

# 区块链课程期末Project -- Ether-Idle 项目报告

---

姓名：陈明亮

学号：16340023

(Github代码仓库) (<https://github.com/Palette25/Ether-Idle>)

## 一、项目简介

1. Ether-Idle 是一款基于区块链的线上交易所，中文名为 以太闲置物交易所，是一款面向所有用户的 Web 应用。本项目利用区块链的去中心化特性，可以是用户的购买记录，个人信息等隐私存储在区块链节点上，避免了某些中心化应用存在的隐私泄露问题，同时也解决了用户对闲置物品的买卖需求。
2. 在 Ether-Idle 中，用户通过特定的标识地址，须注册其 EI 账号，以进行系统上的其余交易行为。每个用户都可以发布自己拥有的一些闲置物品拍卖信息(物品名，物品描述，额外信息)，价格单位为 EI Coin，供给有需要的买家进行购买。买方则需要花费相应的 EI Coin 进行物品的购买，当余额不足时则可以通过交易所内部的兑换功能，进行从以太币 Ether 到 EI Coin 的兑换。
3. 每当交易所内部有一项物品被买家拍卖下来，买卖双方双方的相应的 EI 账户余额并不会马上更改，而将所有权交由系统管理层，监视该笔交易的正确进行，并且仅提供买家确认货物收到的权限，即对 Ether-Idle 内部交易的 Confirm 权限。当买家在应用上确认物品收到时，交易所才会真正将这笔交易存储到区块链中，同时更改双方的账户余额，确保交易的合法性与公正性。
4. Ether-Idle 支持用户对其当前的交易记录的查看，包括 Pending 和 Confirmed 两种状态的交易历史记录。其中 Pending 类型的交易记录包括当前用户作为买家与卖家双方身份的交易，但系统的权限管理层只对当前用户开放其作为买家的交易确认权利，确保运行正常。Confirmed 记录则包含了其参与的每一项已完成的交易，该功能使用户能够正常使用交易所，提升用户友好度。

## 二、项目背景依据

1. 本应用的主体用户应该是校园内部群体，亦或是其余一定小范围区域内部的群体。Ether-Idle 相对于当前市面上的同类型应用，有用户数据隐蔽性，以及交易的可靠性等优点，交易不需要泄漏真实地址(控制真实地址数据在双方完成交易时才可见)，所有的双方信息都经过传输前加密，确保用户隐私安全。
2. 闲置物的定义其实不仅限于真实物品，卖家也可以通过发布一篇文章，或是一些教材等等物品，提供相应的资料获取地址，都是可以在交易所上进行拍卖的，应用实际上具有广泛运用范围，运用场景多样化。

3. Ether-Idle 的另一个亮点在于与现实生活中某些中心化网店的功能相近，但 Ether-Idle 却能够达到去中心化的交易效果，避免用户信息的集中化。本项目的核心选题背景实际上应为对当前市场的交易应用的去中心化改进，基于区块链技术实现的点对点交易，同时也保证了用户信息隐蔽，开放了交易物品的范围，使生活中的物品都可以成为交易所内部的拍卖项，建立服务于多用户的区块链物品交易所。

## 三、项目合约架构概述

Ether-Idle 的智能合约主体为三层架构，分别为：Users 用户层，Goods 商品层，EI 管理层。区别于之前的合约部署报告，对合约内容进行了较多的更改，此处主要对改进之后的项目合约进行分层介绍，从底层到管理应用层，详细解释合约内部的函数功能。

### 1. 用户层 Users

- 用户层合约主要用于处理用户注册登录等逻辑，同时提供存储用户注册数据的内部 Mapping 对象，对相应用户提供信息查询方法，也提供用户在交易所上的账户余额 EI Balance 进行查询、更改功能，于底层构建基本的交易所内部用户逻辑。

- Users.sol 合约内部主要方法：

1. 用户地址存储合法性判断，主要依据当前数组对象的内容进行传入地址的存在性验证

```
// User address existion check
function checkUserAddressExist(address addr) public constant returns (bool isExist){
    if(userAddresses.length == 0) return false;
    uint targetUserIndex = userList[addr].index;
    // Check target user's index pointing address equal to input or not
    return (userAddresses[targetUserIndex] == addr);
}
```

2. 创建新的 EI User，依据传入的注册地址进行重复判断，若为新地址则执行相应注册操作(于前端进行密码加密)

```
// Create a new user
function createNewUser(address addr, string name, string password) public returns (bool result, string errMsg){
    // Check address and username duplication
    if(checkUserAddressExist(addr)){
        return (false, "The user address has already registered an account!");
    }else if(checkUserNameDuplicate(name)){
        return (false, "The username has already been used, please pick another one!");
    }
    // Remember to store new user into userList, also create entity in userAdrrs
    userAddresses.push(addr);
    userNames.push(name);

    userList[addr] = User(addr, name, password, now, userAddresses.length - 1, 100);
    userAdrrs[name] = UserAddress(addr, userNames.length - 1);
    return (true, "");
}
```

```
}
```

3. 登录当前的 `EI User`，同时对前端传入的加密密码进行哈希比较，确保登录正确性判断

```
// Check user's address, username and password
function loginUser(address addr, string name, string password) public returns (bool
result, string errMsg){
    if(!checkUserAddressExist(addr)){
        return (false, "User address not exist!");
    }
    require(checkUserAddressExist(addr));
    User storage targetUser = userList[addr];
    if(keccak256(targetUser.username) != keccak256(name)){
        return (false, "Valid address with wrong username!");
    }else if(keccak256(targetUser.password) != keccak256(password)){
        return (false, "Valid address and username, but wrong password!");
    }
    return (true, "");
}
```

4. 获取当前对应地址的用户信息，返回合约内部的存储数据

```
// Get target user's info
function getUserInfo(address addr) public constant returns (bool result, string
username, uint userTime, uint userIndex){
    if(!checkUserAddressExist(addr)){
        return (false, "", 0, 0);
    }
    return (
        true,
        userList[addr].username,
        userList[addr].timestamp,
        userList[addr].index
    );
}
```

5. 获取对应地址用户的 `EI Balance` 具体数值，同时也提供对其账户余额的修改操作

```
// Check user balance
function getBalanceByAddr(address addr) public constant returns (uint balance){
    return userList[addr].balance;
}

// Increase user balance

function increaseBalance(address targetAddr, uint count) public payable returns (bool
```

```

result){
    if(userList[targetAddr].balance + count <= count){
        return false;
    }
    userList[targetAddr].balance += count;
    return true;
}

// Decrease user balance
function decreaseBalance(address targetAddr, uint count) public payable returns (bool result){
    if(userList[targetAddr].balance < count){
        return false;
    }
    userList[targetAddr].balance -= count;
    return true;
}

```

## 2. 商品层 Goods

- 经过合约内容的改进，商品层内部修正了之前存在的：商品不正确删除的问题，同时对合约函数进行了部分修改，但整体内容仍为对交易所货架物品的增加 Pending、卖出 Accepting 与查询 Getting 功能。考虑到智能合约无法对其内部数组对象上的某一元素进行删除操作，合约内部维护一个合法下标数组，维持商品正确添加、卖出行为。
- Goods.sol 合约内部主要方法：
  1. 给交易货架上添加新的拍卖商品，同时限制同一用户添加多个一样名字的商品，更新当前合法商品下标数组

```

// Seller update goods
function createNewGoods(address addr, string name, uint256 price, string realInfos, string descri) public payable returns (bool result, string mess){
    // check seller has ever create same name goods infos or not
    for(uint i=0; i<goodIndexes.length; ++i){
        Good storage temp = goodLists[goodIndexes[i]];
        if(temp.sellerAddr == addr && keccak256(temp.name) == keccak256(name)){
            return (false, "Create target good failed! You have already created another goods with the same name!");
        }
    }

    // New goods info
    uint len = goodIndexes.length;
    length += 1;
    goodIndexes.push(len+1);
    goodLists[len+1] = Good(addr, name, realInfos, descri, price, now, len+1);
    return (true, "");
}

```

2. 删去货架上的对应商品，完成卖出操作，同时对合法商品数组进行更新，防止出现商品未删除的情况

```
// Buyer buy goods
function acceptGoods(uint index) public payable returns (bool result, address addr,
uint price){
    for(uint i=0; i<goodIndexes.length; i++){
        if(goodIndexes[i] == index){
            address add = goodLists[index].sellerAddr;
            uint pri = goodLists[index].price;
            delete goodIndexes[i];
            delete goodLists[index];
            return (true, add, pri);
        }
    }
    return (false, address(0), 0);
}
```

3. 获取对应商品的信息

```
// Get all goods infos
function getTargetGoods(uint order) public returns (address addr, string name, string
real, string dest, uint256 pri, uint trueID){
    Good temp = goodLists[goodIndexes[order]];
    return (temp.sellerAddr, temp.name, temp.realTranscatInfos, temp.description,
temp.price, goodIndexes[order]);
}
```

### 3. 管理层 EI

- 此处的管理层合约进行了完全的重构(这是在此报告中重新阐述合约内容的核心原因)，剔除了之前对 控制 概念的误解，从对底层合约 Users 与 Goods 的所有方法控制，转型到对交易所内部的行为规范，主要体现在：
  1. 为交易双方的账户余额安全性提供保障，负责处理买家对相应交易的 Confirm，而不是马上进行账户余额更改，而是在双方确认之后，调用 Users 的转账功能对买卖双方，进行相应数目 EI Coin 的转账。
  2. 提供交易记录的存储，为用户提供当前未确认，已确认交易的查询接口。
  3. 提供从未确认交易转换到已确认交易的接口，暴露给相应具有执行权利的买家。
- EI.sol 合约内部主要方法：
  1. 对交易所内部的交易行为结构体定义 EI\_Good

```

struct EI_Good {
    address sellerAddr;
    address buyerAddr;
    string goodsName;
    string sellerName;
    string buyerName;
    uint256 price;
    uint timestamp;
}

```

2. 获取对应用户参与 **作为买家或卖家** 的所有交易记录，提供函数分为未确认记录与已确认记录

```

// Get user's target pending transaction, promise providing legal index
function getPendingTrans(address addr, uint index) public constant returns (address
addr1, address addr2, string gname, string name1, string name2, uint256 price, uint
time) {
    uint currIndex = 0;
    for(uint i=0; i<pendingLength; i++){
        if(pendingList[i].sellerAddr == addr || pendingList[i].buyerAddr == addr){
            if(currIndex == index){
                return (pendingList[i].sellerAddr, pendingList[i].buyerAddr,
pendingList[i].goodsName, pendingList[i].sellerName,
pendingList[i].buyerName, pendingList[i].price,
pendingList[i].timestamp);
            }else {
                currIndex += 1;
            }
        }
    }
    return (address(0), address(0), "", "", "", 0, 0);
}

// Get user's target confirmed transaction, promise providing legal index
function getConfirmedTrans(address addr, uint index) public constant returns (address
addr1, address addr2, string gname, string name1, string name2, uint256 price, uint
time) {
    uint currIndex = 0;
    for(uint i=0; i<confirmedLength; i++){
        if(confirmedList[i].sellerAddr == addr || confirmedList[i].buyerAddr ==
addr){
            if(currIndex == index){
                return (confirmedList[i].sellerAddr, confirmedList[i].buyerAddr,
confirmedList[i].goodsName, confirmedList[i].sellerName,
confirmedList[i].buyerName, confirmedList[i].price,
confirmedList[i].timestamp);
            }else {
                currIndex += 1;
            }
        }
    }
}

```

```
        return (address(0), address(0), "", "", "", 0, 0);
    }
```

3. 将未确认的交易记录转换为已确认的交易记录，同时进行双方账户余额的相应更改，增加到对应的已确认交易历史记录中(代码较长，可查看项目仓库中的 `contracts/EI.sol`)

```
// Confirm pending EI goods (Only buyer can confirm the transaction)
function confirmPendingTrans(address sellerAddr, address buyerAddr, string gname)
    payable returns (bool result) {}
```

