功，则认为身份未经顶替。

数字证书：

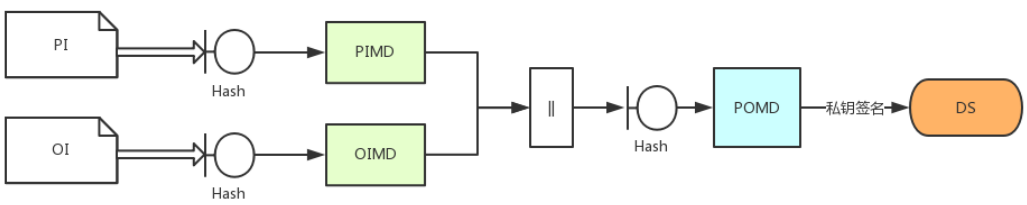
项目中数字证书结构示意图

为了避免数字签名技术中公钥可能被替换的漏洞，与 CA 协作使用数字证书技术进行身份认证。

3.6.3 防重放攻击技术：

为了防止用户登录或资金转移过程中发送的报文被中间人劫持，然后原封不动的重新发送以绕过安全措施，采用防重放技术进行处理，主要采取加盐 Hash 技术，即将密文与一个随机盐值进行连接后 Hash，盐只使用一次，用后即废，保证了拦截到的消息无法通过下一次认证，保证了安全性。

3.6.4 双签名技术：



双签名技术流程示意图

与商家协作的支付接口采用双签名技术验证交易的有效性，银行接收订购信息 OI 的哈希值与双签名，验证双签名的有效性，若有效，则进行转账操作，否则拒绝转账

生成过程：

1） 生成 OI（订单信息）、PI（支付信息）的消息摘要 OIMD、PIMD

2） 将 OIMD 和 PIMD 连接，形成 PO，再把 PO 经过散列计算得到 POMD

3） 用户使用其私钥 K 对 POMD 加密生成 DS

使用过程：

1） 银行获得 DS、OIMD 和支付信息

2） 商家获得 DS、PIMD 和订单信息

验证过程:

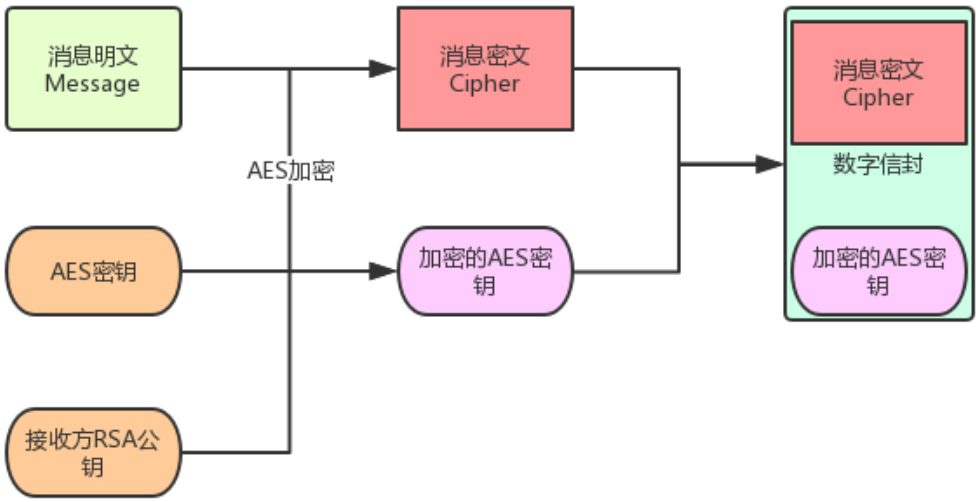
银行验证：银行首先计算支付信息的消息摘要 POMD，然后和商家的 OIMD 连接，得到 OI，对 OI 求信息摘要。将 DS 用用户的公钥解密，将得到的结果和 OI 的信息摘要对比，一致则说明正确。

商家验证：商家首先计算订单信息的消息摘要 IOMD，然后和银行的 PIMD 连接，得到 OI，对 OI 求信息摘要。将 DS 用用户的公钥解密，将得到的结果和 OI 的信息摘要对比，一致则说明正确。

双签名技术保证顾客传递给商家和银行的信息相互隔离，同时又确保信息的一致性，杜绝被顾客、商家和银行其中任一一方伪造。

3.6.5 数字信封技术：

。



项目中使用的数字信封技术示意图

与电商平台的交互过程采取数字信封技术，电商使用接口时，用银行的公钥加密一个 AES 密钥，其余信息则使用 AES 密钥加密，之后的信息交互则都使用 AES

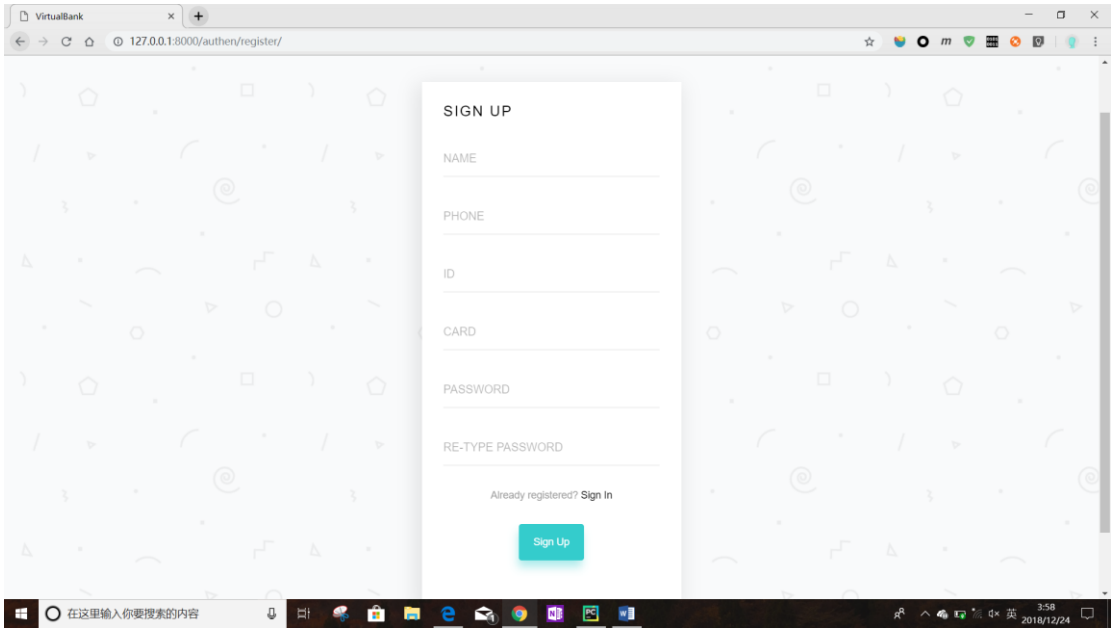
加密，数字信封的好处是提高了加密速度，避免了对称密钥的分发，保证了数据的机密性。只有用接收者的私钥才能够打开此数字信封，确保只有接收者才能对消息密文解密，也避免了 RSA 加密对消息长度的限制。

