**基于区块链的教育证书系统**

目录

[一、前言 2](#_Toc60691534)

[1、电子证书的发展背景 2](#_Toc60691535)

[2、电子证书结合区块链技术需求十分迫切 2](#_Toc60691536)

[二、基于区块链的教育证书系统 3](#_Toc60691537)

[1、基于区块链教育证书系统的架构设计 3](#_Toc60691538)

[2、基于区块链教育证书系统的运作流程： 6](#_Toc60691539)

[3、基于区块链教育证书系统的业务流程： 7](#_Toc60691540)

[三、基于区块链教育证书系统的优点： 8](#_Toc60691541)

[1、不可伪造： 8](#_Toc60691542)

[2、不可篡改： 9](#_Toc60691543)

[3、节约证书成本： 9](#_Toc60691544)

[4、信息可追溯： 9](#_Toc60691545)

[5、确权问题： 9](#_Toc60691546)

[6、隐私安全问题： 10](#_Toc60691547)

[四、基于区块链教育证书系统的代码实现和系统界面截图： 10](#_Toc60691548)

[1、基于区块链教育证书的智能合约代码 10](#_Toc60691549)

[教育证书信息上链（合约调合约—certmain.sol调用Recipient.sol） 10](#_Toc60691550)

[教育证书发布系统（合约调合约—MyDapp.sol调用GameItem.sol） 12](#_Toc60691551)

[2、各高校调用教育证书智能合约代码实现信息上链 16](#_Toc60691552)

[web调用合约实现证书信息上链和存储地址的发送（index.js） 16](#_Toc60691553)

[3、 教育厅基于各高校的教育证书实现签名 41](#_Toc60691554)

[实现教育证书文件的下载（0xfd3ac0e515d4867b6aa6c3947244d1e05953a78515c1aa28d695a8747b4cd5d0.html） 41](#_Toc60691555)

[下载至本地的教育证书文件（cert\_bxy.json） 43](#_Toc60691556)

[教育厅加密签名区块链教育证书文件（encrypt.js） 44](#_Toc60691557)

[签名后的教育证书文件（cert\_bxy.json） 45](#_Toc60691558)

[4、 学生基于学生教育证书发布查询系统查询证书是否发布 46](#_Toc60691559)

[教育证书的证书查询代码（证书查询.html，index.js部分方法） 46](#_Toc60691560)

[5、 验证机构验证教育证书是否有效 51](#_Toc60691561)

[web调用Recipient.sol实现信息校验（upload.html） 51](#_Toc60691562)

[五、基于区块链教育证书系统的有关问题： 59](#_Toc60691563)

[1、教育证书的个人数据在区块链上安全吗？ 59](#_Toc60691564)

[2、为什么区块链教育证书系统要使用区块链技术？ 60](#_Toc60691565)

[3、教育证书能否打印后使用？ 60](#_Toc60691566)

[4、个人能否查询他人的教育证书？ 60](#_Toc60691567)

[5、教育证书文件使用安全吗？他人不能复制我的教育证书文件并像我验证教育证书文件一样假冒验证吗？ 60](#_Toc60691568)

[6、教育证书可以被吊销吗？ 60](#_Toc60691569)

[7、教育证书如何实现个性化设计？ 61](#_Toc60691570)

[六、总结和体会 61](#_Toc60691571)

# 一、前言

## 1、电子证书的发展背景

在传统的教育体系中，所有学历证书，包括学校证书，高等教育证书和其他技能证书都是保存在纸上，把所有这些文件都写在纸上给人们带来了许多弊端。

设想这样一个场景:假如这些纸质证书的文件丢失或被盗。在这种情况下，你可能面临的第一个弊端是他人未经授权的情况下便可使用你的文件。第二个弊端是你需要从一个部门跑到另一个部门去获取文件的副本。但是，如果你能以一种安全可靠的方式将你的文档存储在网上，那将解决很多问题。区块链可以为教育部门提供这种解决方案。在区块链上存储文件可节省纸张和打印成本，一旦这些文档存储在区块链上，它们将永远不会丢失，这些文档也不可能被更改或伪造。

人们接受适当的教育是找工作的前提要求。对于市场上的每份工作、对每个招聘人都有其特定的资格标准。作为一名求职者，你必须出示你的证书来证明你是否有资格被聘用。如今，为了找工作而伪造学历的行为越来越多。由于存储在区块链上的数据是不可篡改的，并且可溯源验证，所以一些欺诈和伪造的案件就可以很容易地在区块链技术的帮助下被解决。另外，这些分布在区块链这种分布式账本上的教育证书可以很容易地与招聘者共享，这样招聘者就可以在聘用或面试前轻松地核实应聘者的教育信息和判断应聘者是否有对应招聘要求的教育资格。区块链技术甚至可以彻底改变整个教育系统。

## 2、电子证书结合区块链技术需求十分迫切

国内目前的电子证书管理平台主要采用集中共享模式，由中心数据库来完成证书的制作、存储、信息查询和交换共享，数据库的拥有者掌握着数据库的访问和更新权限。但由于需要管理的人口众多、社会发展水平不均衡，短时间内把不同地区、部门的公民信息完全集中并实时联网，建立“集中的公民信息库”很难实现，因为建立“集中的公民信息库”需要通过体制改革与机构合并实现数据集中和信息联网，在当前条件下，这在管理和技术上的实现都相当困难，存在跨地区、跨部门应用流程复杂、管理性能不高等问题，无法适应大规模、多样化电子证书管理和验证服务需求。同时集中式数据库并没有有效控制证书信息的保密制度，或者有目的性的指定授权，使其对所有办事机构公开，这样证书持有人的信息没法得到有效保护，被攻击篡改和泄露的风险较大，致使证书可靠性打了折扣。