## 四、项目使用说明

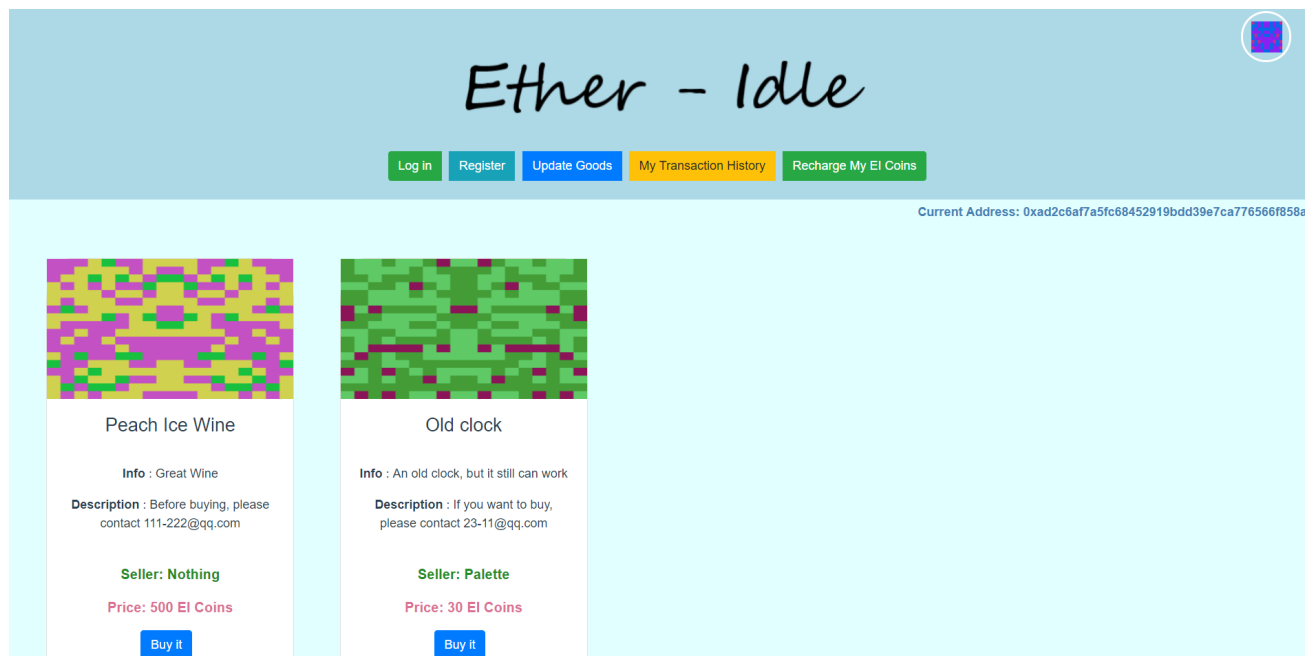
- 进入项目的 Github 代码仓库 `Ether-Idle`，将项目 `clone` 到本地文件夹。由于本项目采取 `vue-cli` 进行前端开发，故进入项目根路径执行 `npm install` 安装依赖，执行完成之后则使用 `npm run dev` 运行本地项目，使用 `http://localhost:8080` 访问项目所在地址，即可开始使用项目。
- 项目环境依赖于 `MetaMask` 插件，最好使用 `Chrome` 浏览器进行测试。
- 项目仓库包含 `video` 文件夹，内部包括本人对项目的测试视频，具体使用过程也可以参考该视频。项目挂载在网站(<https://angry-albattani-d69151.netlify.com/>)，若需要进行测试，可以通过访问该网站进行使用，网站使用 `Ropsten` 测试链进行连接，第一次使用时需要获取测试币，须翻墙到(<https://faucet.metamask.io/>)获取对应数量的测试币。

## 五、项目 Web 端开发架构与合约交互

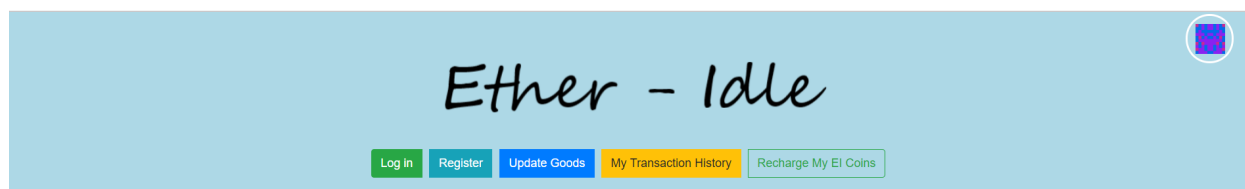
1. 项目采用 `Vue` 进行前端页面设计，以及与相关合约层的数据交互。采取的开发框架为 `Vue-cli`，通过分层架构进行与 `MetaMask` 的数据获取，以及前端页面渲染，和面向不同层合约发起 `payable` 函数请求。
  - 静态层 `assets`：存储静态图片文件。
  - 工具层 `utils`：负责与 `MetaMask` 进行初始化数据的获取，初始化前端的各项信息，同时监听区块链上用户信息的更改，进行 `polling` 操作保证数据新鲜。
  - 基本数据层 `store`：负责存储部署后的合约地址以及合约 `ABI`，同时负责全局信息存储对象 `$store` 的初始化。
  - 路由层 `router`：本应用为基于 `Vue` 的 `SPA` 实现，在单页面中通过调用不同的组件进行逻辑操作，路由层实际并未起用处。
  - 组件层 `components`：通过编写相应的页面组件，对用户请求与输入数据进行接收处理，调用相应组件执行相关行为，同时执行内部的合约请求函数，获取对应数据返回给用户。
2. 本项目合约部署仍采用 `Remix` 进行合约编译，通过 `Injected Web3`，基于 `MetaMask` 与私有链进行合约的部署。获取相关合约地址与 `ABI`，供给前端进行基于 `web3` 的合约交互。

## 六、项目使用说明

项目全局视图



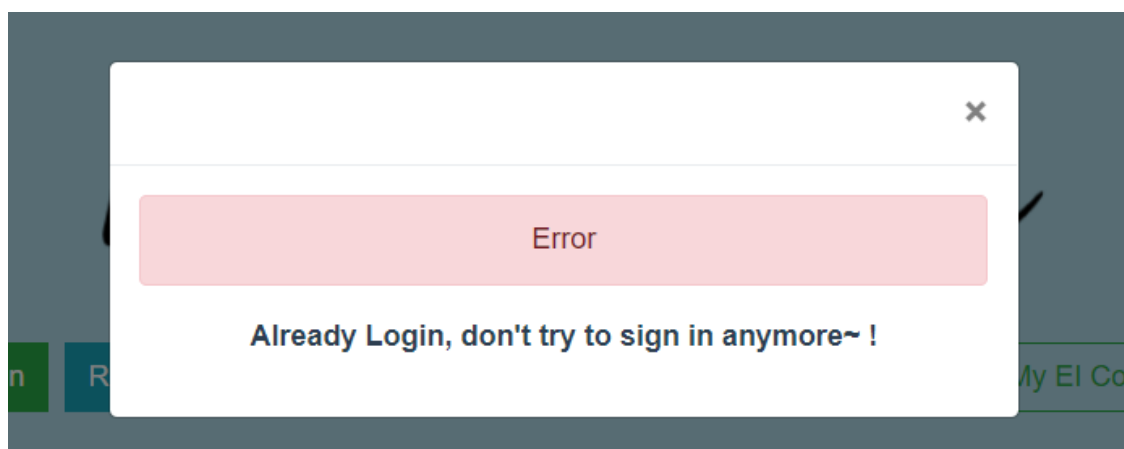
1. Ether-Idle 上方导航栏，提供不同按钮进行功能调用：



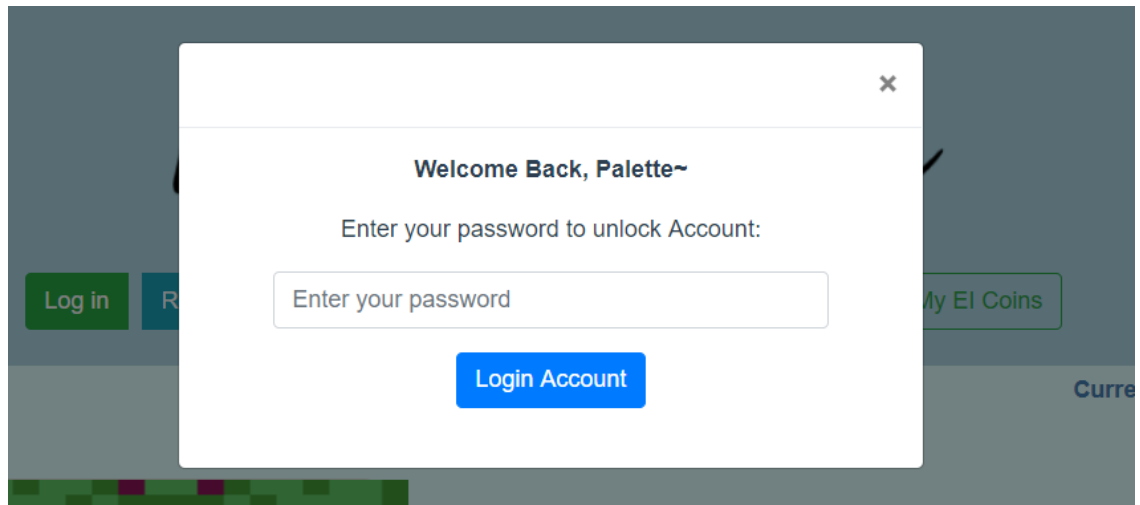
上方的按钮分别对应于五个不同功能：

1. 用户登录功能 Log In

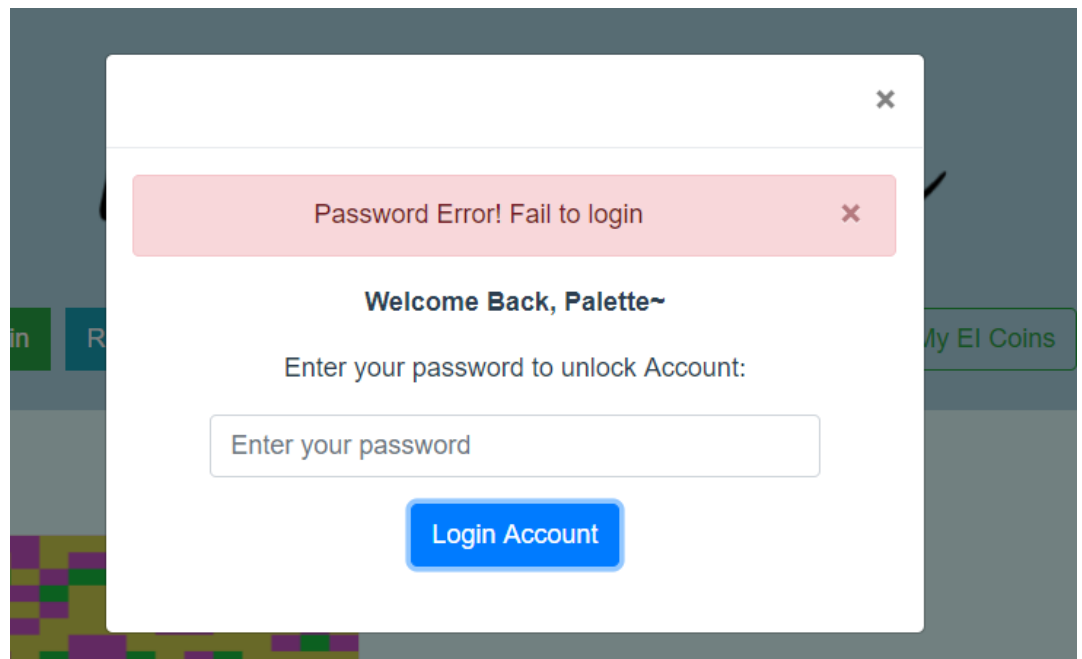
- 登录功能不仅可通过点击按钮实现，本应用仍实现了用户进行交易所即检测其地址注册与否，分别自动弹出对应的登录框或注册框。
- 前端实现对当前用户地址的自动检测，判断其是否以及注册过 Ether-Idle 的账户。若未注册，则点击本按钮会弹出提示错误框，若已注册但未登录则弹出勿重复登录信息，否则弹出登录密码接受框。

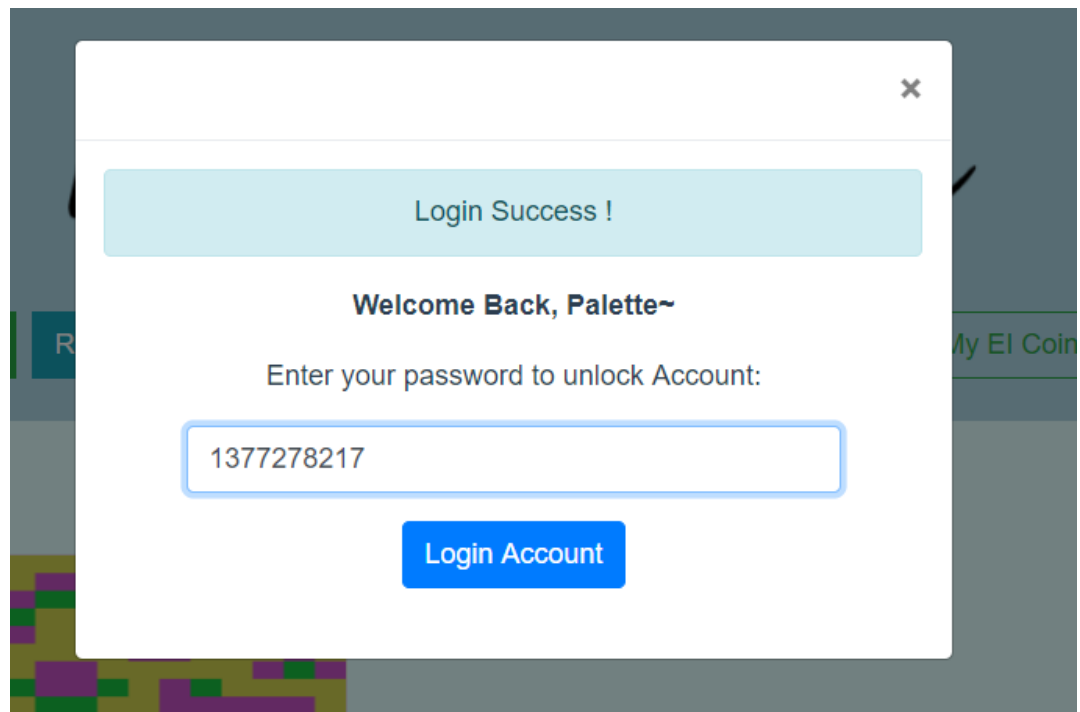






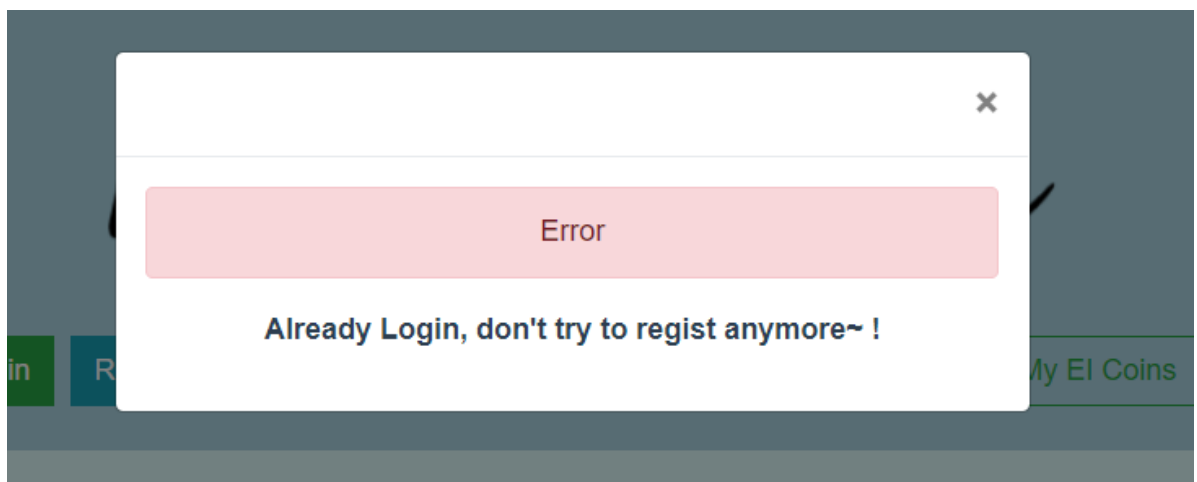
- 对于登录窗口而言，在用户输错密码的同时，上方会出现对应的错误提示；反之输入正确，则显示正确登录信息。