4. 详细设计

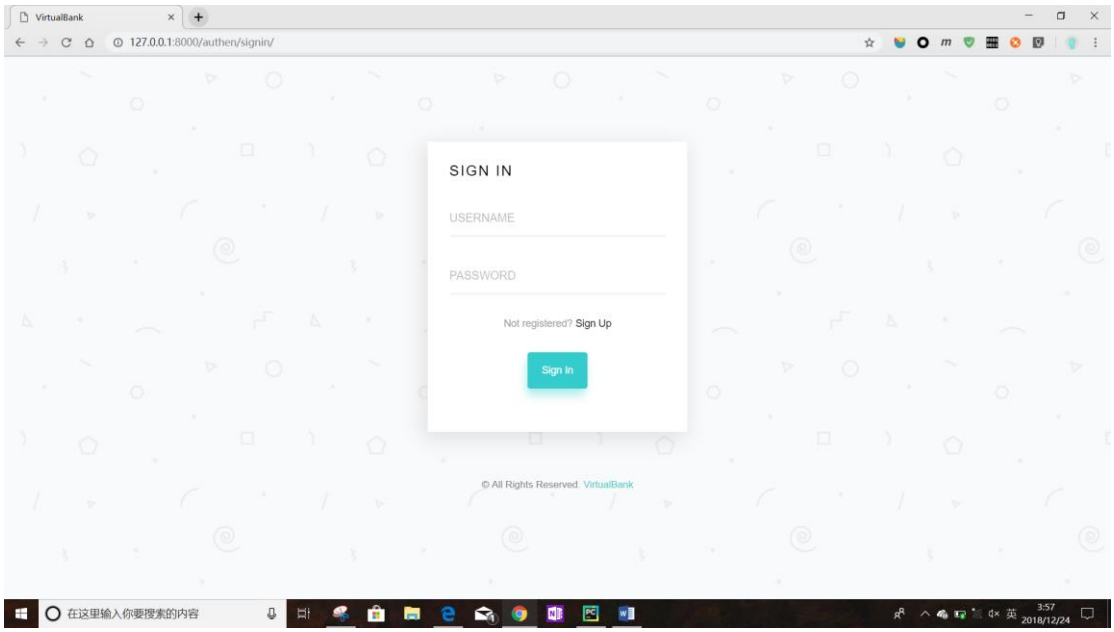
4.1 界面设计

1）注册，登录界面设计：

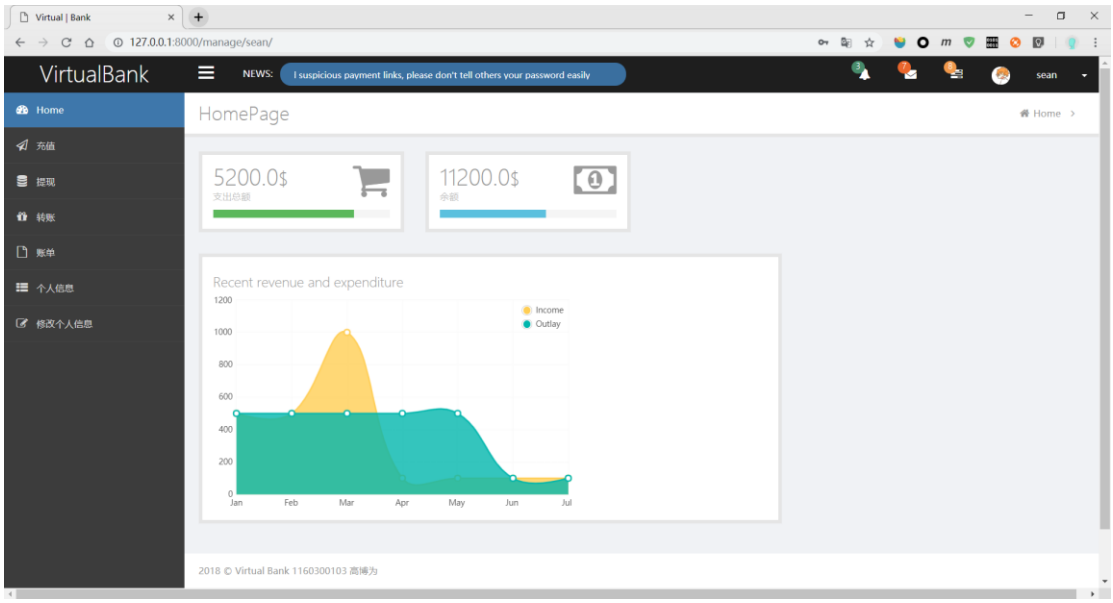
注册：

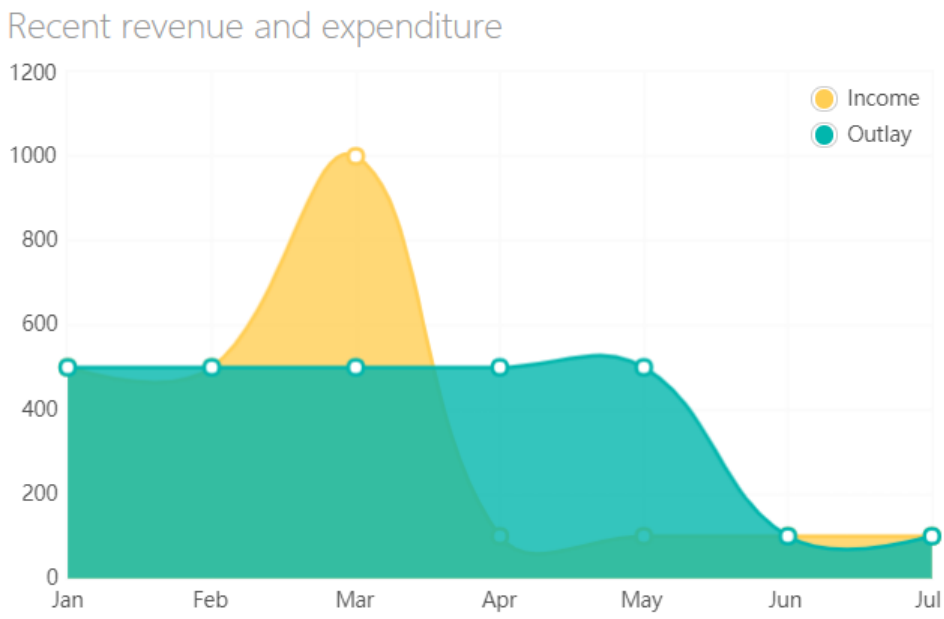


登录：

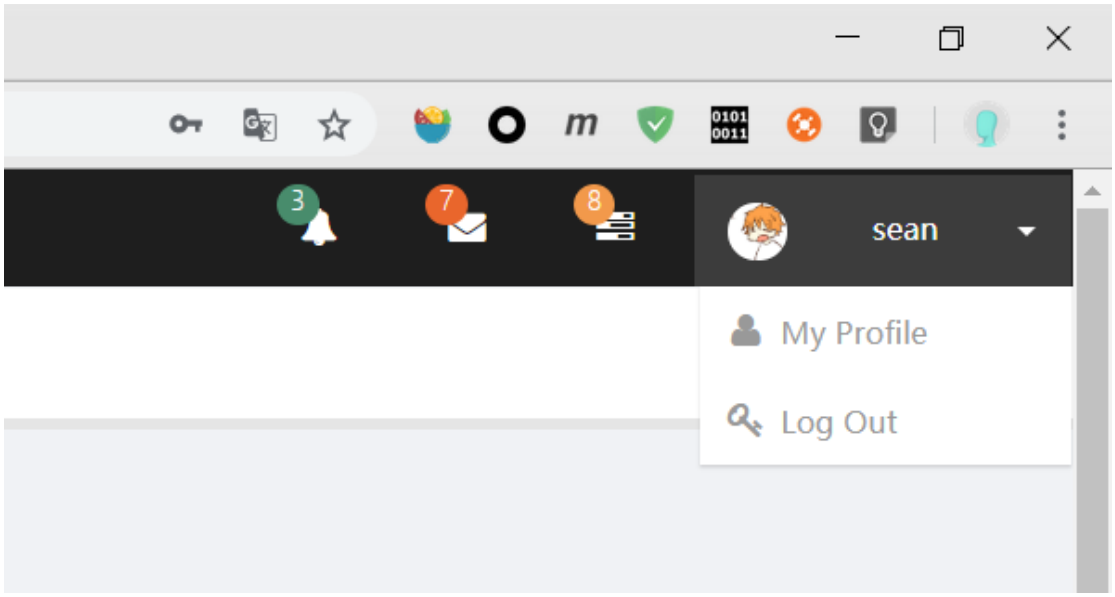


2）用户主页：





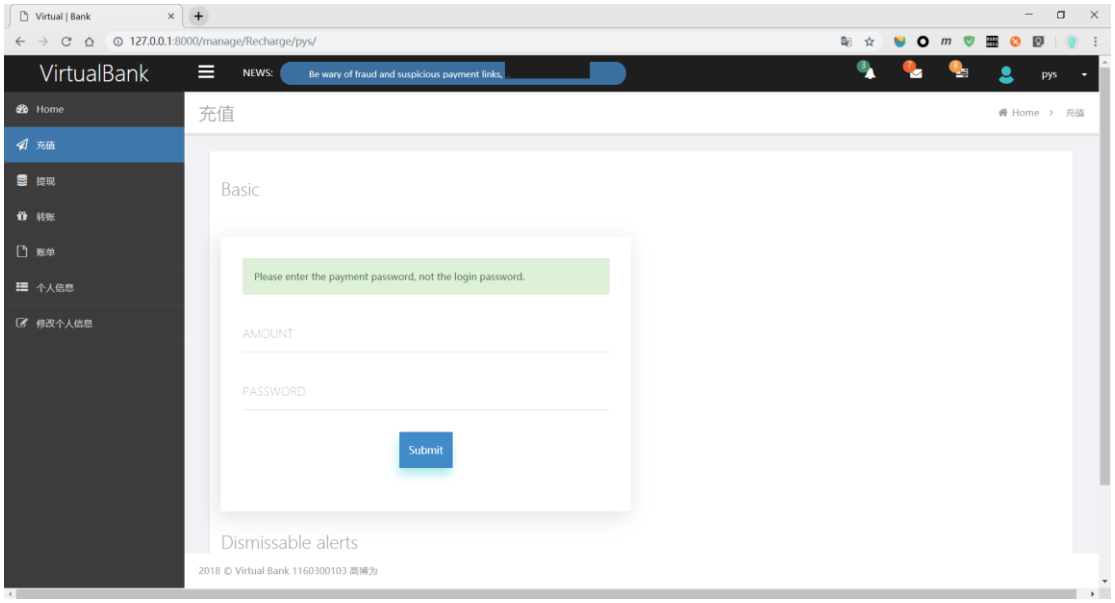
可以看到图中左侧为导航侧边栏，可以引导用户至各个功能页面，而 Home 页面中间为支出总额与账户余额，下方为一个根据用户近期收入与支出状况绘制的图表，可以为用户提供简介直观的账户信息，侧边栏与最上方信息条为整个系统用户页面共享复用的部分，白色部分则为各不同页面重写的部分。

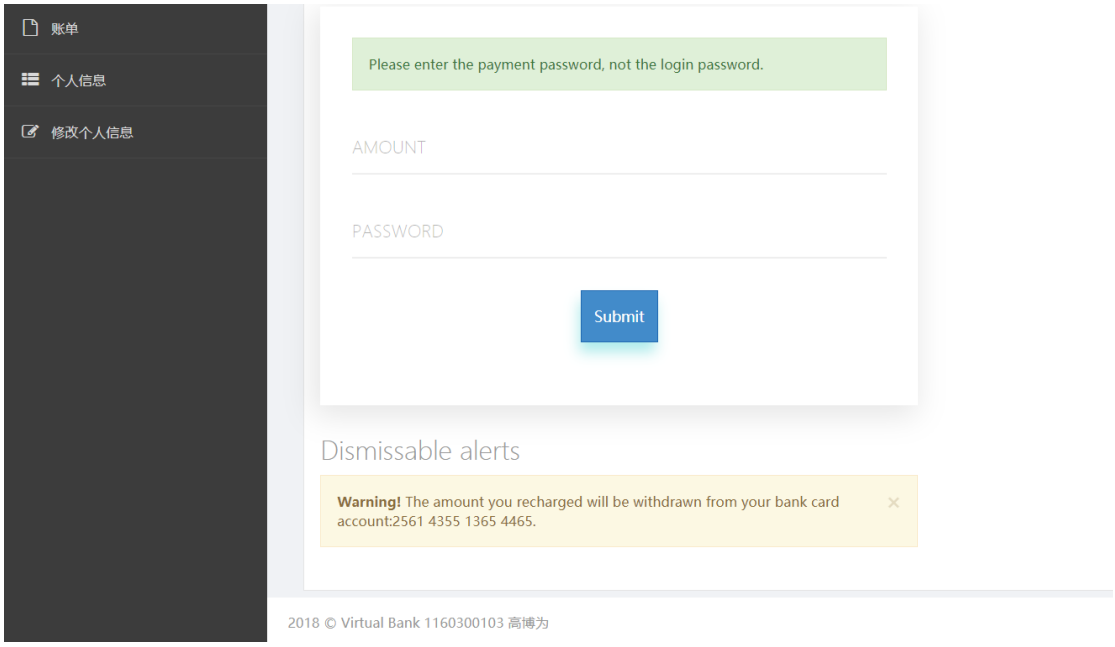


点击用户头像区域可以看到两个按钮，my profile 会引导用户至个人详细信息页面，Log out 则是注销功能，可以清除服务器中 session 所保存的登陆信息，防止他人恶意登录。

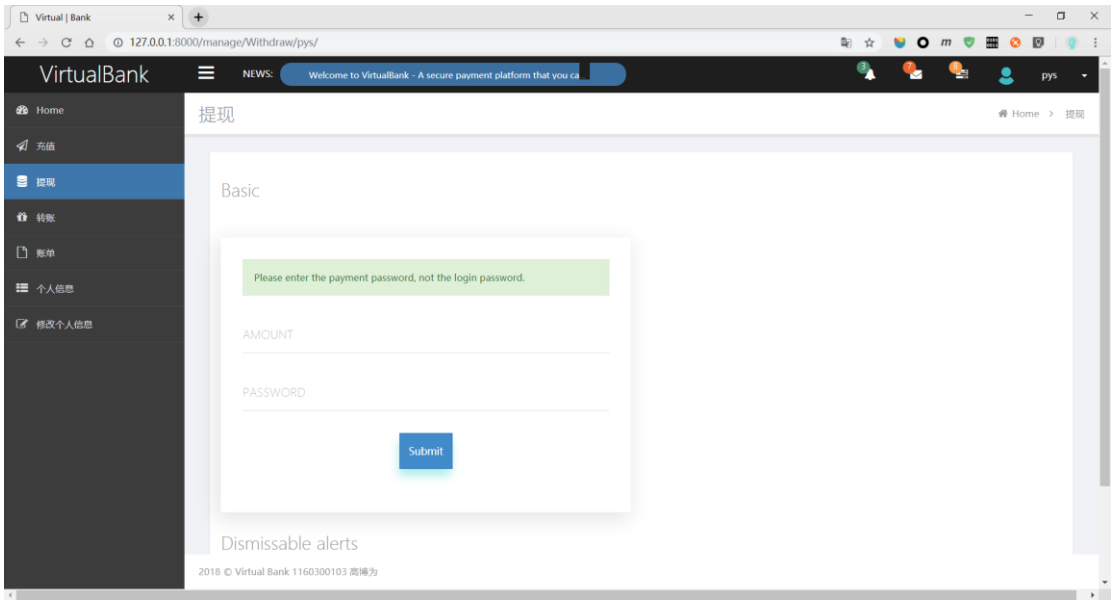
3）充值与提现页面

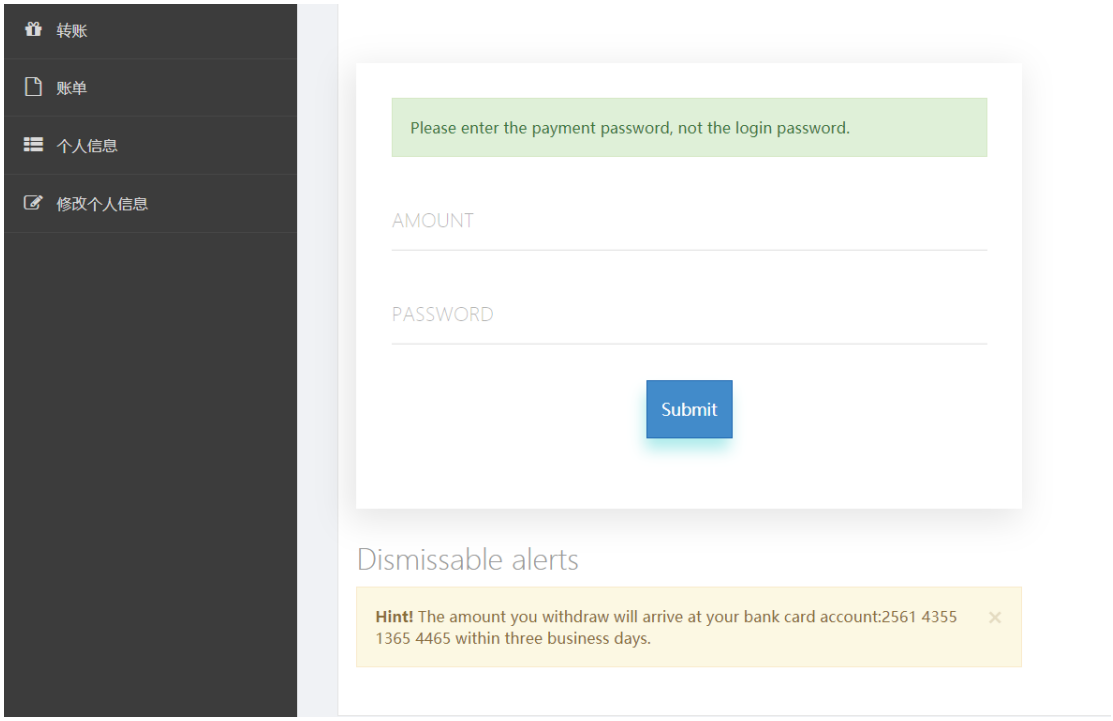
充值页面：




账单
个人信息
修改个人信息
Please enter the payment password, not the login password.
AMOUNT
PASSWORD
Submit
Dismissable alerts
Warning! The amount you recharged will be withdrawn from your bank card account:2561 4355 1365 4465.
2018 © Virtual Bank 1160300103 高博为

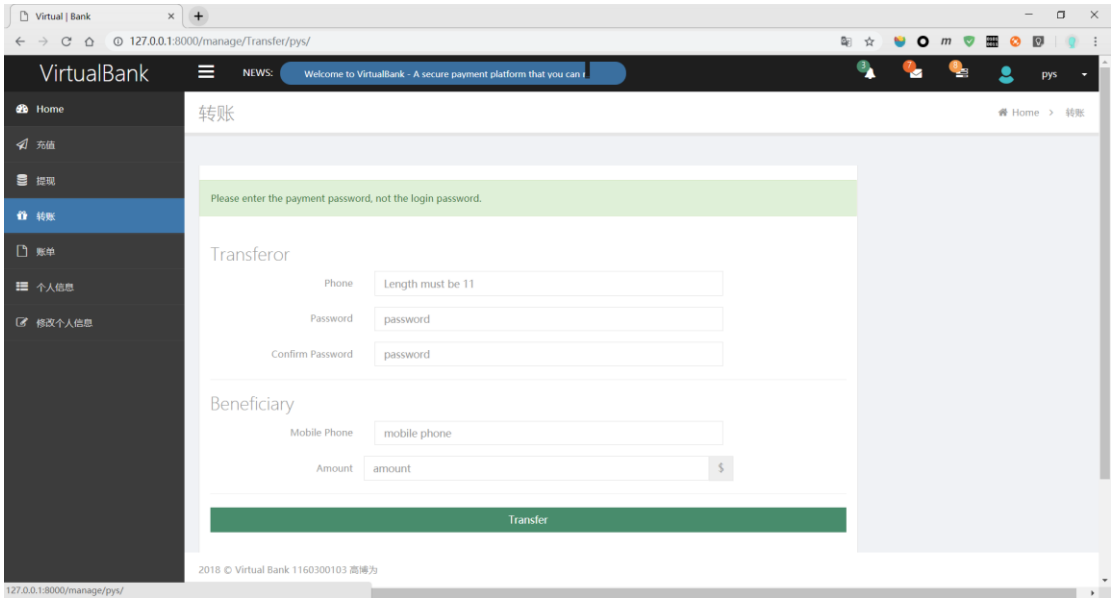
提现页面：

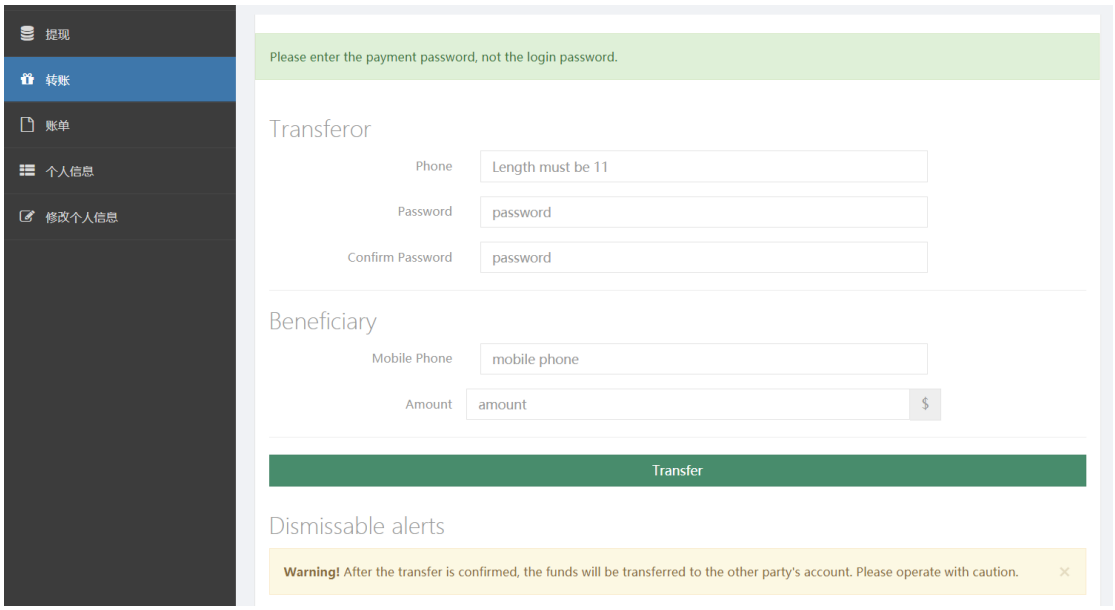

Virtual | Bank
127.0.0.1:8000/manage/Withdraw/pys/
VirtualBank
NEWS:
Welcome to VirtualBank - A secure payment platform that you ca
pys
Home
充值
提现
转账
账单
个人信息
修改个人信息
提现
Home › 提现
Basic
Please enter the payment password, not the login password.
AMOUNT
PASSWORD
Submit
Dismissable alerts
2018 © Virtual Bank 1160300103 高博为



可以看到，充值与提现页面非常类似，页面非常简洁，要输入的信息也十分清晰，同时，底部会有根据用户的卡号渲染的提醒栏，为用户提供必要的提醒信息。

4）转账页面

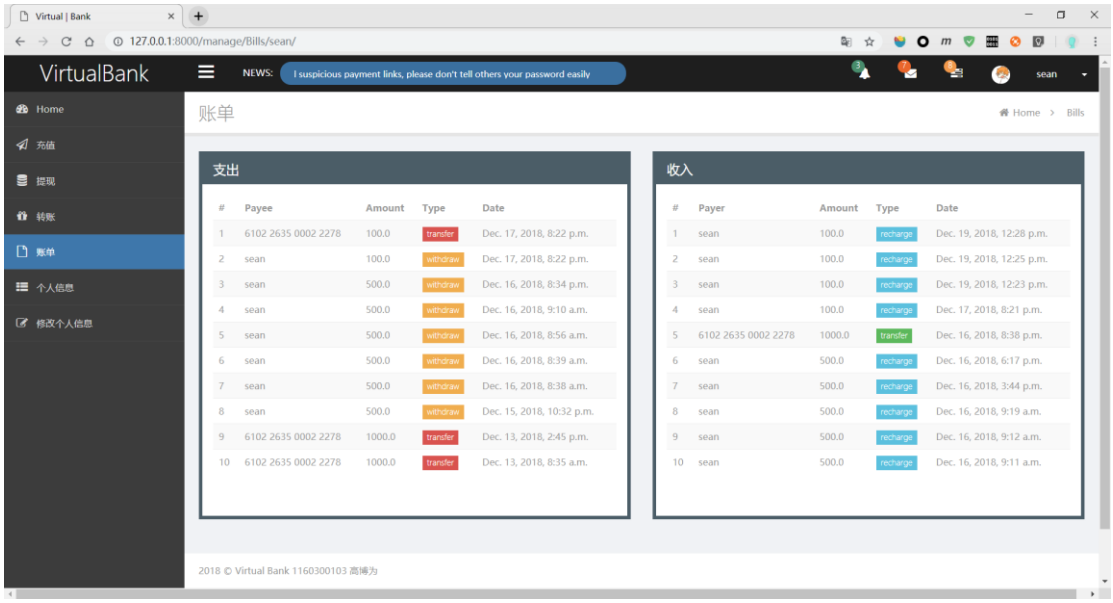


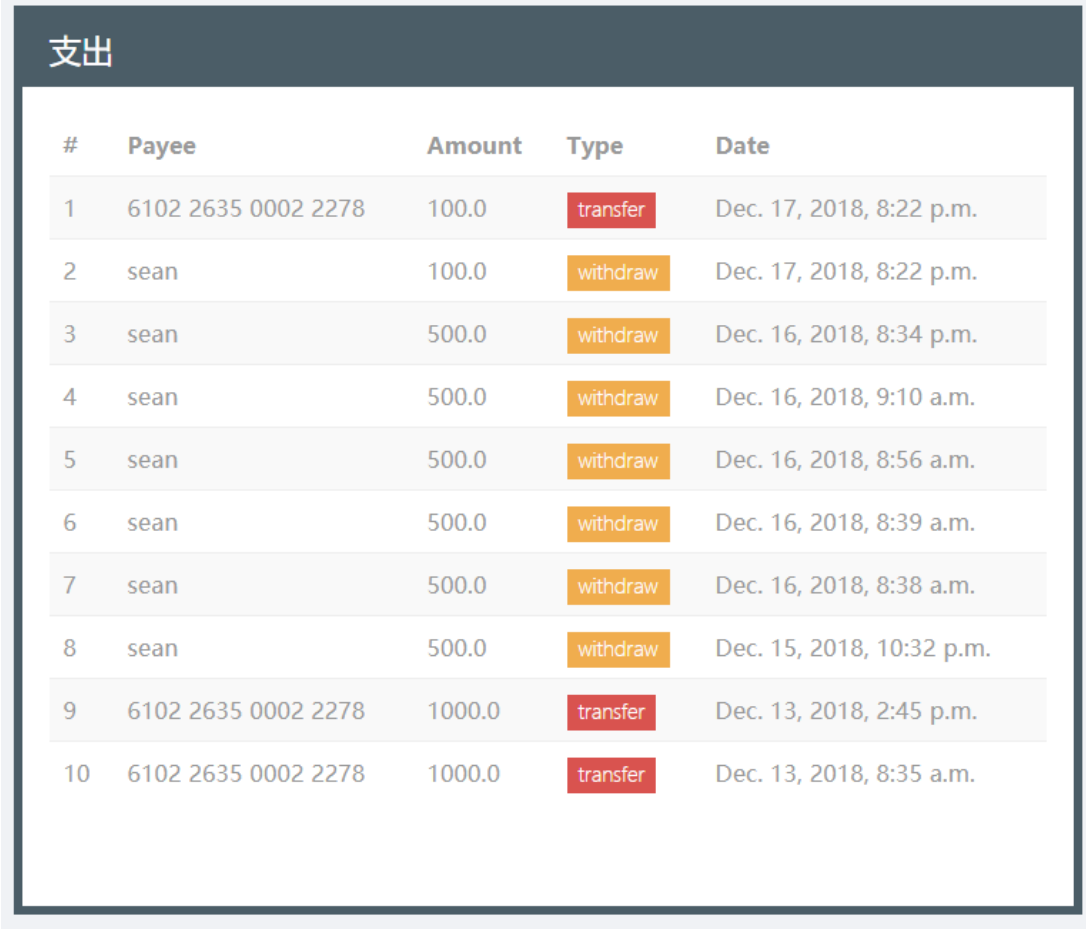


转账页面要提供的信息稍为复杂，包括用户自己的电话密码，还有收款人账户，以及金额。同时也提供了简洁的警告信息。

5）账单页面

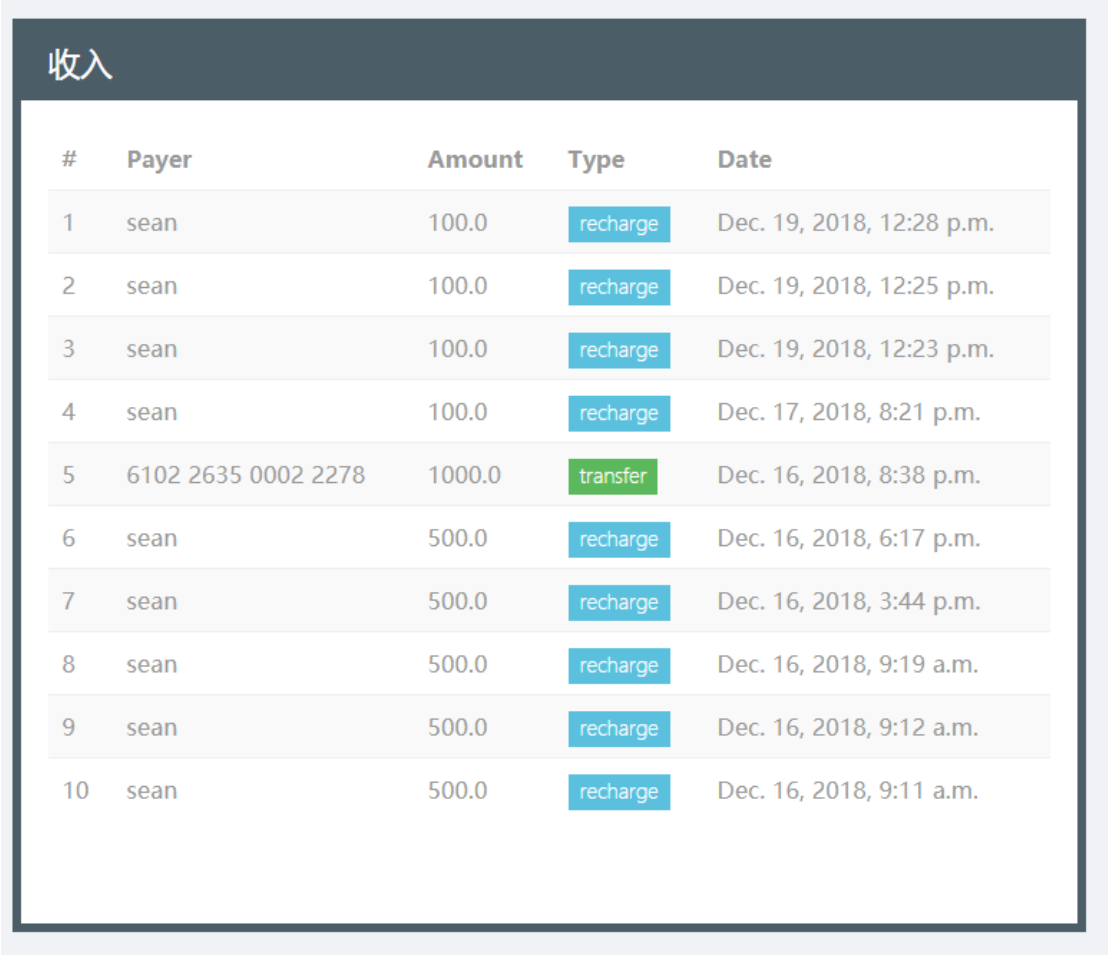
页面中将支出与收入分开显示，并为不同的账单类型设置不同颜色的标签，每条账单信息只显示最为重要的信息，即收款人或支付人，金额，账单类型，交易时间，没有多余的信息，做到了尽量直观与简介的账单显示页