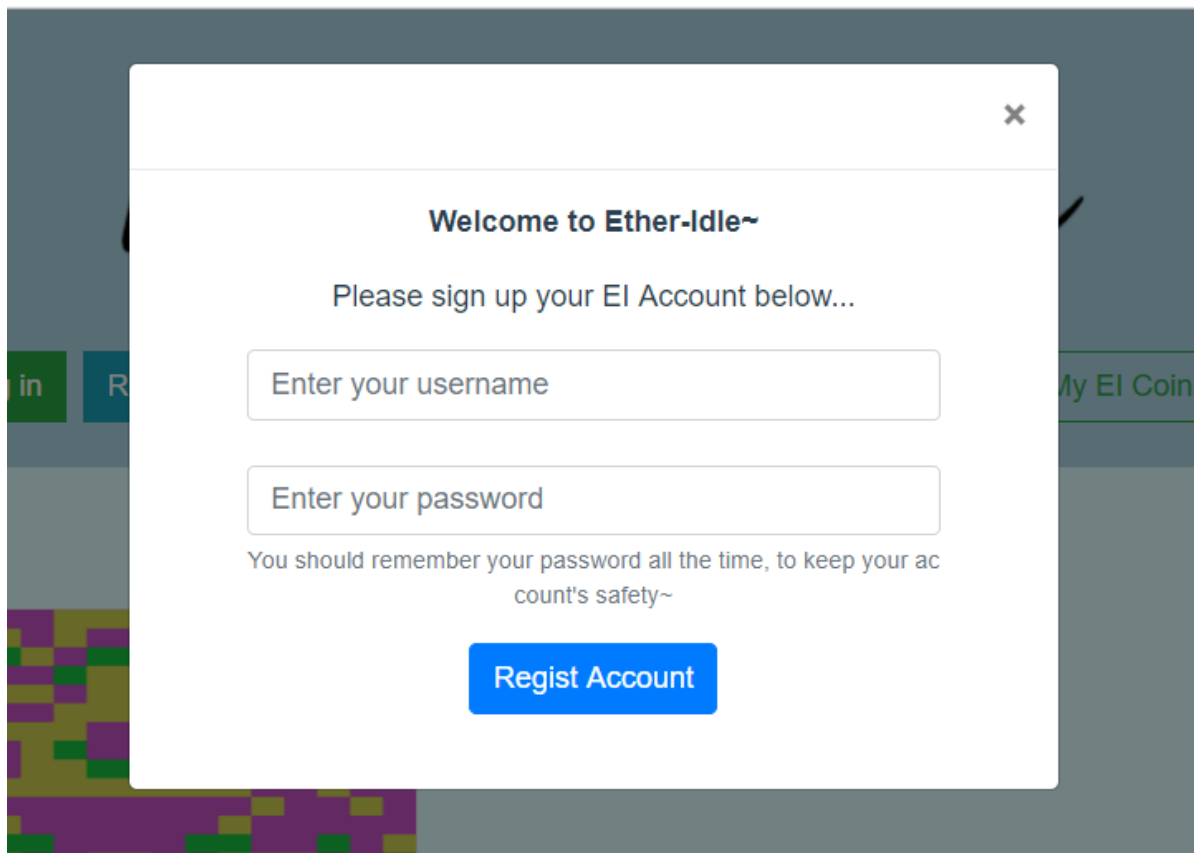




## 2. 用户注册功能 Register

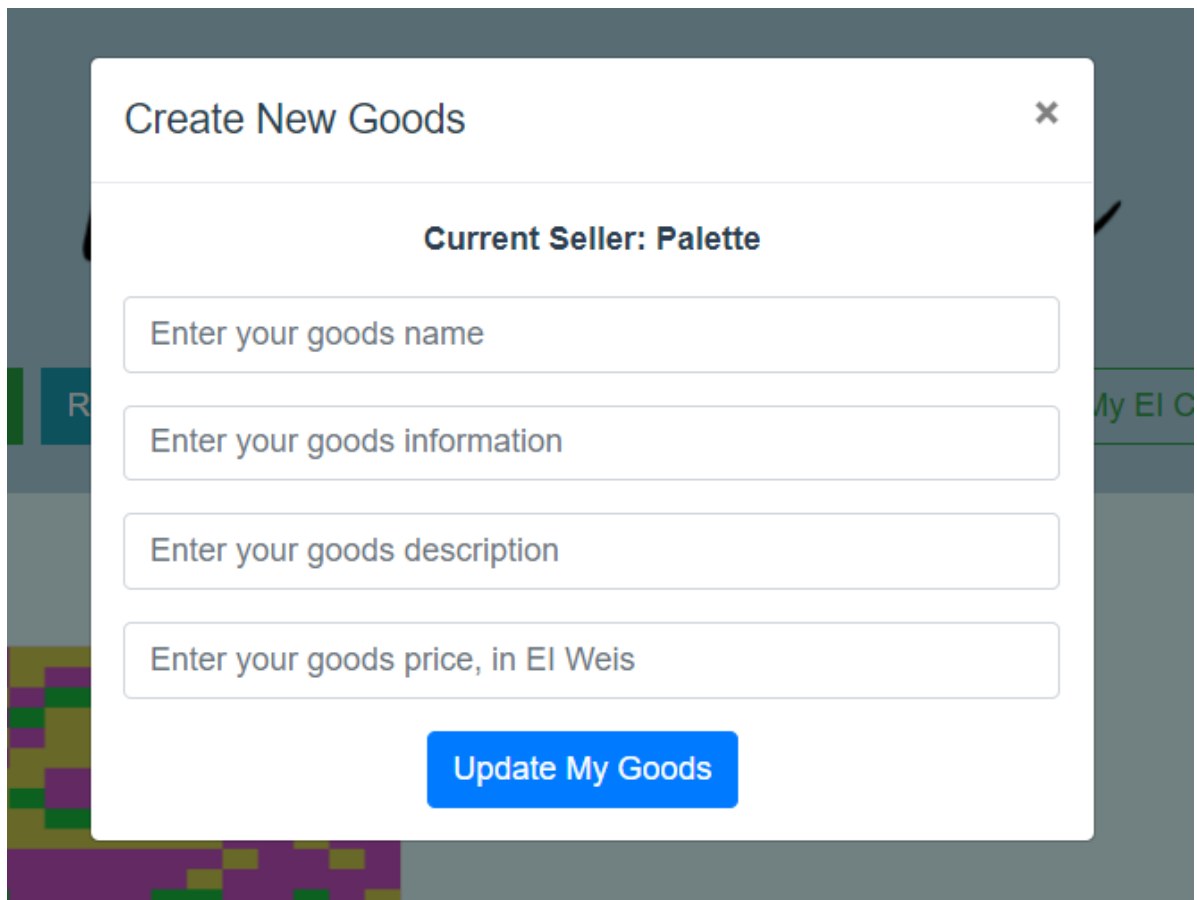
- 注册功能不仅可通过点击按钮实现，本应用仍实现了用户进行交易所即检测其地址注册与否，分别自动弹出对应的登录框或注册框。
- 与登录功能类似，当用户按下注册按钮时，系统判断当前操作合法与否，才决定弹出错误提示框，还是正确的注册框。





### 3. 用户上传新商品功能 `UpdateGoods`

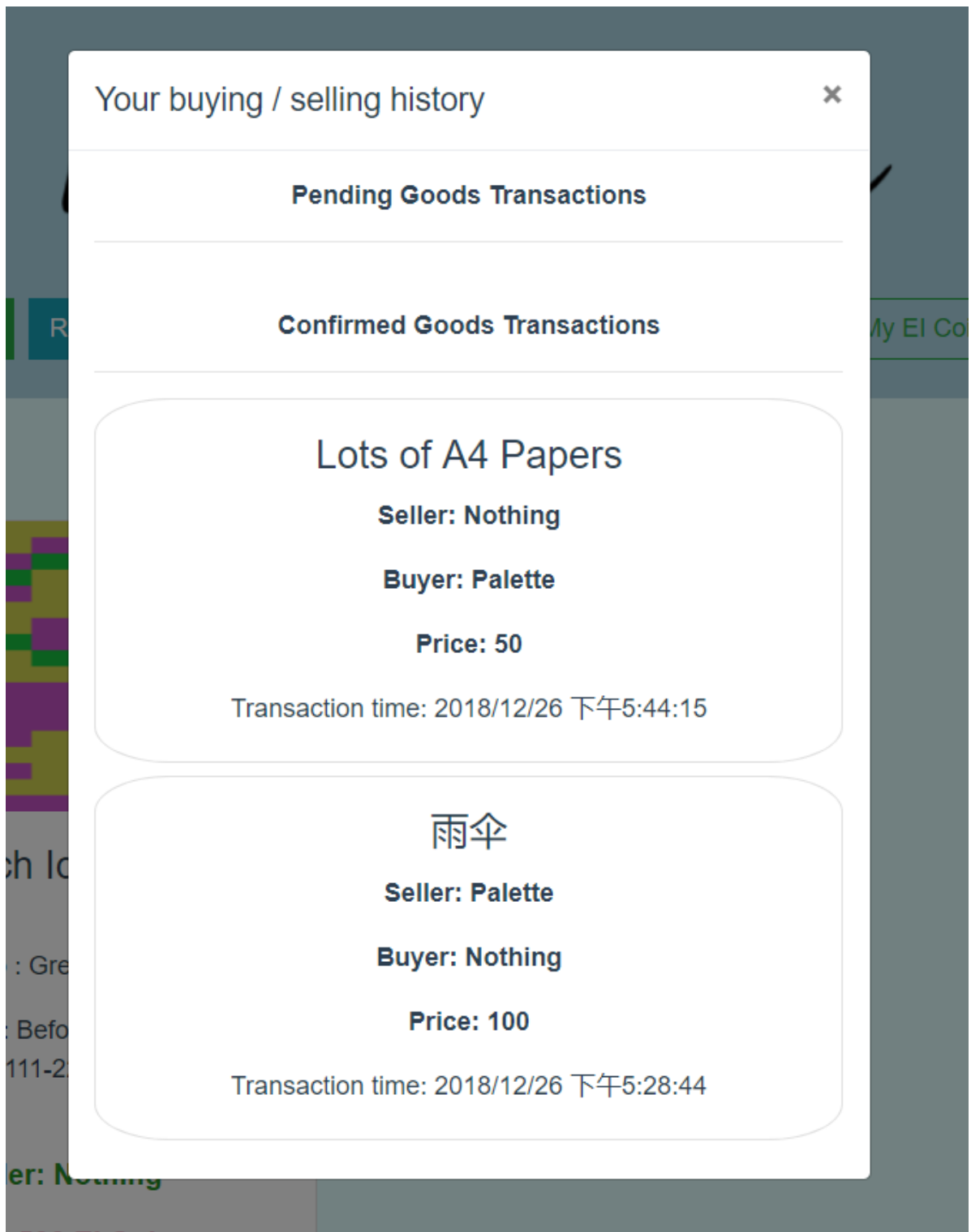
- 上传商品功能提供弹出窗口，用户通过输入对应商品的相关信息，上传其具有的闲置商品，供给其他用户进行购买。



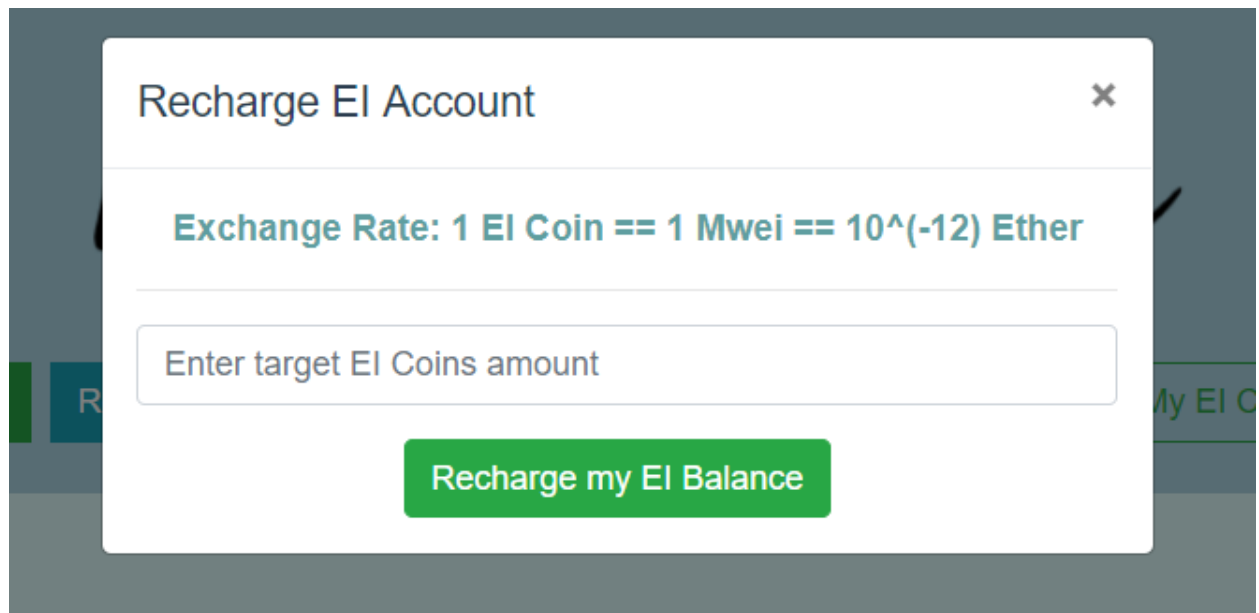
The image shows a modal window titled "Create New Goods" with a close button (X) in the top right corner. Below the title, it says "Current Seller: Palette". There are four text input fields stacked vertically, each with a placeholder text: "Enter your goods name", "Enter your goods information", "Enter your goods description", and "Enter your goods price, in El Weis". At the bottom right of the modal is a blue button labeled "Update My Goods".

#### 4. 用户查看历史交易记录功能 `My Transaction History`

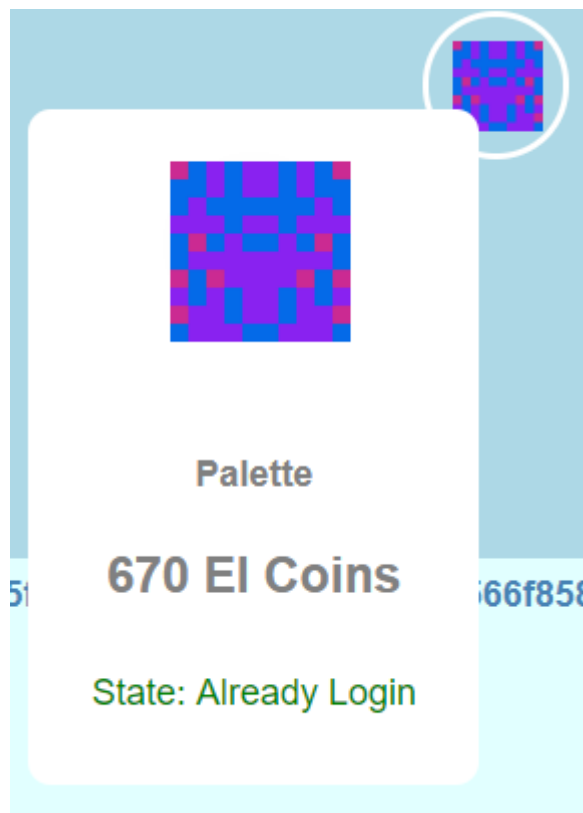
- 此功能负责给用户展示其历史参与的所有交易，包括当前未确认与已确认的所有交易，同时，为那些在未确认交易中做为买家的用户，提供 `confirm` 权利。(此处没有包含未确认的交易)



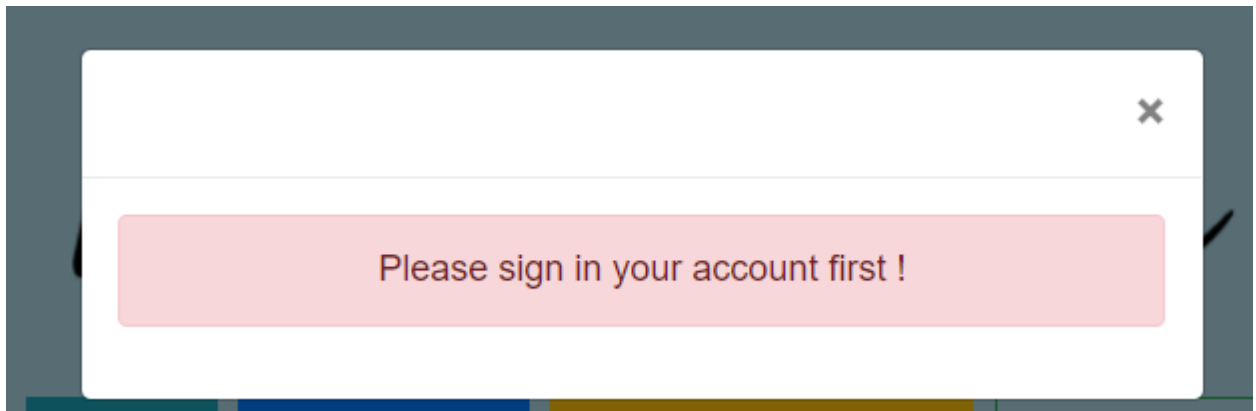
5. 用户使用以太币进行 [EI Coin](#) 兑换功能 [Recharge My EI Coins](#)



6. 在应用的上部分导航栏框内，除了五大功能函数调用之外，在右上方有用户的简略信息浮窗按钮，点击即可查看当前用户的基本状态



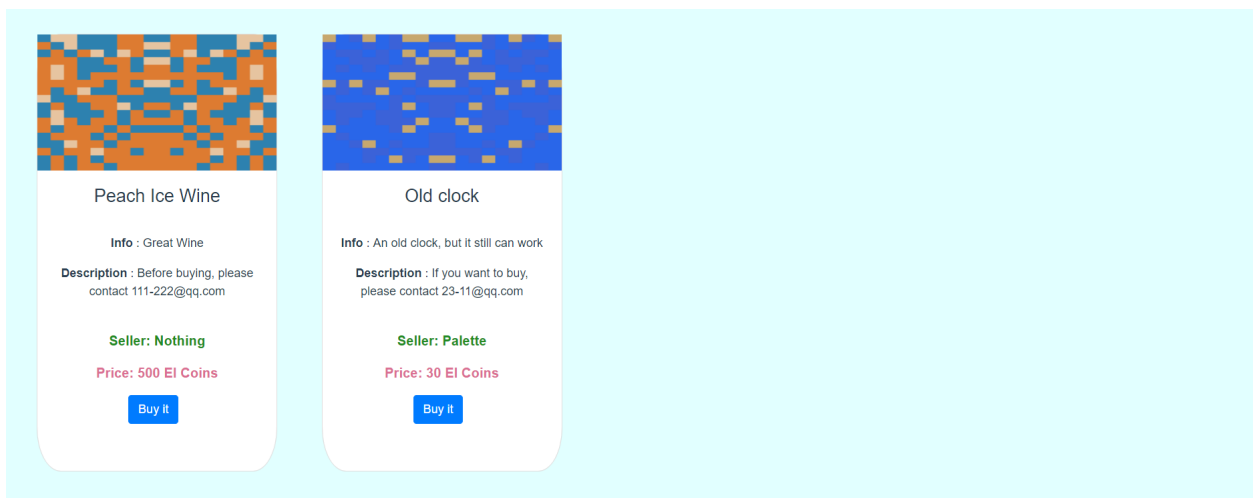
7. 对于所有的基本功能操作，交易所严格规定用户必须以登录状态执行操作，否则弹出错误提示框