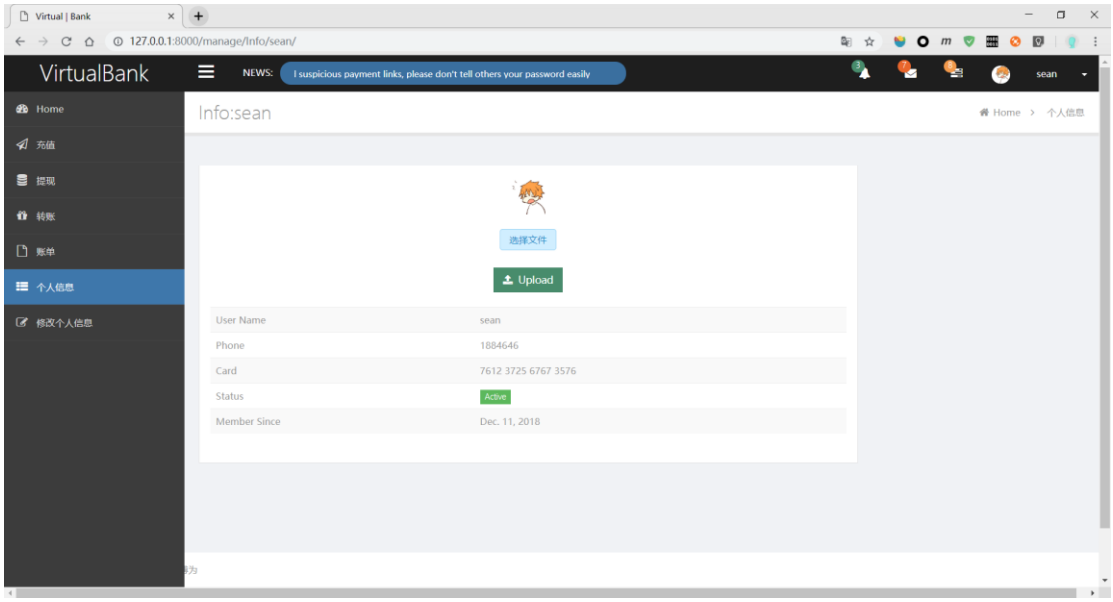


支出

#	Payee	Amount	Type	Date
1	6102 2635 0002 2278	100.0	transfer	Dec. 17, 2018, 8:22 p.m.
2	sean	100.0	withdraw	Dec. 17, 2018, 8:22 p.m.
3	sean	500.0	withdraw	Dec. 16, 2018, 8:34 p.m.
4	sean	500.0	withdraw	Dec. 16, 2018, 9:10 a.m.
5	sean	500.0	withdraw	Dec. 16, 2018, 8:56 a.m.
6	sean	500.0	withdraw	Dec. 16, 2018, 8:39 a.m.
7	sean	500.0	withdraw	Dec. 16, 2018, 8:38 a.m.
8	sean	500.0	withdraw	Dec. 15, 2018, 10:32 p.m.
9	6102 2635 0002 2278	1000.0	transfer	Dec. 13, 2018, 2:45 p.m.
10	6102 2635 0002 2278	1000.0	transfer	Dec. 13, 2018, 8:35 a.m.

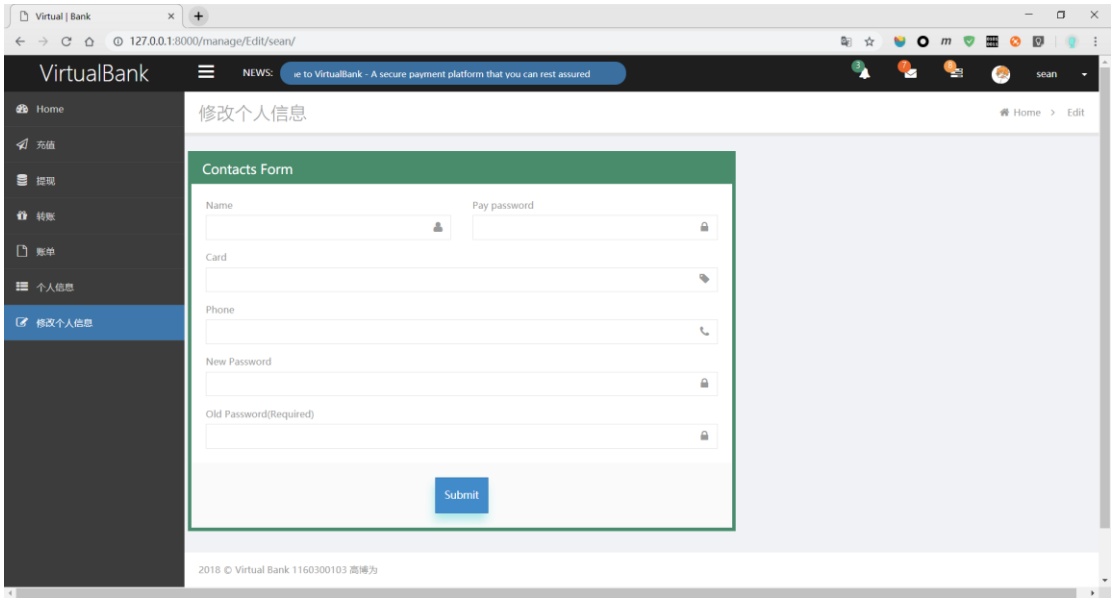


6）个人信息页面：



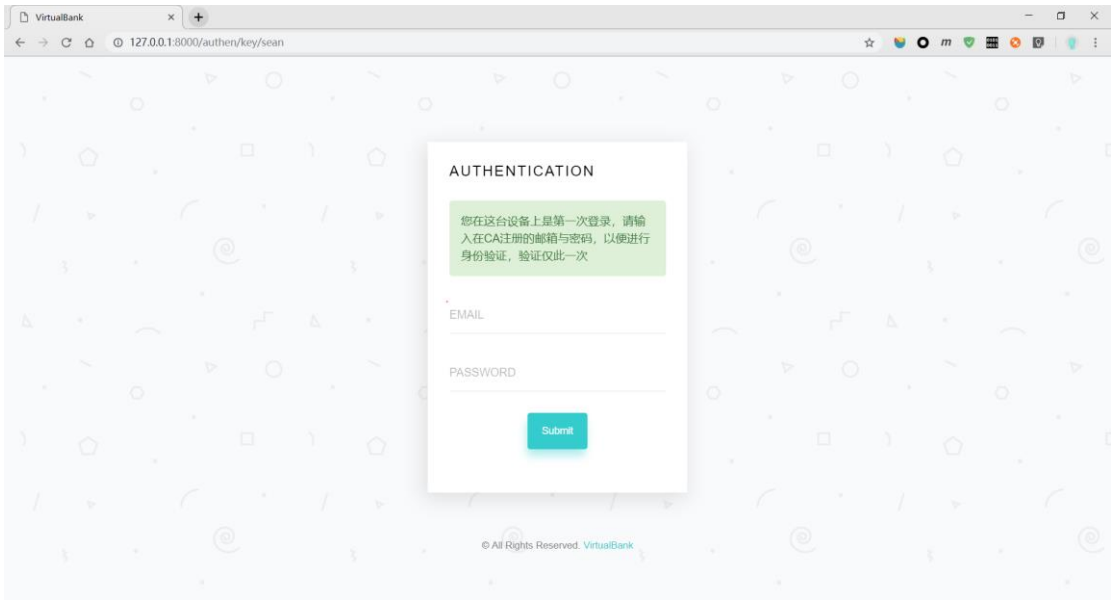
这个页面展示了用户的详细个人信息，例如卡号，预留手机号，账户状态等，还可以在这里上传新的头像，实现个性化定制

7）信息修改界面：



这个页面可以为用户提供修改个人信息的功能，修改时旧的登录密码是必须的。

8）新设备登录验证页面：



在新设备登陆时，若要进行资金转移，必须登录 CA 验证用户身份，以防止密码泄露后造成资金损失。

4.2 详细功能设计

4.2.1 用户功能

用户模块的主要功能有：注册用户、登陆系统、充值、提现、付款、转账等功能。

1）用户注册：用户通过注册成为系统用户并登录验证身份后，可以使用该系统提供的服务。注册时包括用户名、密码等基本信息。

2)用户登陆：用户通过系统的相关验证设计，登陆系统验证身份。在用户登录

时为了保障用户账户的安全性，要求用户首次登陆时需要设定付款密码，付款密码不能和登陆密码相同。同时在首次登录一台设备时，需要登陆一次 CA 进行身份验证，以便将私钥保存在本地

3）充值：用户可以通过银行卡，给自己系统账户充值。

4）提现：用户可以把账户内余额，提取现金到指定银行卡。

5）转账：系统用户之间可以进行系统内转账，输入自身预留电话等，输入对方账号，转账金额和付款密码完成转账。

6）付款：在商家平台购物完成后，跳转至银行提供的支付页面，输入预留手机号和支付密码，完成付款。

7）账户管理：用户可以管理自己的账户，修改个人信息，修改登录密码和付款密码，预留手机号，绑定银行卡等，还可以上传自定义头像

8）账单管理：用户可以查看自己的消费记录，消费记录分为转账、付款、提现和充值四种类型，每种使用不同的标签标识。

4.2.2 管理员功能

主要功能有：登陆、账户及账单等管理。

1）管理员登陆：管理员通过网页端通过验证后完成登陆，登陆系统后可以完成相应的管理操作

2）账户管理：管理员可以查看系统的所有注册用户账号，以及所有数据库内信息，方便系统管理。

4.3 安全性保障详细设计

4.3.1 注册安全性

1）用户向服务器发起注册请求，服务器收到注册请求信息之后，将自身 RSA 公钥 public_key 发送给用户（即前端）。

2）用户在前端将 预留手机号、密码和用户个人信息等用公钥 public_key 加密后，将结果向虚拟银行后台发送。

3） 认证中心收到消息后，使用私钥 private_key 解密，然后使用散列函数 MD5 对用户密码进行散列，得到 h(Pwd)，将 h(Pwd)作为密码字段，与用户信息一起存入安全数据库。

5）认证中心将用户信息录入安全数据库之后，向用户发送注册成功的消息。并将跳转至登陆页面。

以上方式进行的同时进行了输入检查，防止非法字符输入，防止空字段被发送：

```
var pub_key;
if ($('#name').val() == "") {
    alert("请填写用户名");
    name.focus();
    return false;
}
if (phone.value == "") {
    alert("请填写联系电话");
    phone.focus();
    return false;
}
if (id.value == "") {
    alert("请填写身份证号码");
    id.focus();
    return false;
}
if (card.value == "") {
    alert("请填写银行卡号");
    card.focus();
    return false;
}
if (passwd.value == "" || re_passwd.value == "") {
    alert("请输入密码");
    passwd.focus();
    return false;
}
if (passwd.value != re_passwd.value) {
    alert("两次密码不一致");
    $('#password').val("");
```

4.3.2 登录安全性

登陆界面会检查用户输入的用户名、密码是否包含非法字符，然后使用 RSA 加密算法，以银行公钥 public key 作为密钥加密所有要发送的字段，银行后台收到后，以自身私钥解密密文，然后将用户密码哈希后和数据库的哈希值进行比对，如果相同则登陆成功，服务器在 session 中将用户登录状态设置为 True，拒绝同时登录多次，只有当浏览器关闭或用户执行注销操作或 session 有效期到期才会清除，在这之前下次交互时凭借 cookie 进行身份验证。

另外还有防止跨站点伪造请求的处理，在 html 的表单前加入 Django 自带的 csrf 标签如下：

```
<!-- Start Sign In Form -->
<form action="#" class="fh5co-form animate-box" data-animate-effect="fadeInRight"
    {% csrf_token %}
    <h2>Sign In</h2>
```

该标签工作原理如下：

1 django 第一次响应来自某个客户端的请求时，会在服务器端随机生成一个 token，把这个 token 放在 cookie 里。然后每次 POST 请求都会带上这个 token，这样就能避免被 CSRF 攻击。
2 在返回的 HTTP 响应的 cookie 里，django 会为你添加一个 csrftoken 字段，其值为一个自动生成的 token，在所有的 POST 表单时，必须包含一个 csrfmiddlewaretoken 字段 （只需要在模板里加一个 tag， django 就会自动帮你生成，见下面）
3 在处理 POST 请求之前，django 会验证这个请求的 cookie 里的 csrftoken 字段的值和提交的表单里的 csrfmiddlewaretoken 字段的值是否一样。如果一样，则表明这是一个合法的请求，否则，这个请求可能是来自于别人的 csrf 攻击，返回 403 Forbidden.