8. 用户地址存储部分，主要用于显示当前用户对应的地址，无其余功能

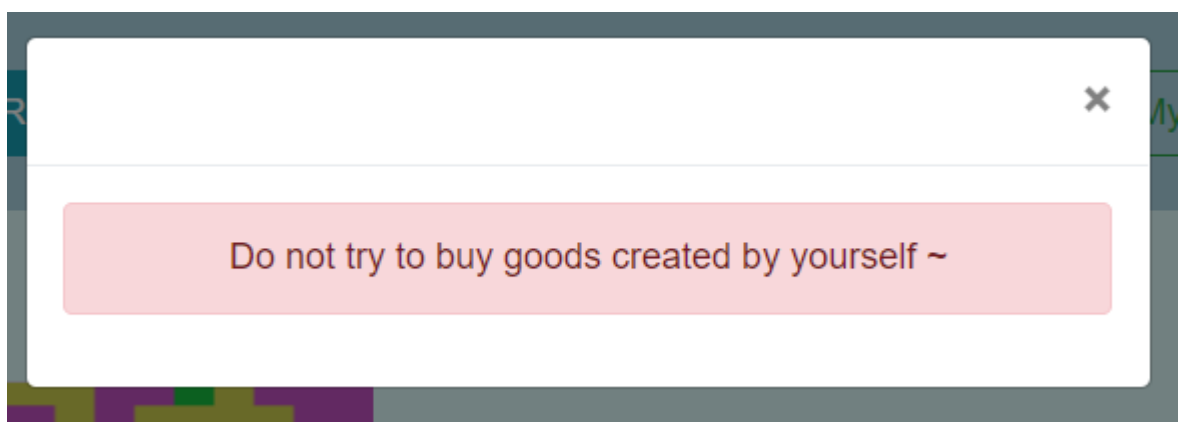


9. Ether-Idle 商品货架展示

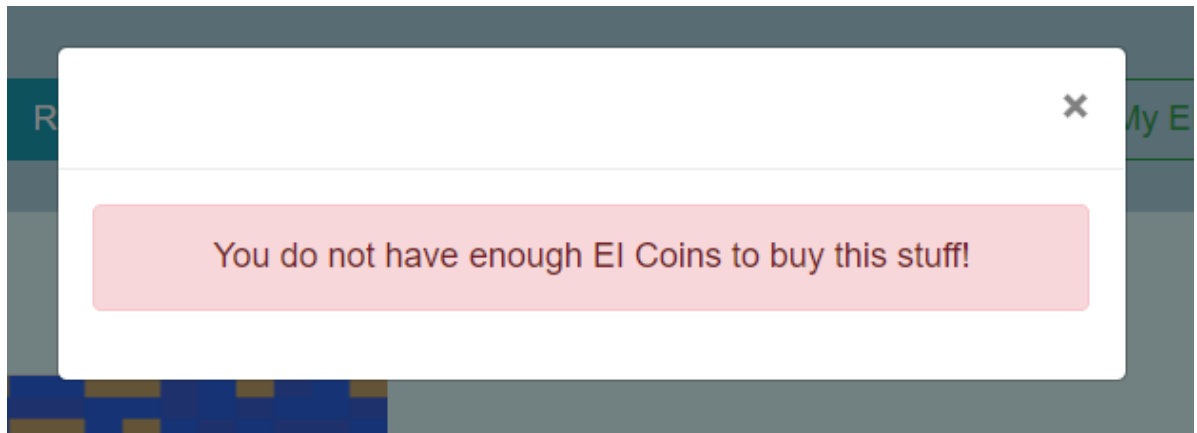


货架上的商品以 Bootstrap 展示，用户可以通过之前的 UpdateGoods 功能进行商品的添加。

1. 限制用户不能购买自己发布的商品

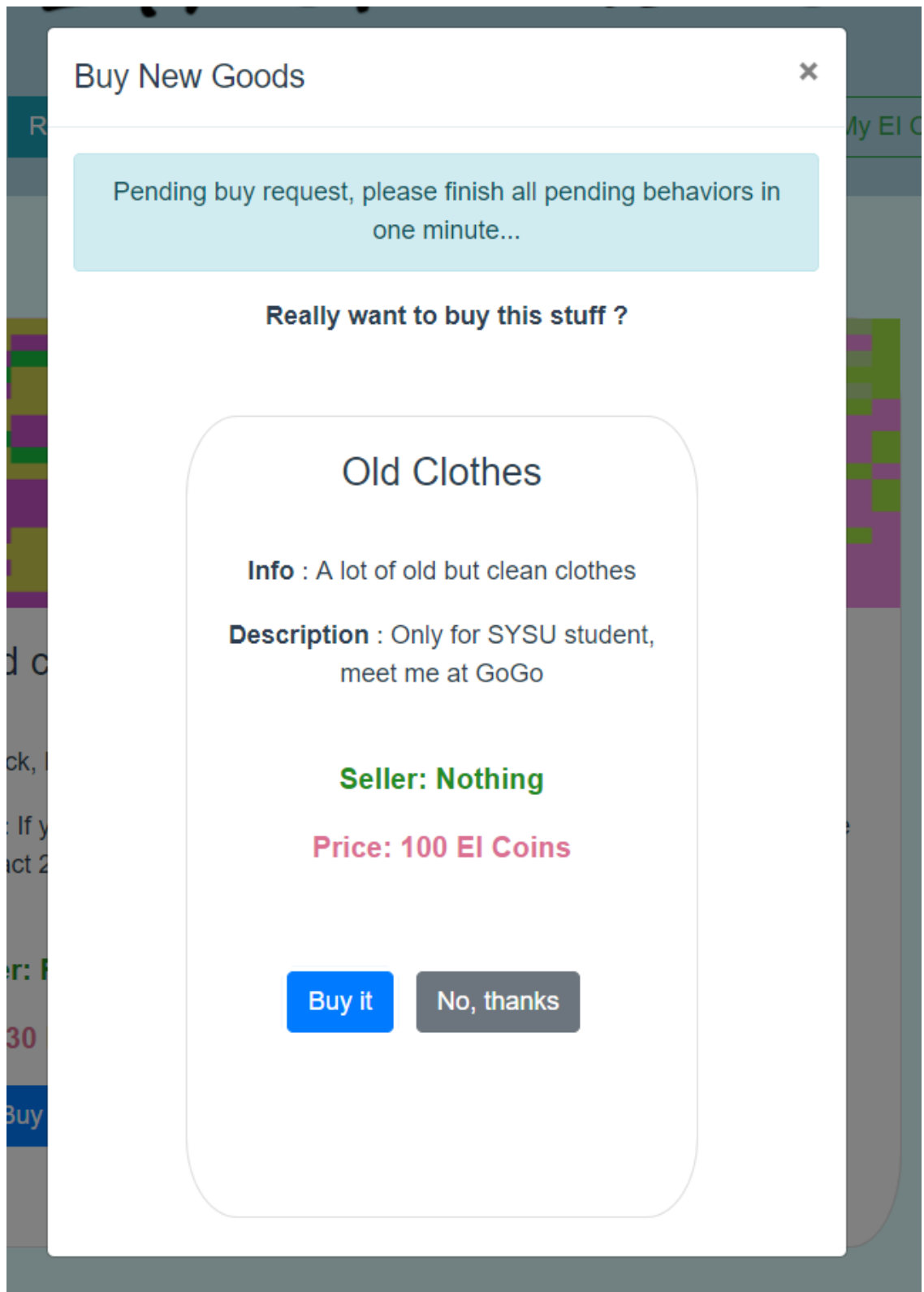


2. 限制用户不能购买比自己余额还贵的商品



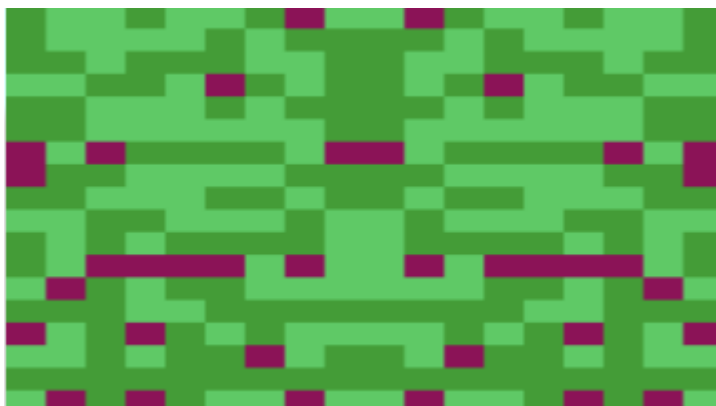
3. 用户合法操作，点击某个物品购买按钮，出现详情页(截图时间不同)





10. 相信通过之前的截图展示，你肯定注意到了用户、商品的图片部分。

- o 此处的图片考虑到不可能让区块链来存储此些数据，同时为了界面美化与独特性展示，采用了 `blockies` 组件进行基于对象哈希值的像素图像生成，分别加入到目标对象中，以此美化展示。



- 但此组件对于图像的生成也依赖于生成的时间戳，故在不同时间查看对应商品的图像，总会出现不同。

## 七、项目测试

项目测试过程大致如下：

- 首先在 `MetaMask` 上创建两个用户 `Palette` , `Nothing` , 重新在 `Remix` 上部署三层合约(合约依赖的 `Solidity` 编译器版本为 `0.4.22` ), 使用 `Ropsten` 测试链简化测试过程, 启动对项目的整体性测试。

creation of Users pending...

<https://ropsten.etherscan.io/tx/0xd19d4d6dddcac8ee91b6e56826d5b2fe94966892ee2a7e36a16a69627e3b04a9>

creation of Goods pending...

<https://ropsten.etherscan.io/tx/0x875718f2ebc9300dcf96243ffc6f29594854d86c2f18e8f9bea3af3c5ae2bba>

creation of EI pending...

<https://ropsten.etherscan.io/tx/0x96f69c7863f44e27a70cf0d2fb3e1b06303cc057978b7c7066892beeadd5e5c9>

✓ [block:4705852 txIndex:15] from:0xad2...f858a to: value:0 wei

Debug ▼

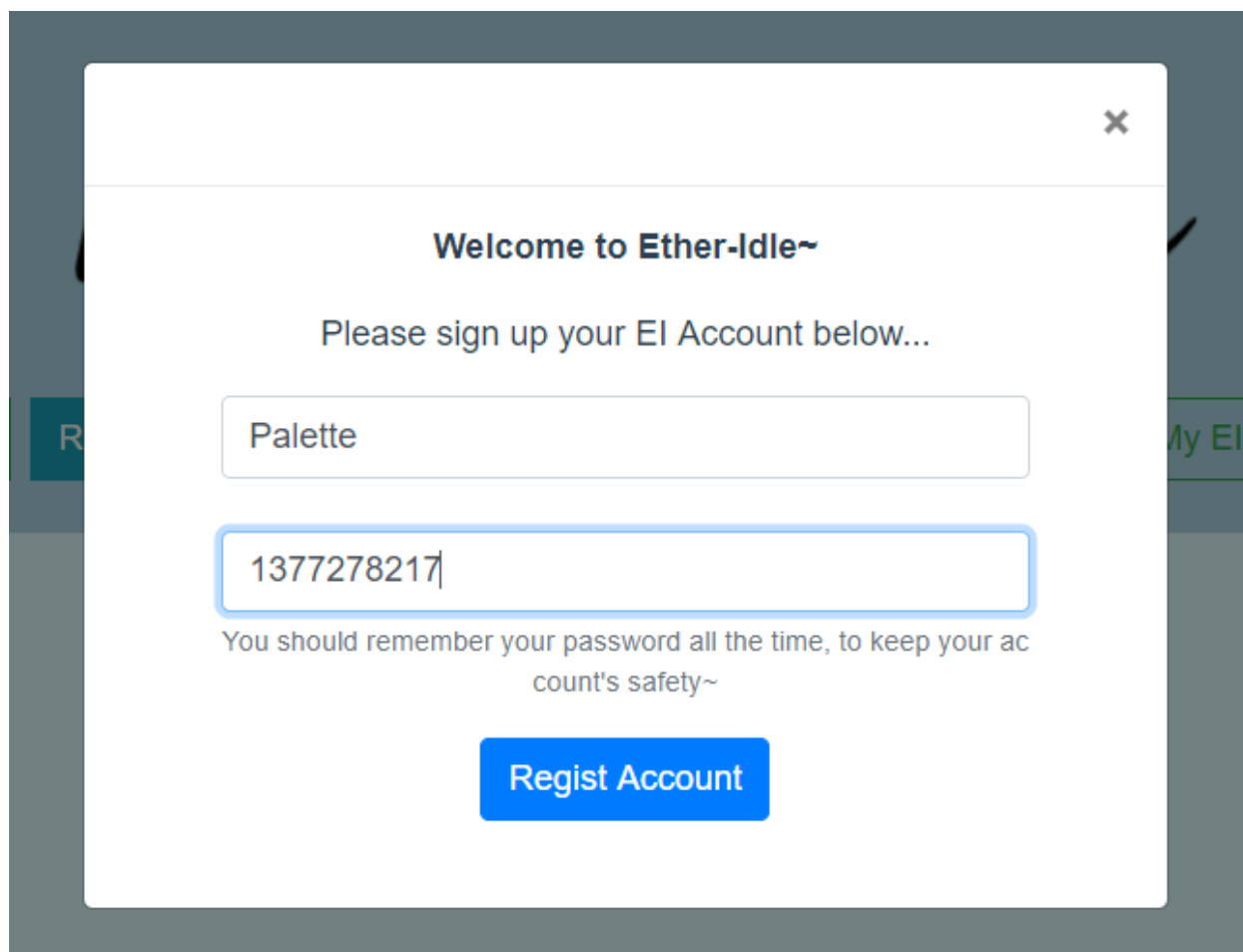
✓ [block:4705852 txIndex:16] from:0xad2...f858a to:EI.(constructor) value:0 wei data:0x608...90029 logs:0  
hash:0x96f...5e5c9

Debug ▼

✓ [block:4705852 txIndex:14] from:0xad2...f858a to: value:0 wei

Debug ▼

- 第一步：验证用户注册登录功能是否正确 (测试用户：Palette)



A registration dialog box titled "Welcome to Ether-Idle~" with a close button (X) in the top right corner. The text "Please sign up your EI Account below..." is centered. Below it are two input fields: the first contains "Palette" and the second contains "1377278217". Below the input fields is a warning message: "You should remember your password all the time, to keep your account's safety~". At the bottom is a blue button labeled "Regist Account".

Welcome to Ether-Idle~

Please sign up your EI Account below...

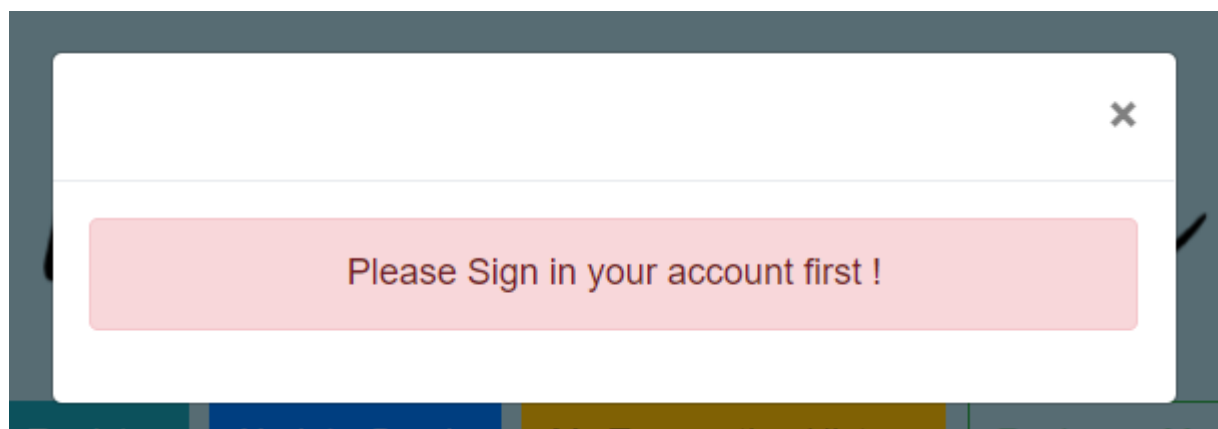
Palette

1377278217

You should remember your password all the time, to keep your account's safety~

Regist Account

成功注册之后，刷新界面，跳出请求登录框，说明注册成功。下一步则对未登录用户权限进行测试。

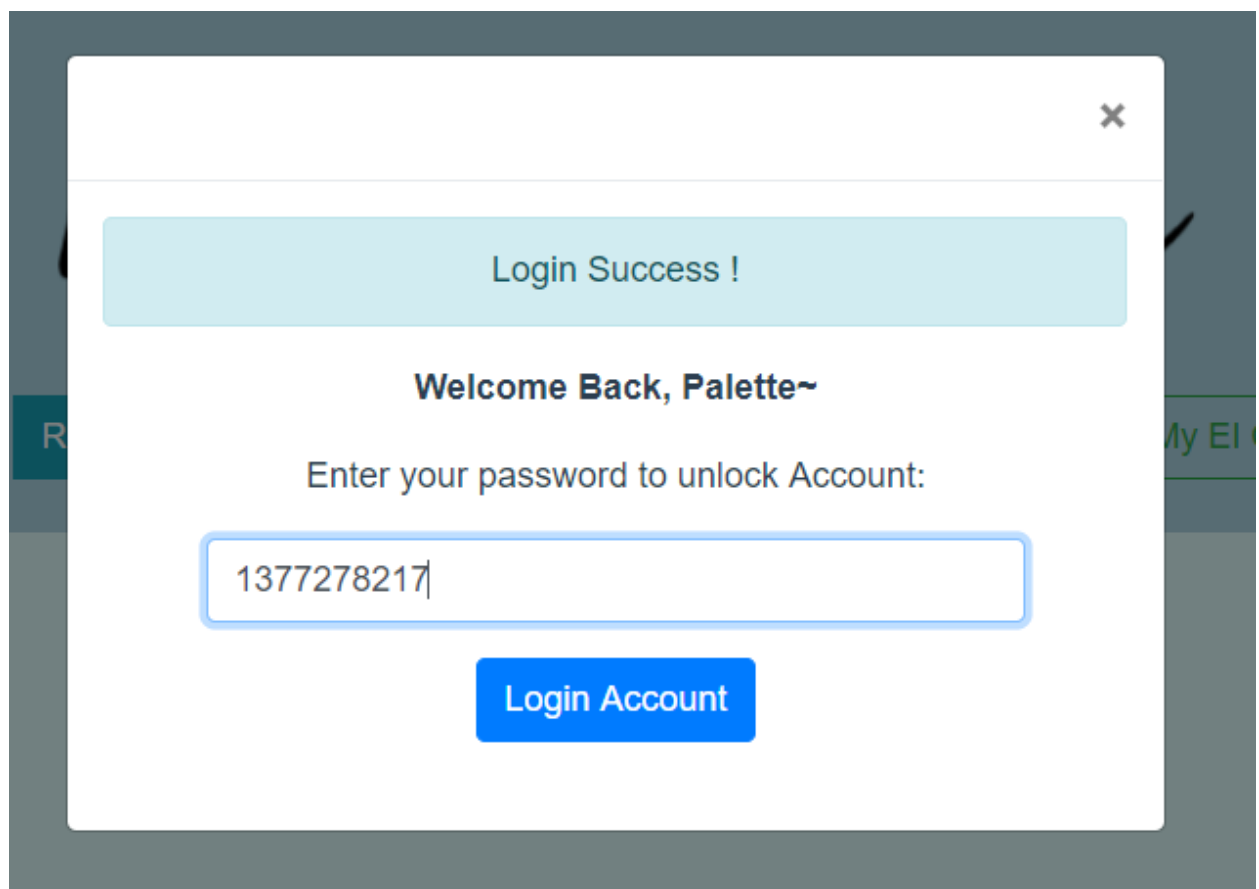


A login prompt dialog box with a close button (X) in the top right corner. It contains a single line of text: "Please Sign in your account first !".

Please Sign in your account first !