加盐的防重放攻击：

另外加密之前，在获取银行公钥时，服务器会产生一个随机的一次性盐，将其与公钥一起发送给用户，用户收到后，将密码哈希后与盐连接再次哈希，然后加密发送，服务器比对时会做相应处理。这样做到了防重放攻击。

```
[salt_id, salt] = set_salt(request)
return JsonResponse({"pub_key": get_rsa_pubkey(), "salt": salt, "salt_id": salt_id})
```

4.3.3 数据传输安全性

机密性：

在网站传输时采用 RSA 方式加密。用户用服务器的公钥 Public Key 加密本次通信需要传输的内容。服务器端利用私钥 Private Key 对信息解密。

完整性：

为了防止信息被他人篡改，将消息加密后，对消息密文 Cipher 生成消息摘要 MAC。服务器接受到信息后将信息生成的摘要和 MAC 对比，如果一致则没有被篡改。

身份验证：

每次发送数据时，尤其是资金操作时，对完整性中生成的 MAC 使用操作的用户的私钥进行签名，服务器得到后，向 CA 请求该用户的证书，对证书验证无误后，取出用户公钥，对 RSA 签名进行验证，若验证无误，则身份验证通过，允许操作。

防重放攻击：

所有有关账户资金的转移提取操作进行数据通信前，在加密之前，获取银行公钥时，服务器会产生一个随机的一次性盐，这个盐用后即废，将其与公钥一起发送给用户，用户收到后，将密码哈希后与盐连接再次哈希，然后加密发送，服务器比对时会做相应处理。这样做到了防重放攻击。

4.3.4 支付接口安全性

机密性：

采取数字信封技术

与电商平台进行数据交互时，采取 RSA 与 AES 混合的数据加密方法，电商平台方面跳转至银行支付网关时，用服务器的公钥 Pub 加密本次通信使

用的 AES 对称密钥 Key,然后将需要传输的内容用对称密钥进行 AES 加密。模式采用 CBC 模式，初始向量 IV 使用与 KEY 相同的字节流，服务器端利用私钥 Private Key 解密得到 Key，利用 Key 对信息解密。

完整性验证：

数字签名，双签名技术

商家对敏感信息，比如金额和收款方卡号加密后的密文生成消息摘要 MAC，使用商家私钥进行签名生成 Signature，并将自身证书与密文和摘要一起发送，银行服务器收到后，首先验证证书，若验证无误，则验证数字签名，若验证无误，跳转至支付页面，生成 PI，对 PI 生成 HashPI，将 HashPI 以协商好的 AES 密钥进行加密，发送给商家平台，生成双签名，并对双签名进行验证（用得到的签名用户的公钥进行验证），若验证成功，则进行转账操作，否则拒绝操作。

身份验证：

使用数字证书和数字签名、双签名技术

与商家交互的每一步都需要进行数字证书的交换与验证

还需要进行敏感信息的数字签名与验证

此外，还要与商家和 CA 认证中心协作进行双签名的生成与验证

4.4 数据库详细设计

项目中一共包含四个数据库表单，分别是：

User：存储用户的个人信息的表单，包含预留手机号等信息：

User 表

字段	类型	说明
Name	CharField	用户名
Id_no	CharField	身份证号码
Card	CharField	绑定银行卡号
Phone	CharField	预留手机号
Passwd	CharField	密码
Pay_paswd	CharField	支付密码
Pub_key	FilePathField	用户公钥存贮路径

Account：存储在银行注册的账户的相关信息：

Account 表

字段	类型	说明
User	CharField	用户的预留手机号
Avatar	ImageField	用户的头像存储路径
Balance	FloatField	账户余额
Cost	FloatField	账户支出总额
Regtime	DateField	注册时间

PayBill：存储用户在商家处支付时的信息，辅助完成支付：

PayBill 表

字段	类型	说明
Amount	CharField	支付金额
Card	CharField	收款人卡号
Key	CharField	此次协商的 AES 密钥
Deal_identify	CharField	此次交易的订单号（商家需要）
Pay_id	CharField	此次交易的支付标识符
Hash_pi	CharField	支付信息的哈希值
Payer_name	CharField	付款人用户名

Bills：存储所有交易账单的数据表单，包括提现，转账在内的四种账单类型：

Bills 表

字段	类型	说明
payer	CharField	付款人的用户名
Payer_card	CharField	付款人卡号
Beneficiary	CharField	收款人卡号
Amount	FloatField	交易金额
Bill_type	CharField	交易类型
Date	DateTimeField	交易日期

5.实现与测试

5.1 实现

服务器基于 Django 框架实现
客户端基于 Web 实现

服务器端实现所用库：

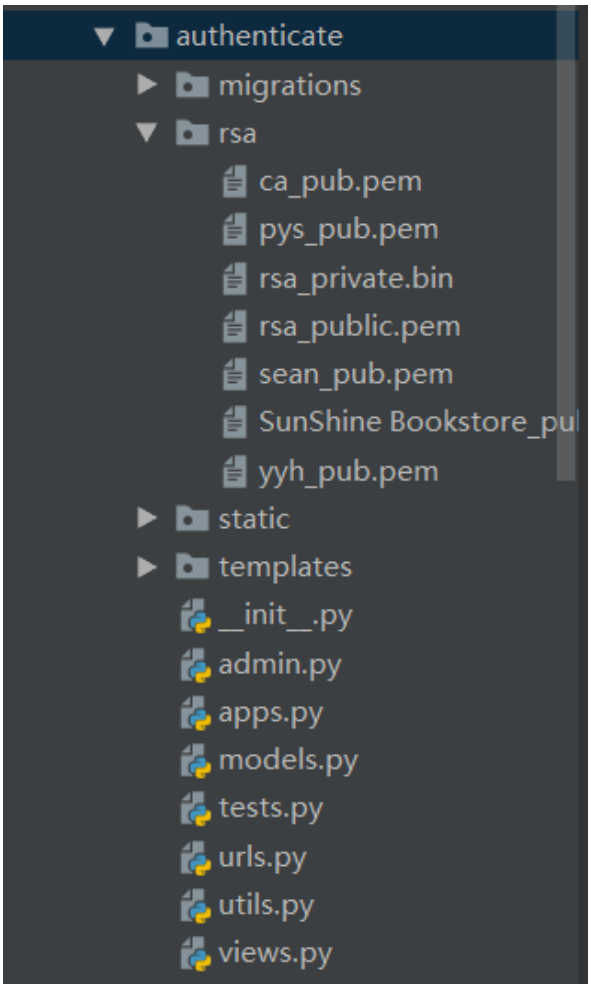
1．pycryptodome
　　这个库包含了本次实验中所用的所有加密算法和签名验签算法的实现
2. hashlib
　　系统自带库，实现了大量哈希算法，本次实验中使用了 MD5 和 SHA256

服务器实现的 app:

在这个 Django 项目中，注册并实现了两个 app，分别是：

Authenticate：

　　这个 app 实现了用户的认证模块，包括登录，注册，设置支付密码，身份认证，支付接口等，结构如下：



其中 rsa 文件夹中包含所有交互过程中所使用的密钥，包括服务器的公私钥对和 CA，商家以及用户的公钥。Static 文件夹中存放所有这些功能所需要的静态文件，即 css,javascript,以及图片等，templates 文件夹中则包含所有要用到的 html 模板文件。Views 文件中实现了所有的处理逻辑函数，models 文件中建立了项目要用到的四个数据库表单中的三个：

---

class User(models.Model)

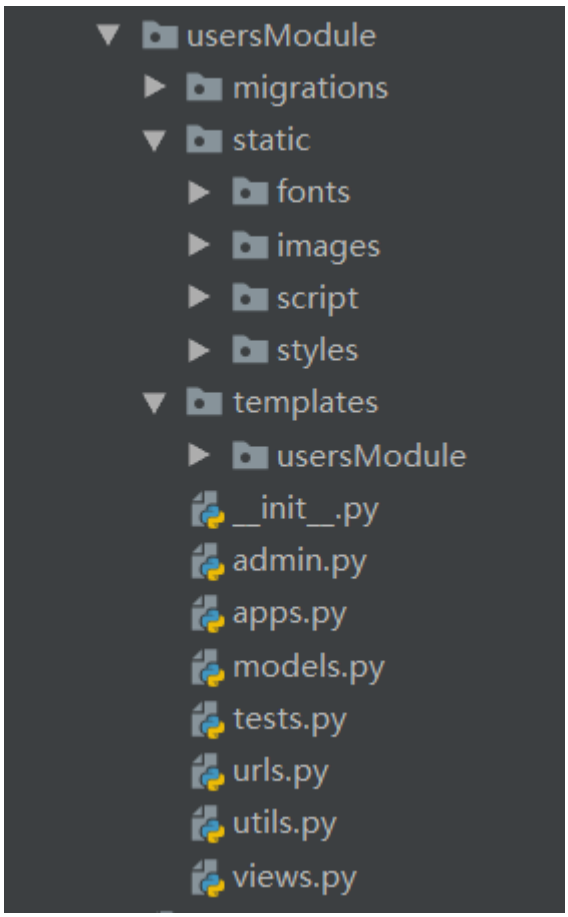
class Account(models.Model)

class PayBill(models.Model)

---

usersModule：

这个 app 实现了登陆后的用户模块，包括所有用户要用到的资金存取，转账，账单查看，修改信息等模块，是用户模块的主体，结构目录如下：



Static 文件夹中存放只这个 app 中用到的静态文件，templates 中存放此 app 内要用的静态 html 模板文件，views 中实现了处理逻辑代码，在 models 中创建了项目中剩下的一个表单：Bills

```
class Bills(models.Model)
```

还有一些特殊的文件与目录如下：
Static：项目中在两个 app 中都需要使用的静态文件，包括 js,css 等
Media：项目中用于存放用户头像的文件夹

OnlineBank/utils：我创建的一个全局工具函数，其中包含大量在项目中会重复用到的代码，将其独立封装于一些函数中，例如 aes 加密，aes 解密，rsa 加解密，md5 哈希等等，在项目中作用巨大，不仅省了很多重复工作，也使得调试更为简单