执行登录操作，可以看到刚刚部署的合约中，货物数量为0。



- 第二步：验证用户之间上传商品，拍卖商品过程的正确运行。

首先由 `Palette` 上传一件商品，商品名为 `Old Sport Shoes`

Create New Goods

Pending new good, please wait several seconds until all things done~

Current Seller: Palette

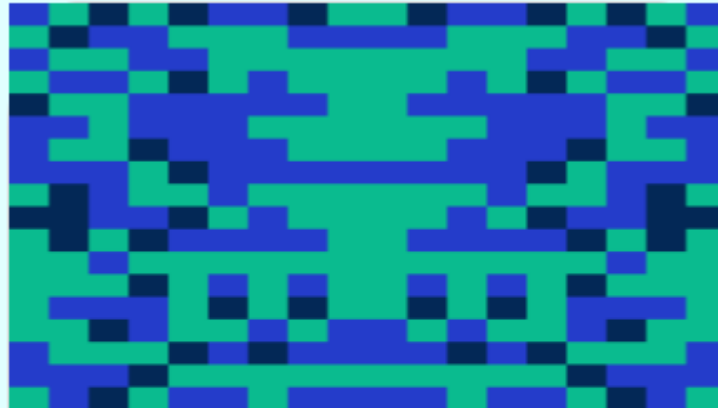
Old Sport Shoes

A pair of great sport shoes

Only for SYSU student, meet me at GoGo

50

Update My Goods



## Old Sport Shoes

**Info :** A pair of great sport shoes

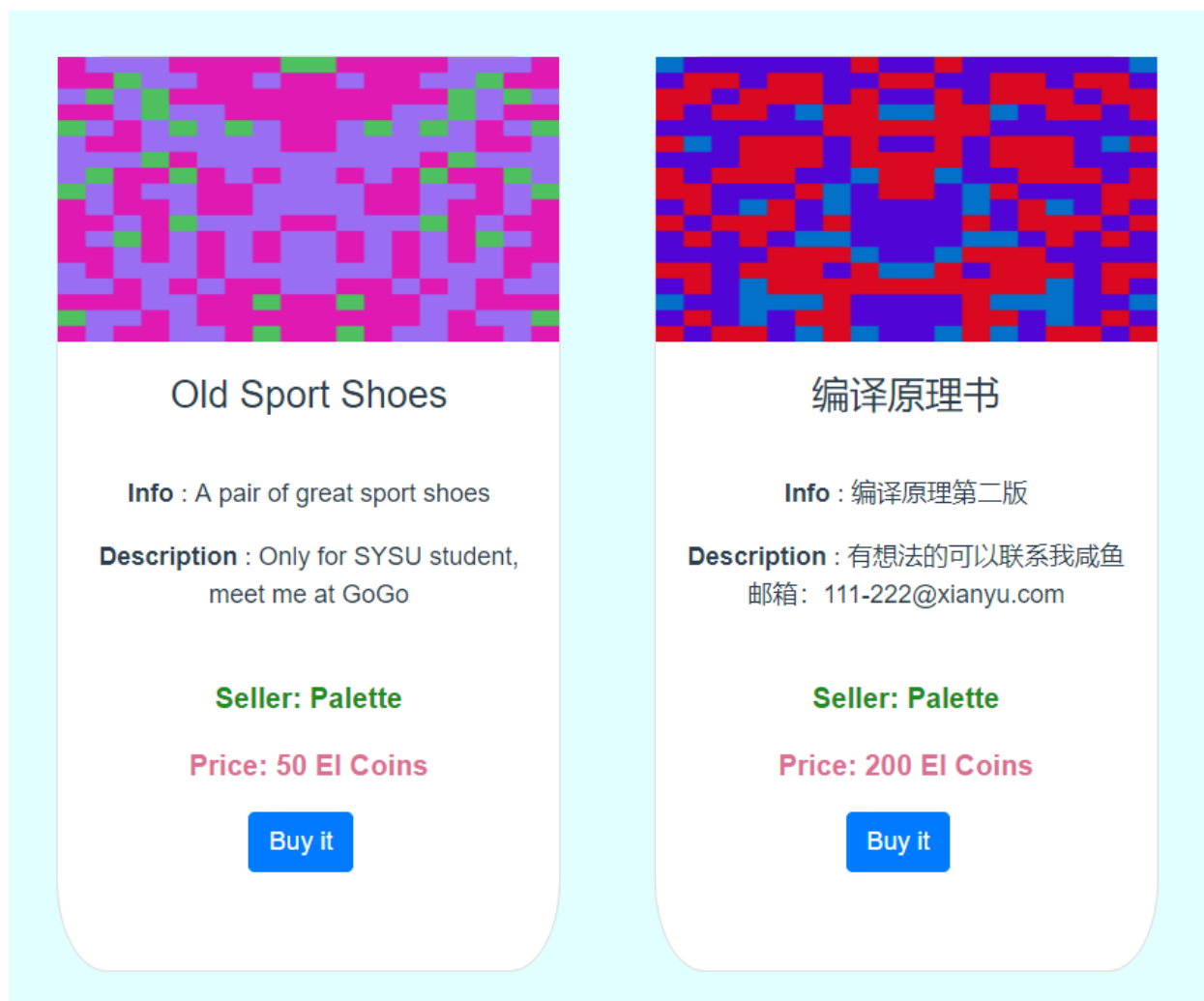
**Description :** Only for SYSU student,  
meet me at GoGo

**Seller: Palette**

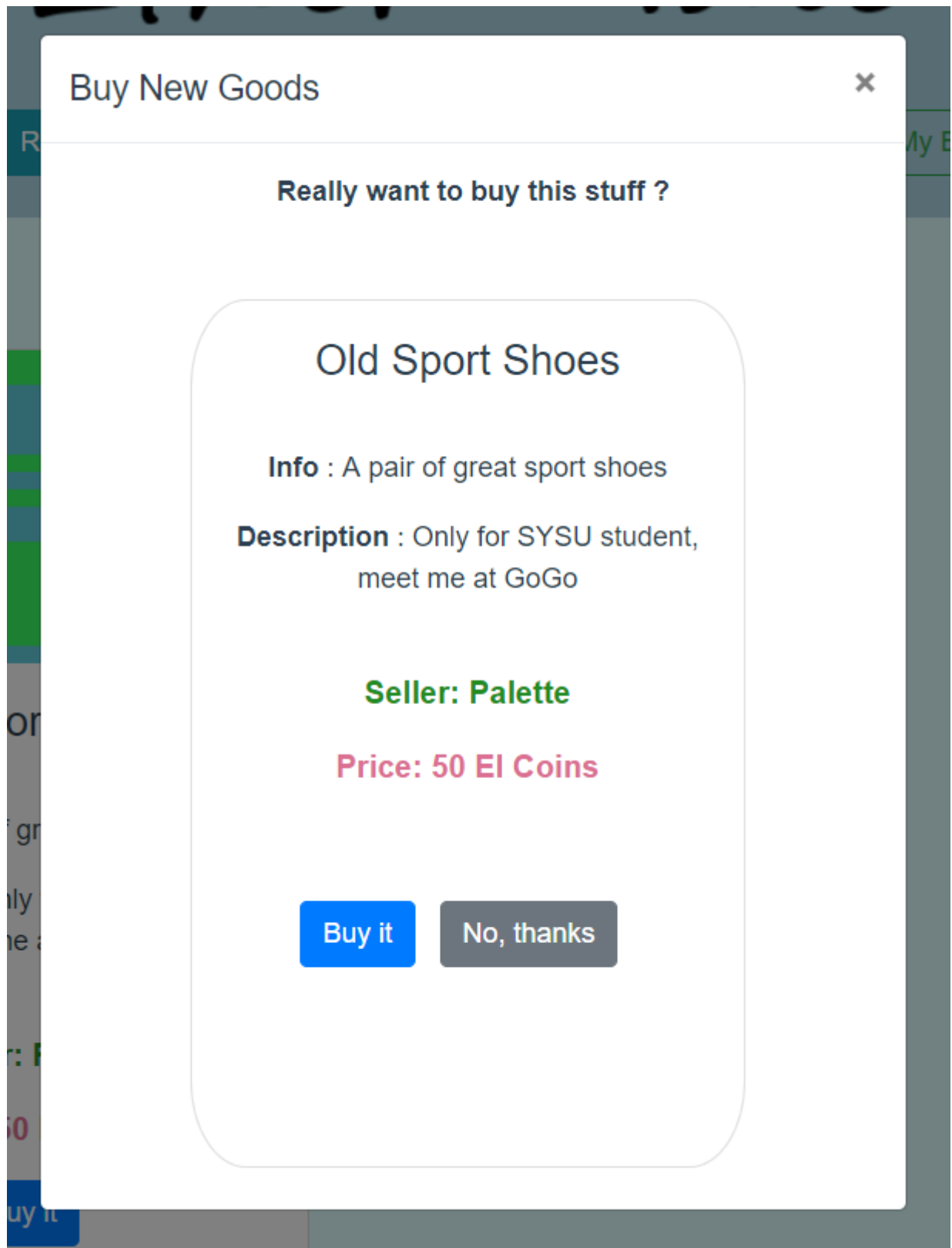
**Price: 50 EI Coins**

Buy it

可以看到上传成功，由 `Palette` 再次上传一件商品，名为 `编译原理书` (进行中文上传测试)



将用户切换到 `Nothing` , 其首先对 `Palette` 的 `Old Sport Shoes` 进行拍卖操作。



点击购买按钮，弹出 MetaMask 确认框，等待交易被确认



## Buy New Goods



Pending buy request, please finish all pending behaviors in one minute...

**Really want to buy this stuff ?**

### Old Sport Shoes

**Info :** A pair of great sport shoes

**Description :** Only for SYSU student,  
meet me at GoGo

**Seller: Palette**

**Price: 50 EI Coins**

Buy it

No, thanks

交易被确认，等待将该笔交易加入到未确认的交易历史中

## Buy New Goods



Buying target goods successfully! Now please finish pending transaction to your personal history!

**Really want to buy this stuff ?**

### Old Sport Shoes

**Info :** A pair of great sport shoes

**Description :** Only for SYSU student,  
meet me at GoGo

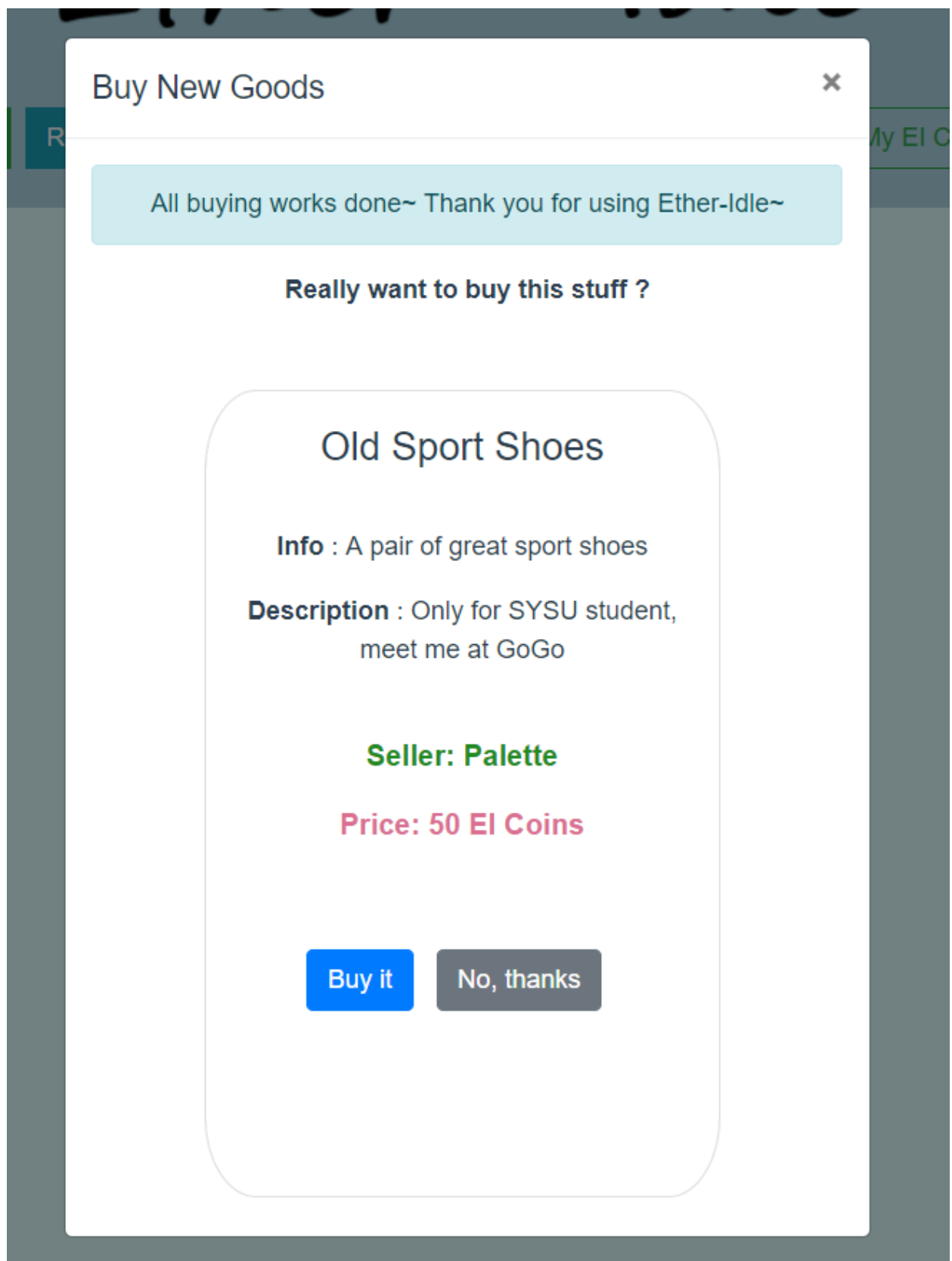
**Seller: Palette**

**Price: 50 EI Coins**

Buy it

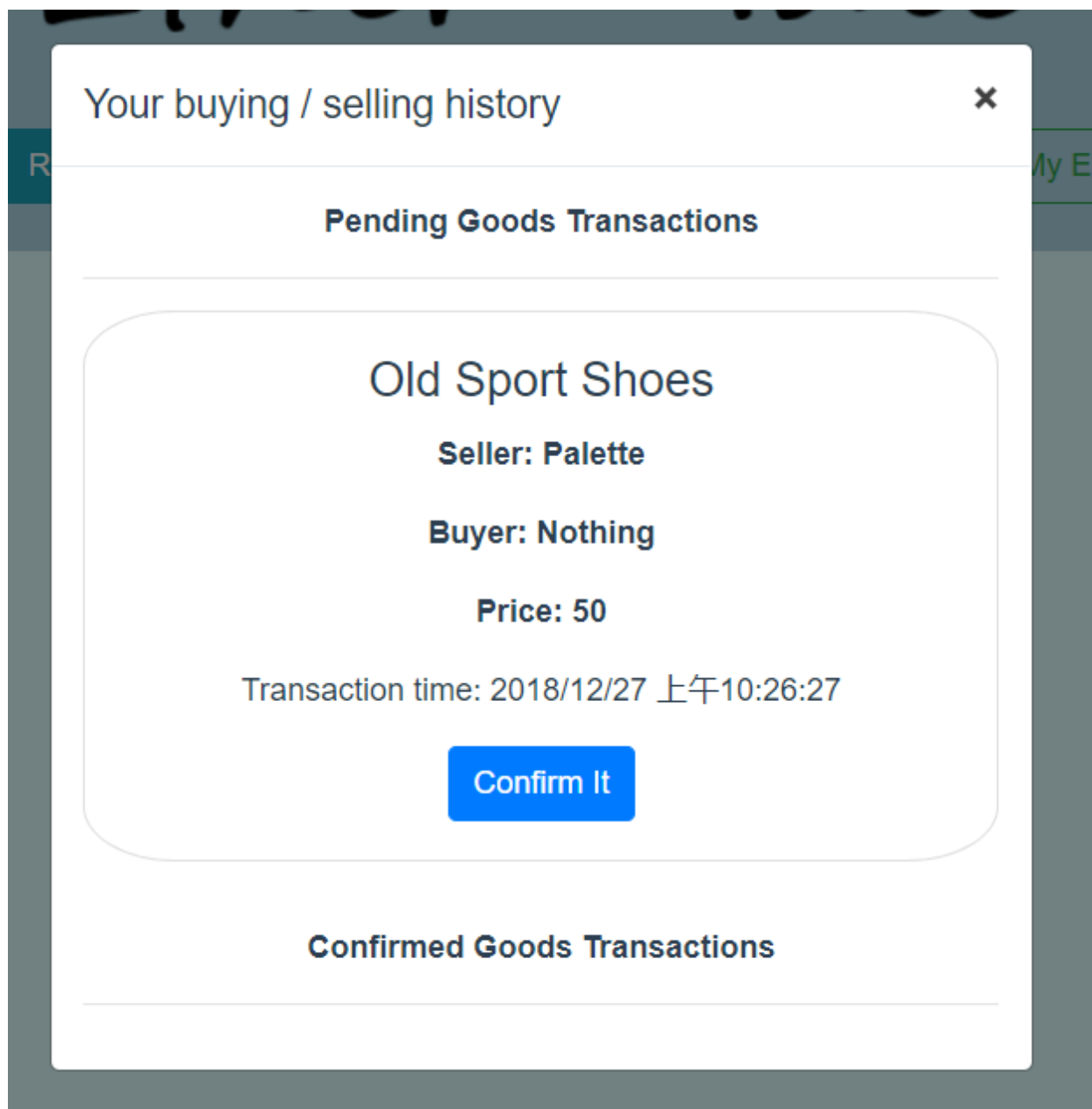
No, thanks

所有操作完成

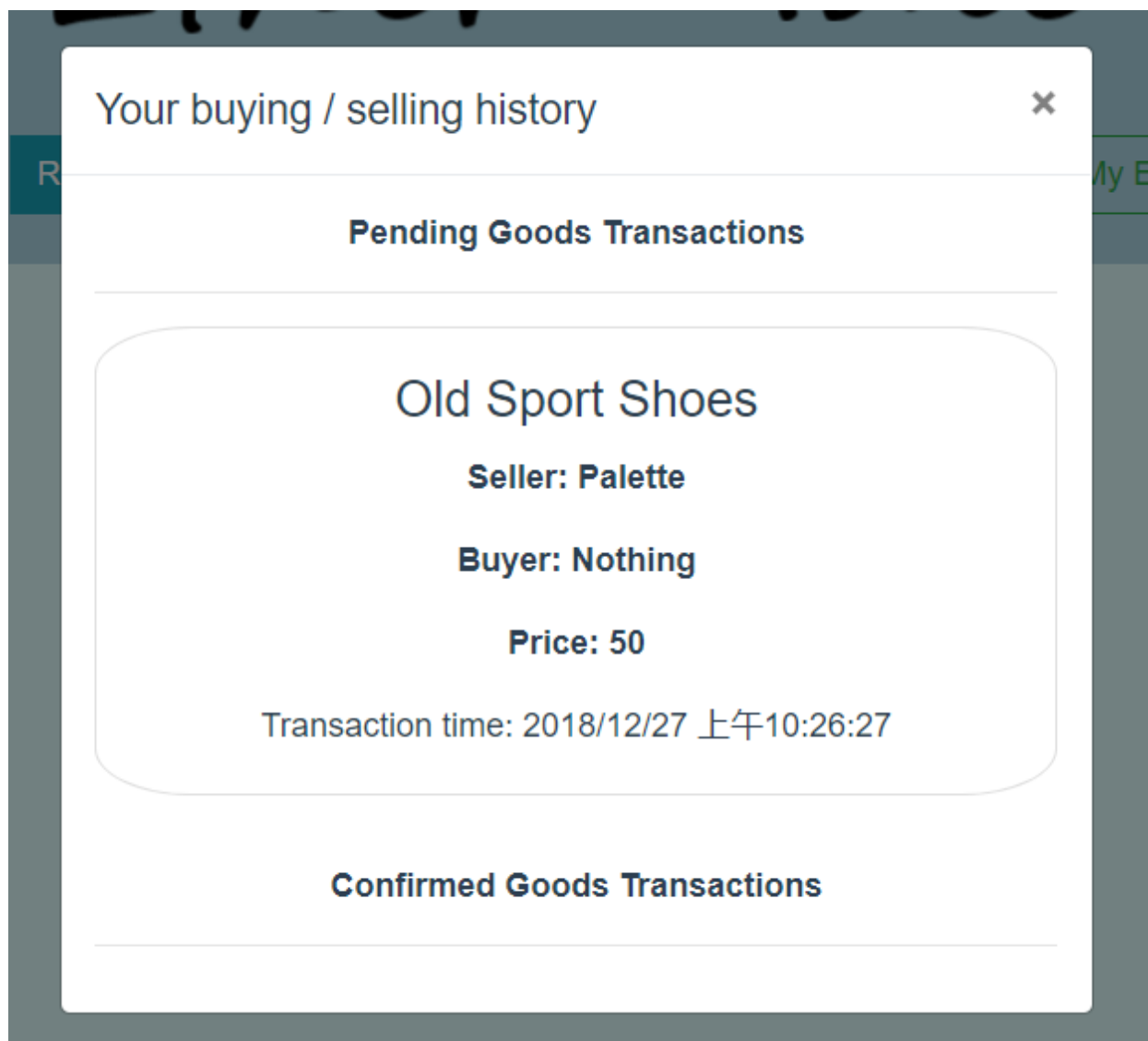


3. 第三步：依赖于之前的拍卖纪录，此时查看两个用户的未确认交易记录，同时测试买家 `Nothing` 对该笔交易的 `Confirm` 权限

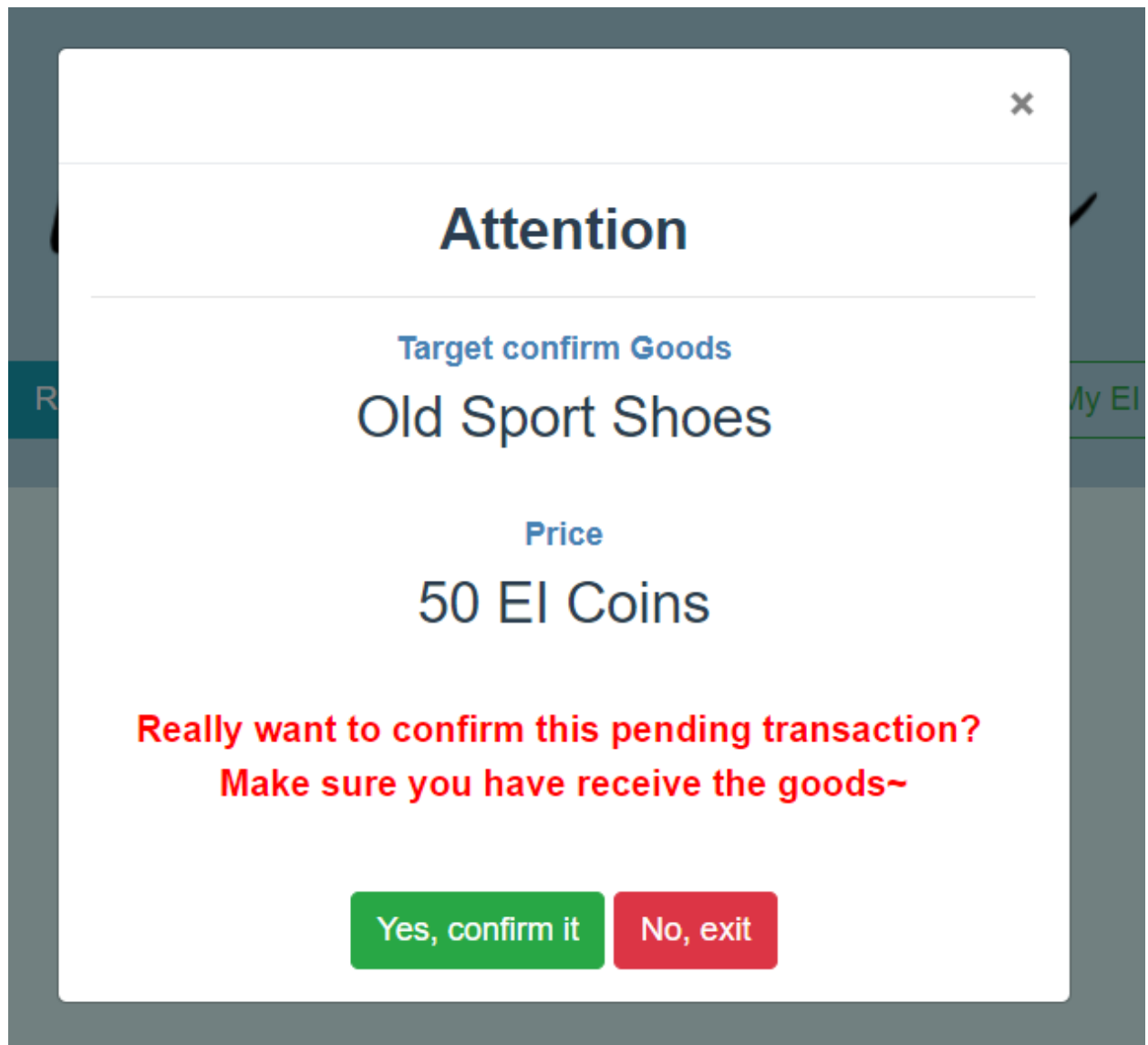
买家 `Nothing` 的交易历史



卖家 `Palette` 的交易历史



现在执行买家的 `Confirm` 功能，确认该笔交易已经完成



交易确认中.....