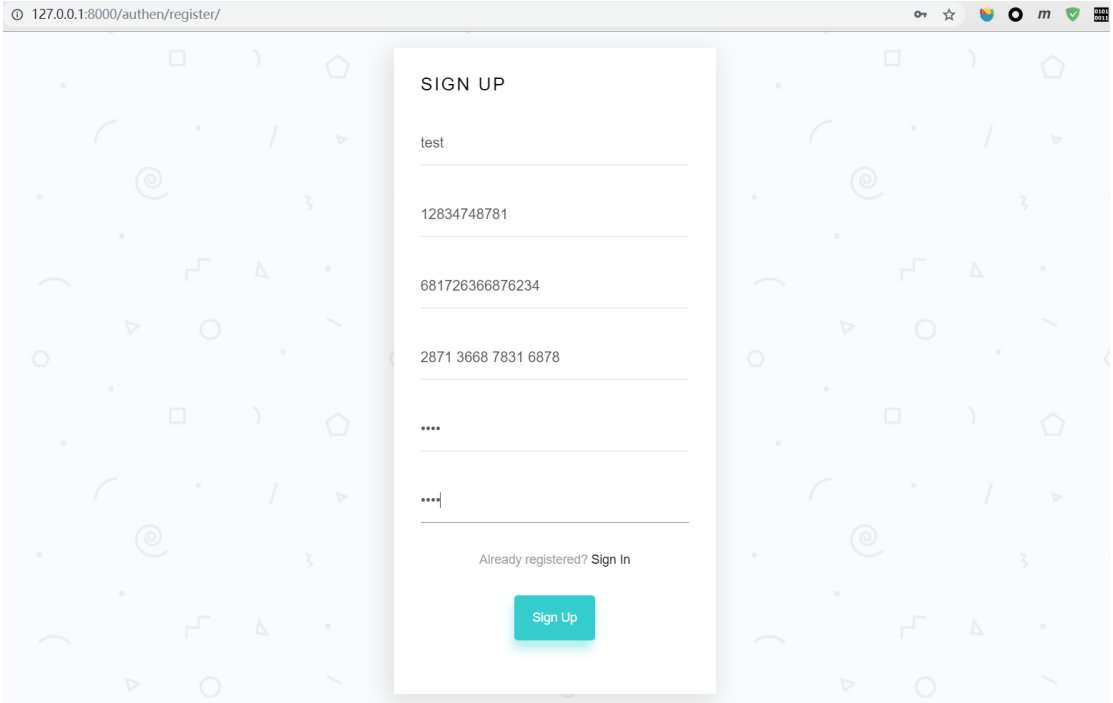
OnlineBank/config：我创建的全局配置文件，在其中创建了一个 Config 类，并写入了很多在代码中会重复用到或是容易改变的值，将他们集中于一个文件中可以避免改动时引起的各种错误，需要改动时也更为简单：

```
class Config:
    CA_Host = "http://192.168.43.59:8000/"  # Ca certification center host
    CA_GetCert = CA_Host+"ca/require/"  # certification require host
    CA_requir = CA_Host+"ca/Require_prik/"
    CA_register = CA_Host+"ca/trader_register/"

    Plat_Host = "http://192.168.43.160:8000/"  # E-commerce platform host
    Plat_PayHost = Plat_Host+"bank_receipt/pi"
    Plat_name = "SunShine Bookstore"  # E-commerce platform's name in CA

    User_Agent = 'Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 ' \
                 '(KHTML, like Gecko) Chrome/34.0.1847.137 Safari/537.36 LBBROWSER'  # Post headers

    key_url = "C:\\Users\omnitrix\PycharmProjects\\virtualBank\onlineBank\\authenticate\\rsa\\"
    # rsa private keys and public keys URL

    max_num = 10  # Bills page max num

    max_saltId = 20  # max number of random salt id
    salt_Length = 8  # salt length

    max_payId = 10000
    min_payId = 0
```