Confirming transaction, please finish behaviors in one minute~

## Attention

Target confirm Goods

Old Sport Shoes

Price

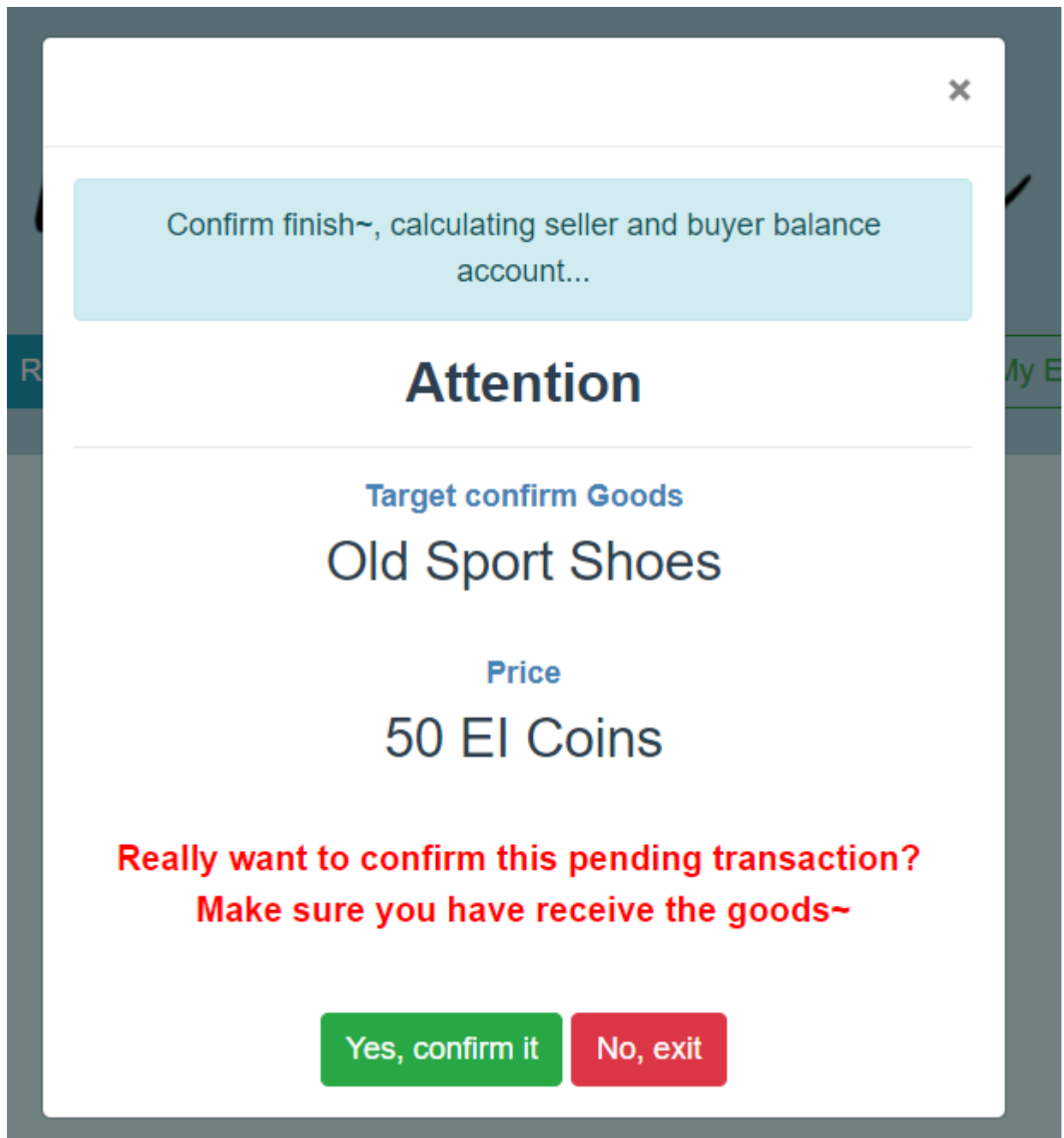
50 EI Coins

**Really want to confirm this pending transaction?  
Make sure you have receive the goods~**

Yes, confirm it

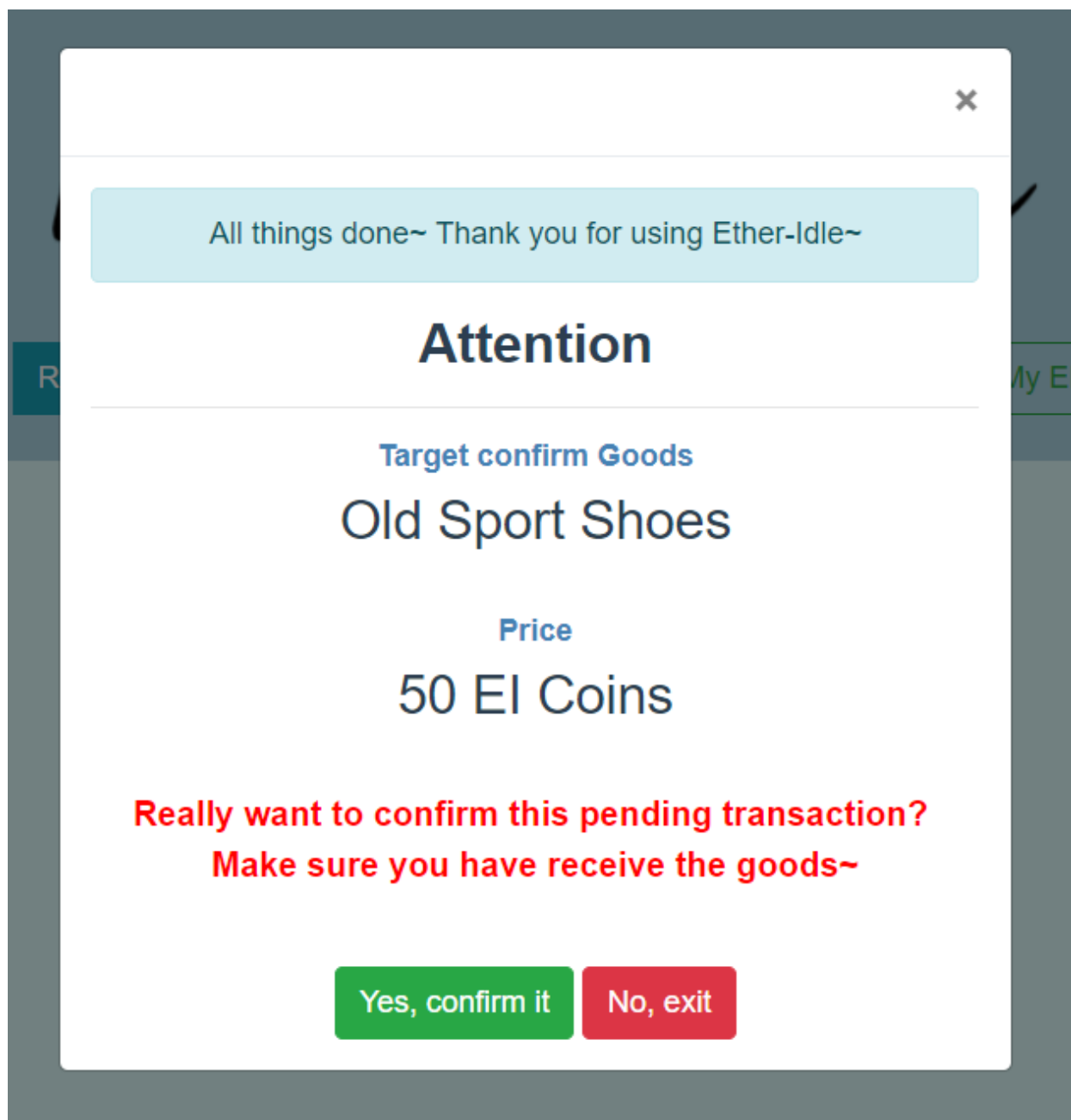
No, exit

交易确认完成，计算双方余额，分别扣去/增加相应的 EI Coins

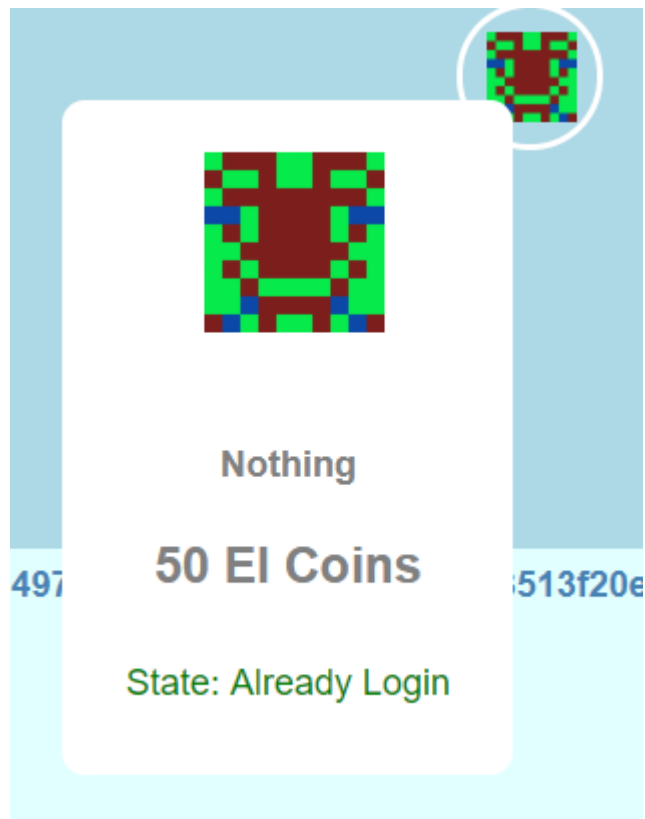


所有工作完成





此时查看对应买家 `Nothing` 的账户余额



买家的已确认历史交易记录

## Your buying / selling history



### Pending Goods Transactions

### Confirmed Goods Transactions

#### Old Sport Shoes

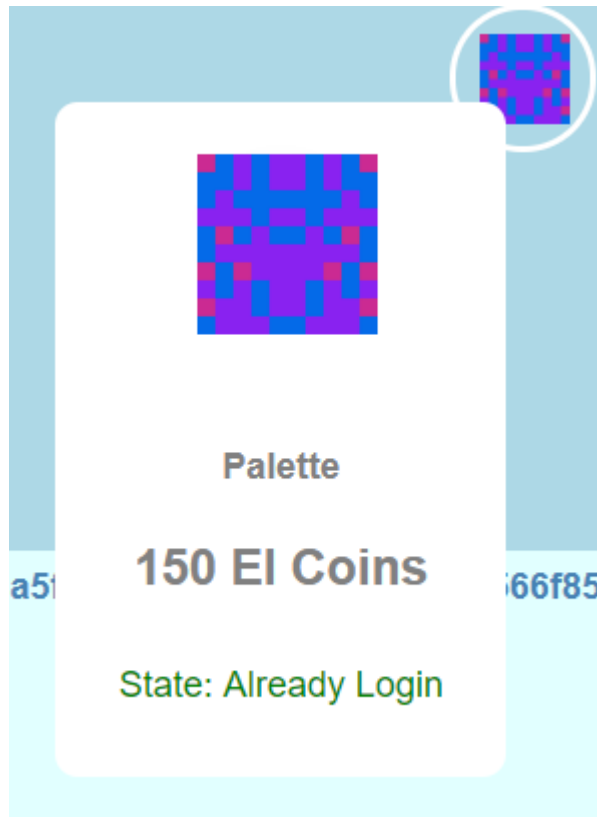
**Seller: Palette**

**Buyer: Nothing**

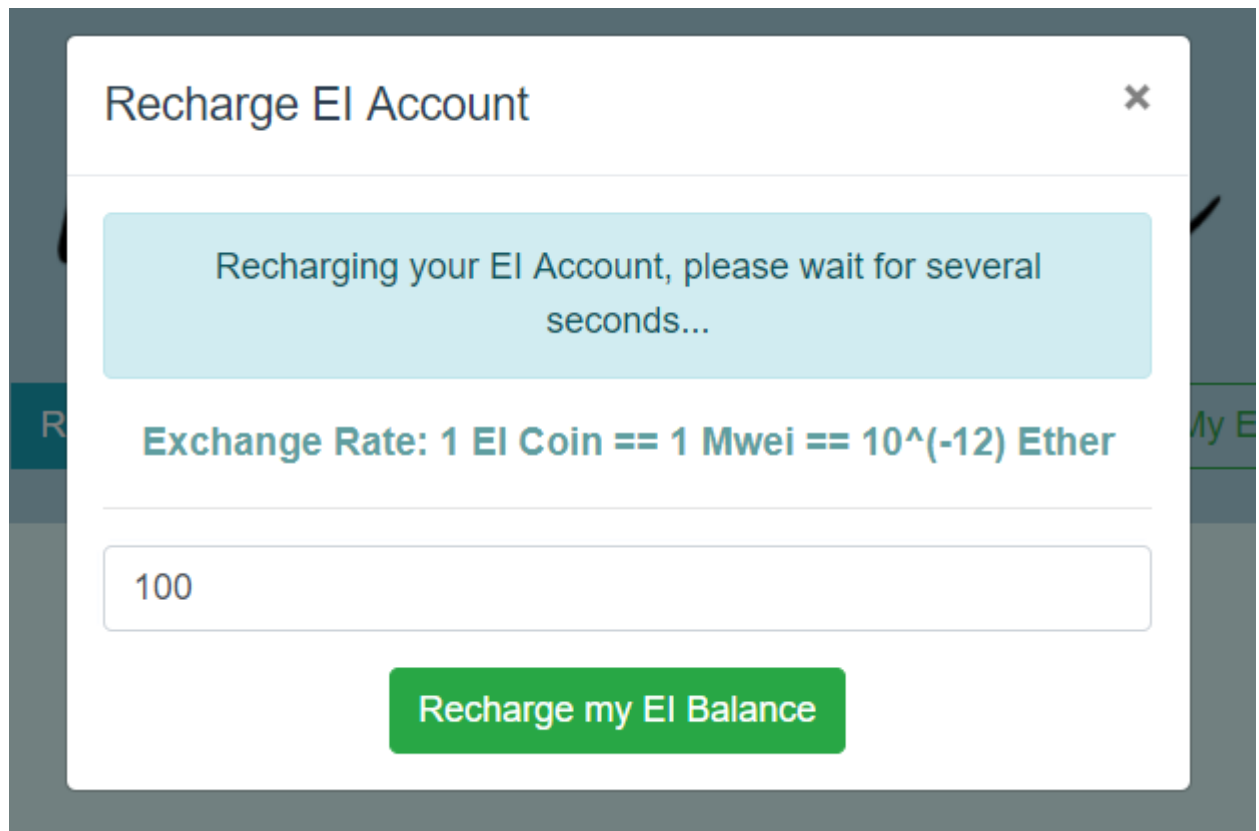
**Price: 50**

Transaction time: 2018/12/27 上午10:33:32

卖家 **Palette** 的账户余额



4. 第四步：对 EI Coin 的充值功能进行测试，测试用户为 Palette，使用测试链上的以太币对用户的 EI Balance 进行充值，充值金额为 100 EI Coins



查看用户当前余额



5. 项目功能至此测试完成，总体来看测试结果均正确，项目运行逻辑正常，预期功能均成功实现。