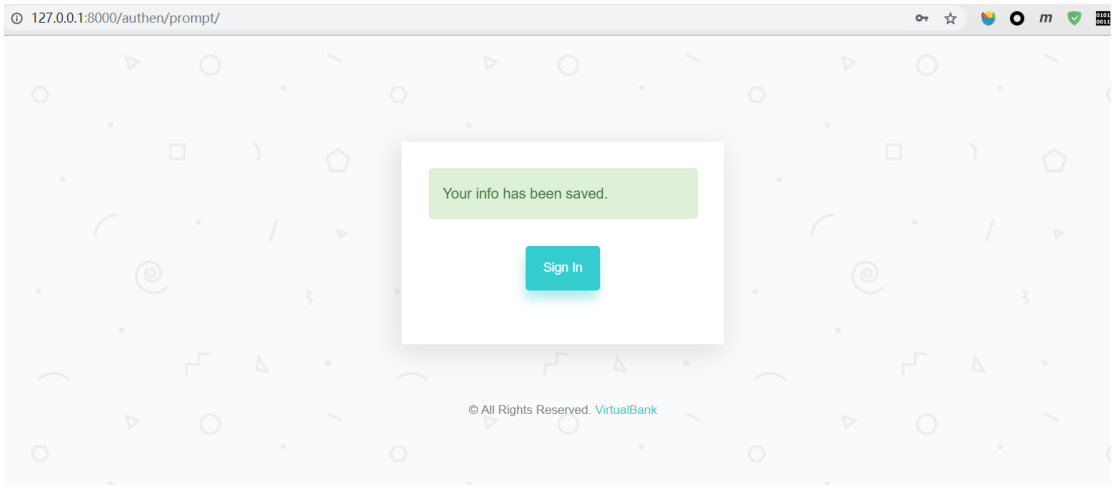
5.2 测试

5.2.1 注册测试：

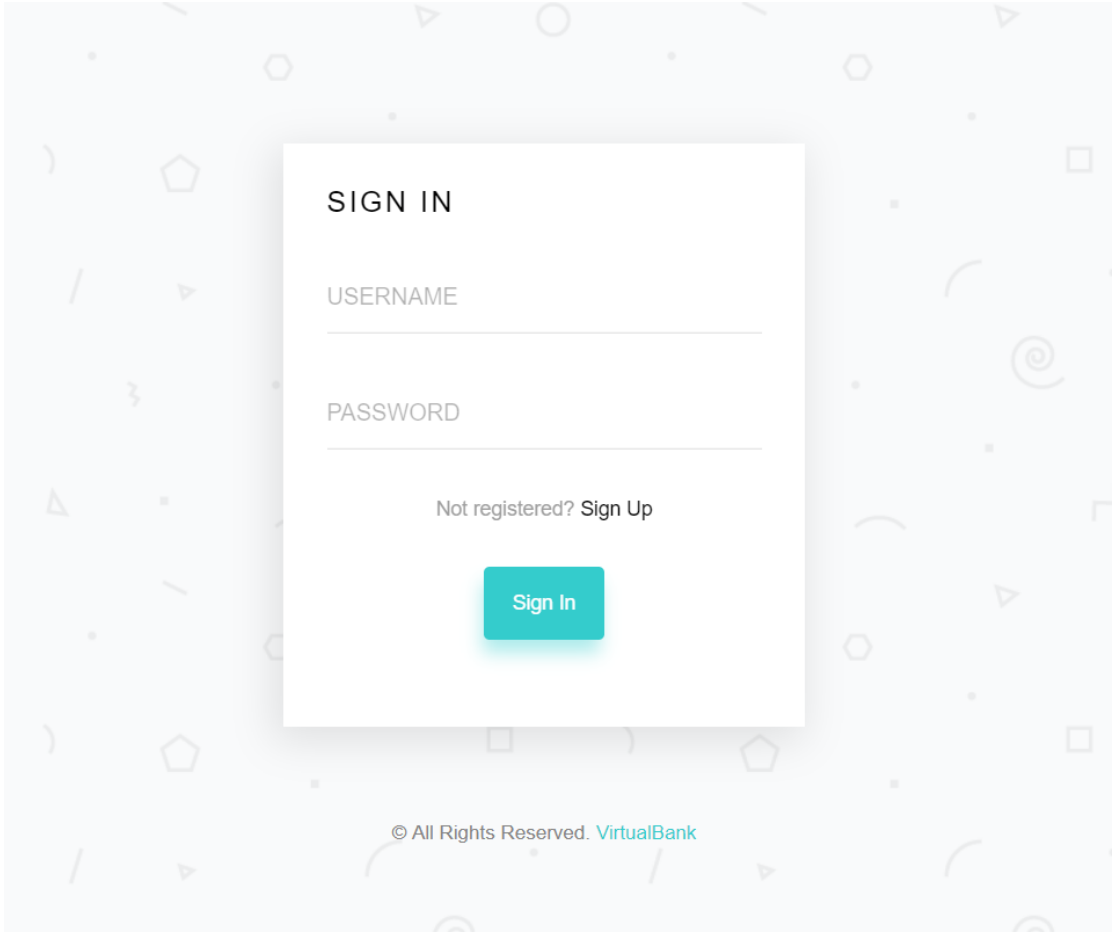
进入注册页面，并输入注册所需信息：



点击注册：

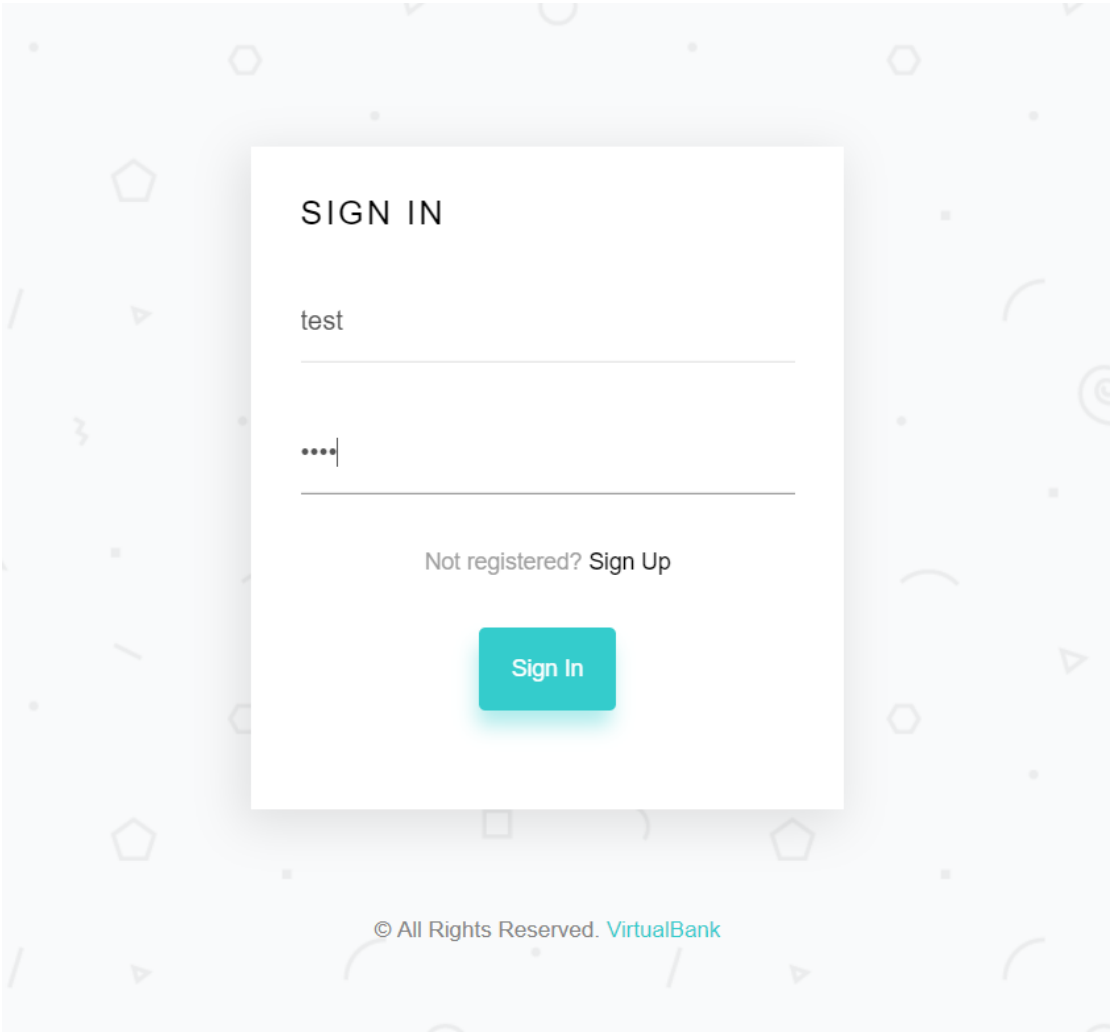


跳转至提示页面
点击登录则跳转至登陆页面：

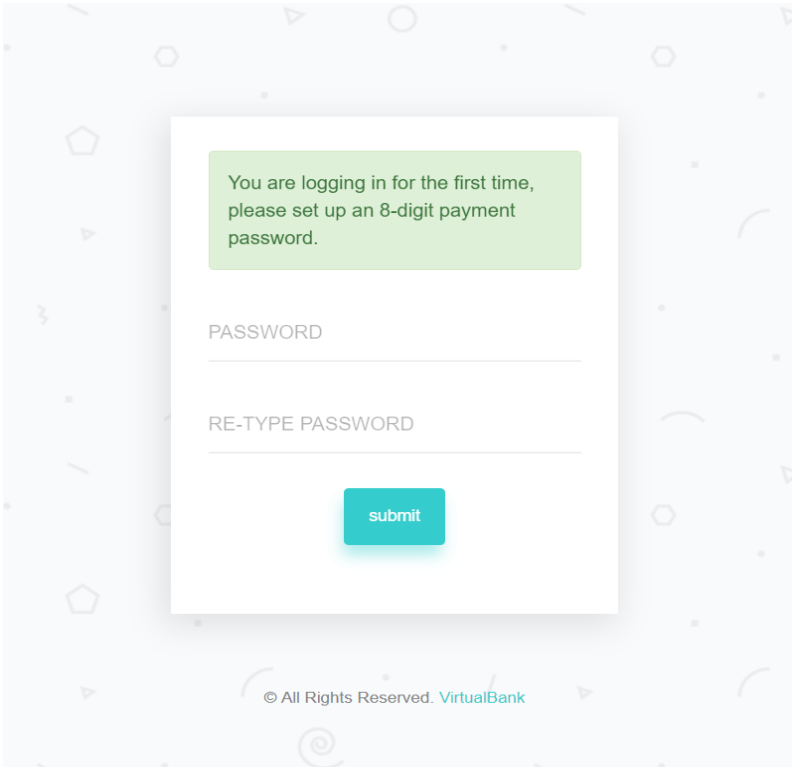


5.2.2 登录测试：

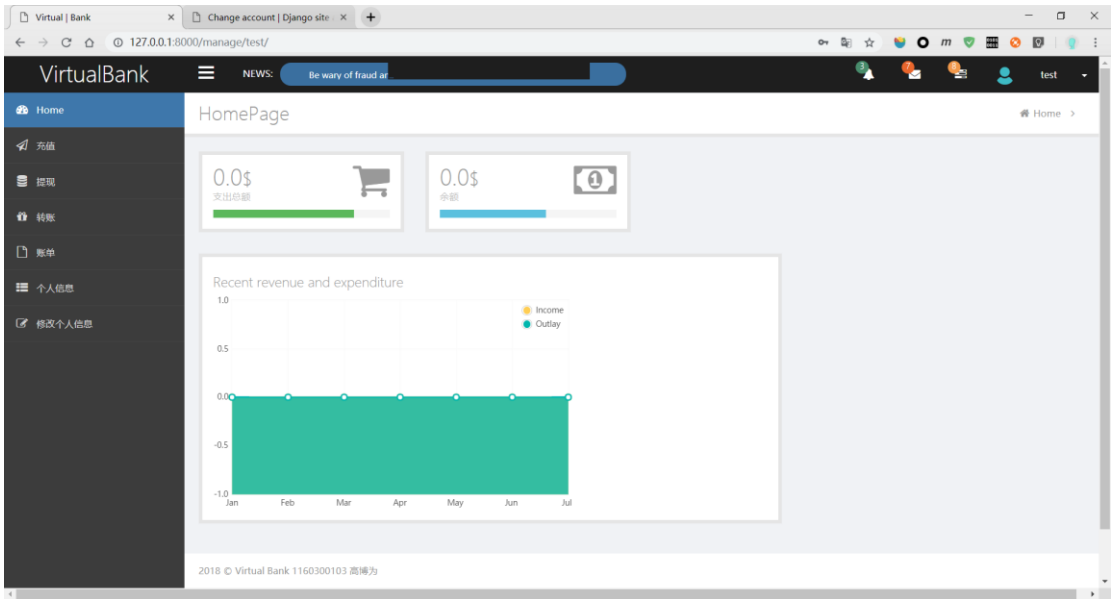
打开登录页面，输入登陆信息：


SIGN IN
test
Not registered? Sign Up
Sign In
© All Rights Reserved. VirtualBank

点击登录：


You are logging in for the first time, please set up an 8-digit payment password.
PASSWORD
RE-TYPE PASSWORD
submit
© All Rights Reserved. VirtualBank

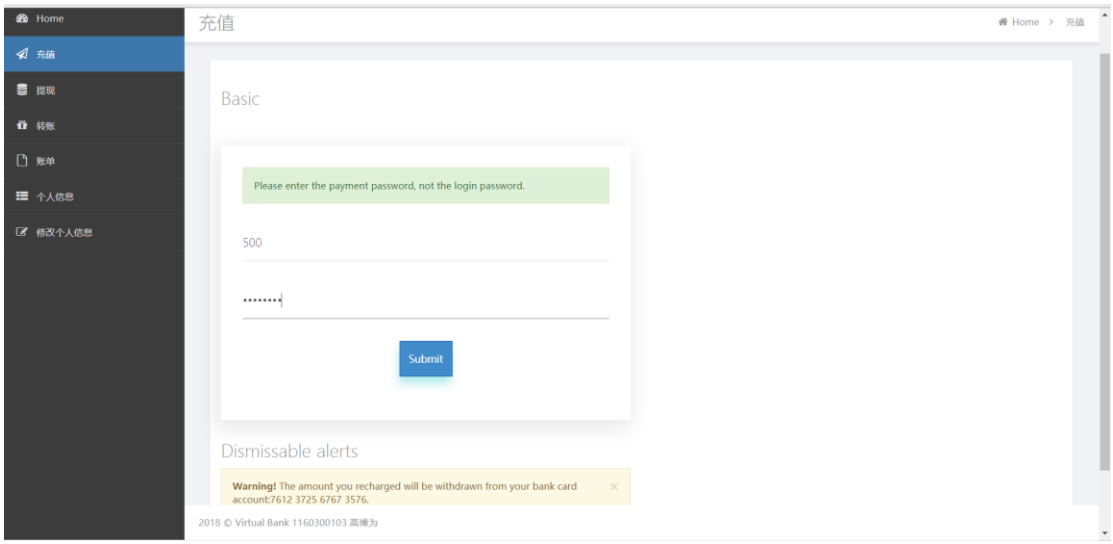
因为是第一次登录该账号，所以跳转至设置支付密码页面。
输入支付密码，点击提交：



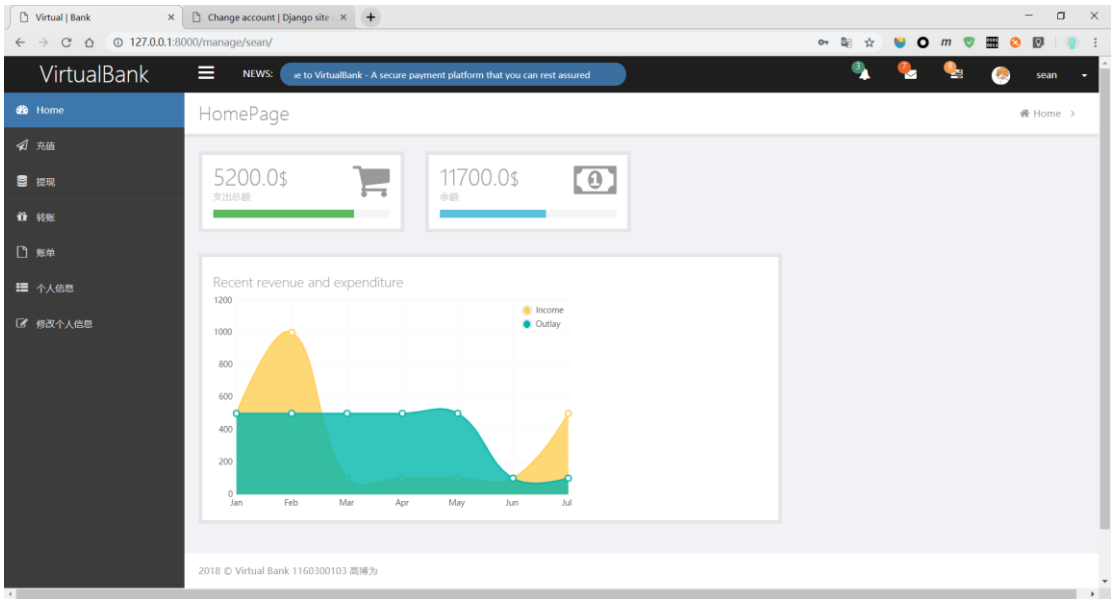
成功跳转至用户主页。

5.2.3 充值测试：

进入充值页面，输入金额和支付密码：

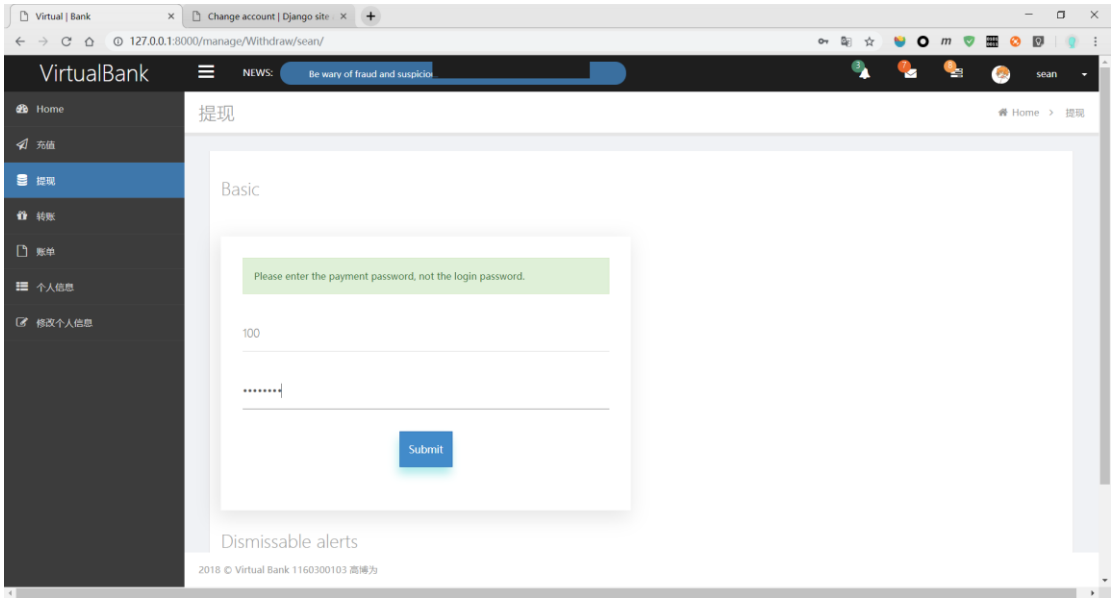


充值成功后显示提示信息，自动跳转至主页：

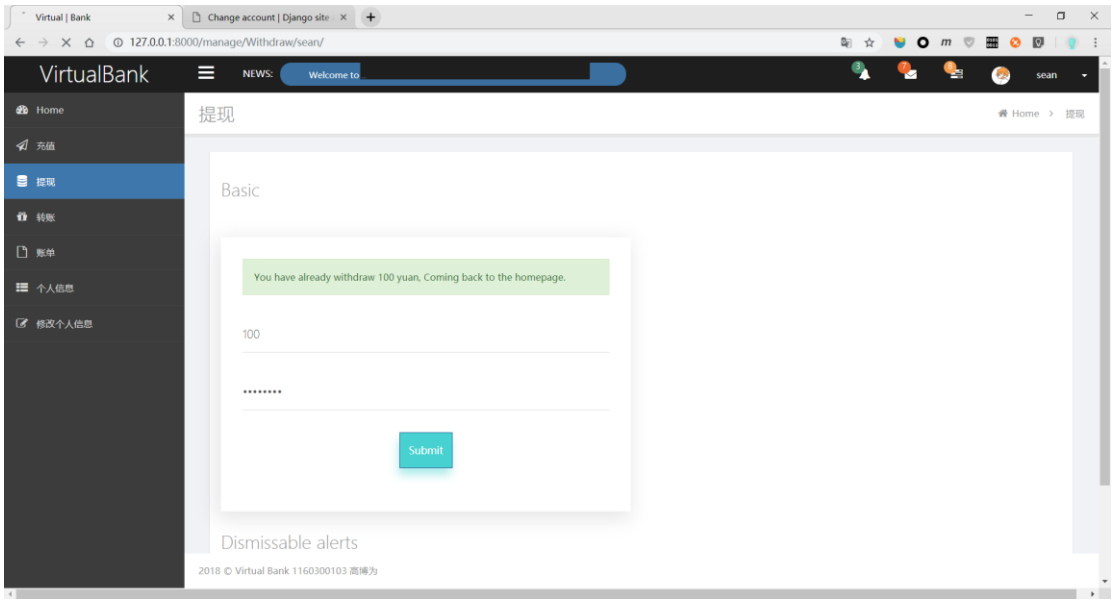


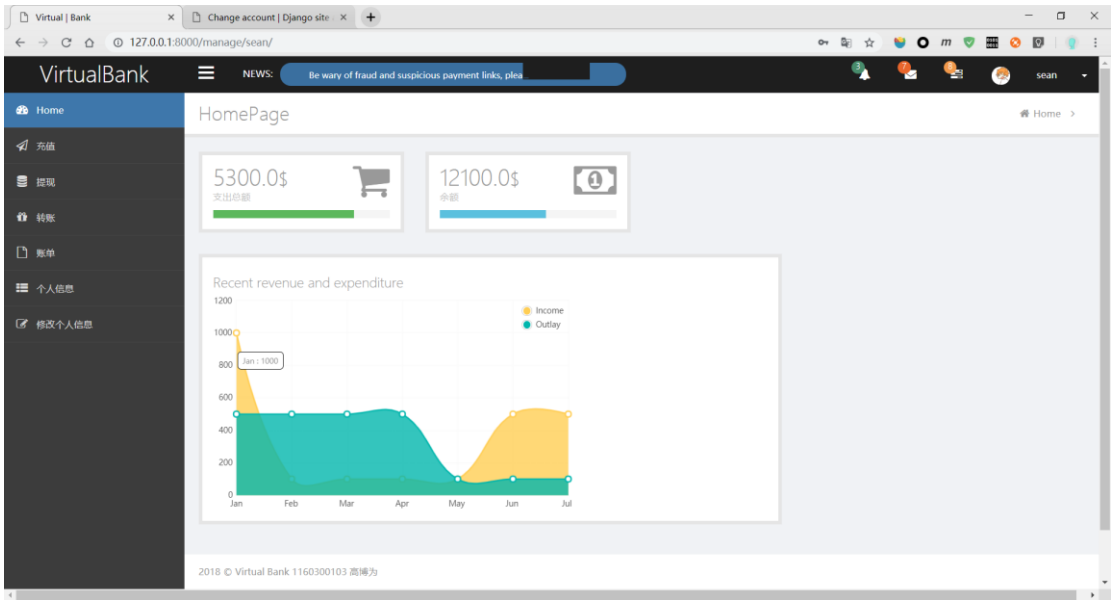
5.2.4 提现测试

进入提现页面，输入必要信息：



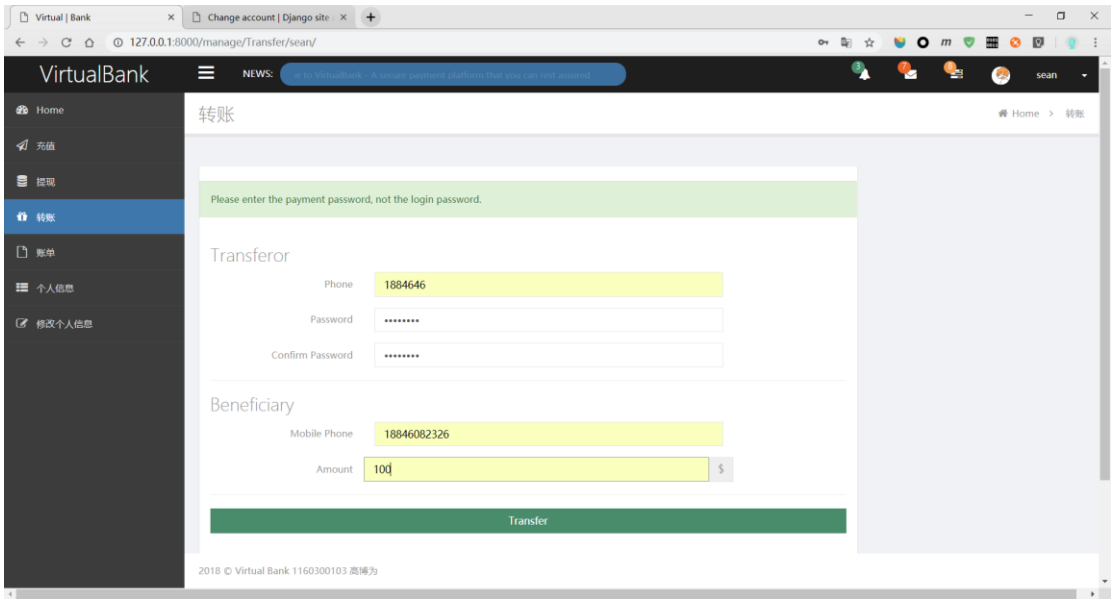
点击提交，出现成功的提示信息，并跳转至主页：



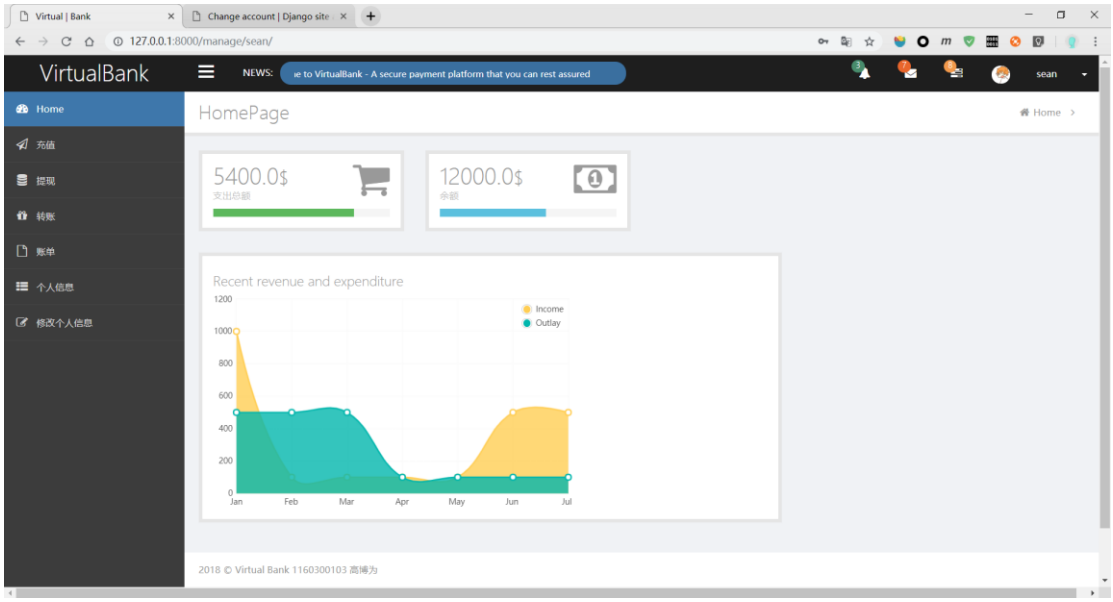


5.2.5 转账测试

进入转账页面，输入信息：
这里以转至“高博为”账户为例：

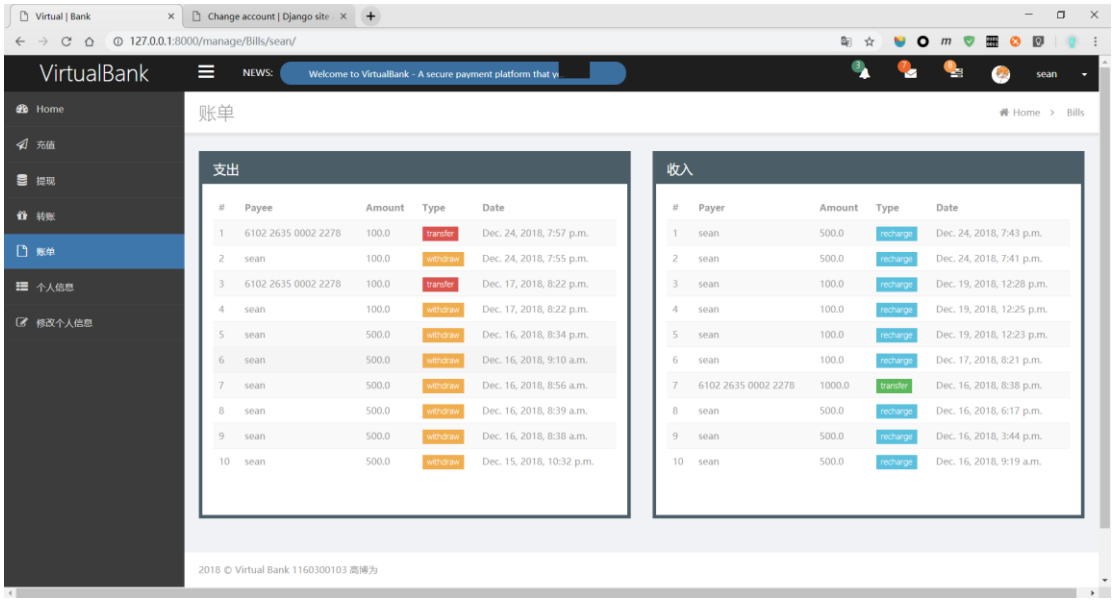

Virtual | Bank
Change account | Django site
127.0.0.1:8000/manage/Transfer/sean/
VirtualBank
NEWS:
sean
Home
充值
提现
转账
账单
个人信息
修改个人信息
转账
Home › 转账
Please enter the payment password, not the login password.
Transferor
Phone
1884646
Password
Confirm Password
Beneficiary
Mobile Phone
18846082326
Amount
100
$
Transfer
2018 © Virtual Bank 1160300103 高博为

点击转账：


Virtual | Bank
Change account | Django site
127.0.0.1:8000/manage/sean/
VirtualBank
NEWS:
sean
Home
充值
提现
转账
账单
个人信息
修改个人信息
HomePage
Home ›
5400.0$
支出总额
12000.0$
余额
Recent revenue and expenditure
Income
Outlay
1200
1000
800
600
400
200
0
Jan
Feb
Mar
Apr
May
Jun
Jul
2018 © Virtual Bank 1160300103 高博为

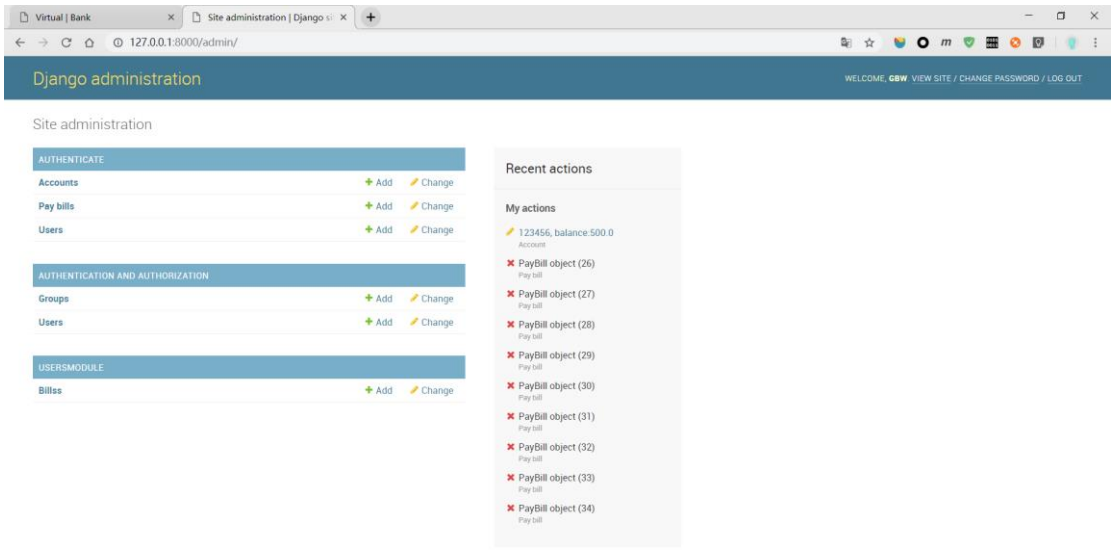
转账成功

5.2.6 账单测试



成功
但这个页面的加载速度会有些慢，是由于查询数据量相对较大引起的。

5.2.7 管理员功能测试



可以看到所有数据表单
也可以对每个表单中的具体数据进行管理

6. 结束语

上个学期，也就是大二年级的第二学期，我们曾初步接触过信息安全的广阔世界，一个学期的的信息安全导论为我们介绍了很多安全方面的知识，而如今在

密码学原理与实践这门课程的学习中，我们有机会将当初还一知半解的知识与新学到的密码学知识相结合，并将其应用在实践项目中，我感到很兴奋，也很有成就感。在本次课程中，安全协议上我对混合加密、公私钥签名认证、双签名技术有了更深刻的理解。而在编程上，我将曾经只是浅尝辄止的 Web 开发知识重新拿出来使用，并在这个过程中受益匪浅。 在整个项目的开发中，从最初的雏形到现在的成品，从开始的单机操作和加解密，到与同组的两位同学协作工作，我完全不是一帆风顺的，可以说遇到了极多困难，但在面对这些困难时自己能主动思考，主动查阅资料，去 Google 遇到的问题，最终在一番波折后解决了一个又一个困难，我感到十分有成就感，也获得了很大的满足，与组员的讨论互动过程也使得我们对这个项目和流程有了弥足深刻的理解。

同时非常感谢翟老师的指导和帮助，为我们的实验起到了极大的促进作用。

参考文献

【1】 翟建宏，信息安全导论，北京：科学出版社，2011.7
【2】 石亦欣，NFC 芯片与 SIM 卡连接的方案研究，复旦大学，2009
【3】 褚宝增，尹立杰，段岩，加密算法在 PKI 体系中的应用，计算机与网络，2012