教育就业作为社会文化传授、传播的窗口，需要实现学生、教育机构以及用人就业单位之间的无缝衔接，以提高教育就业机构的运行效率和透明度。而学生信用体系不完整、未建立历史数据信息链、数据维度有限，导致政府、企业无法获得完整有效信息，这直接导致学生无法便捷、公平地享受应有的服务。学历造假、论文造假、求职简历造假，用人单位、院校缺乏验证手段，蒙受信息不对称产生的损失，降低了学校与企业间、院校与院校间的信任。而要证明“学历证是真的”至少花费百元，动辄数百元。近年来全社会每年用于学历认证的收费也已超过亿元。人们在疑问“在教育部学信网上就能查到我的学历证，为何还要花钱去办一个认证？”。

目前已存在的学历认证基本都在全国高校就业指导中心和教育部留学服务中心以及各地认证机构完成。在我国已建立学历网络查询系统的今天，纸质的学历认证也应逐步淘汰。相关部门应明确网络学历认证与纸质学历认证具有同等效力，应杜绝强制使用纸质学历认证的行为，切实减轻社会负担。而现有学历认证管理中心都是中心化的集中式管理，教育就业机构的运行效率和透明度，可信性都是非常严峻的考验。一旦内部出现权利滥用的问题，整个系统的信任体系将崩塌。区块链技术基于教育领域有着不可估量的潜力。区块链在教育方面最大的优势是将教育证书以数字形式存储，安全可靠。区块链系统的透明化、数据不可篡改等特征，完全适用于学生征信管理、升学就业、学术、资质证明、产学合作等方面，对教育就业的健康发展具有重要的价值。

# 二、基于区块链的教育证书系统

## 1、基于区块链教育证书系统的架构设计

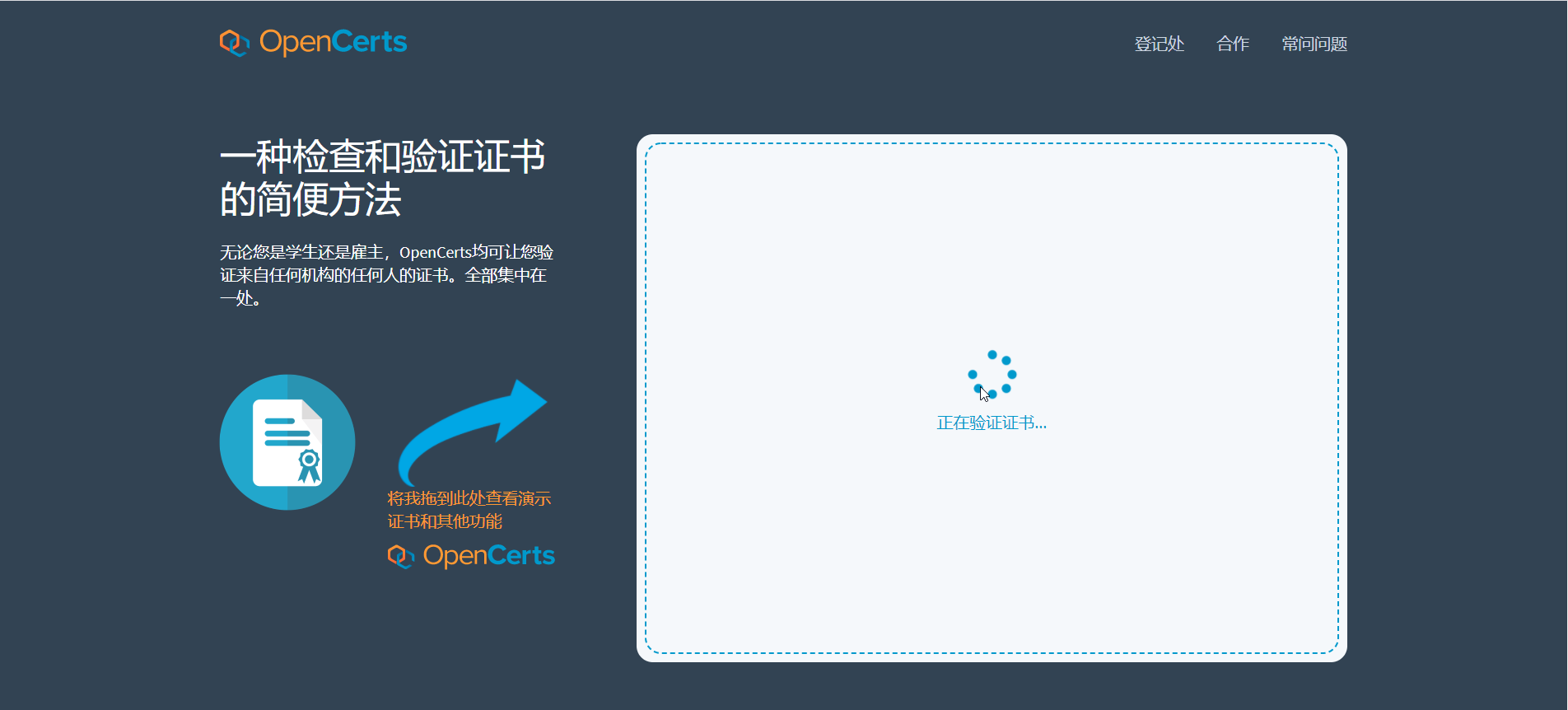
基于区块链的教育证书系统是一个用于发布教育证书的开源架构，也是一套用于为教育证书生成密码保护的工具。该系统大致由三个部分组成：视图、检查、校验。



**视图简易**：用于查看教育证书信息只需将教育证书文件拖拉上传至系统的检验框，检验完毕即可显示教育证书页面，十分便捷。



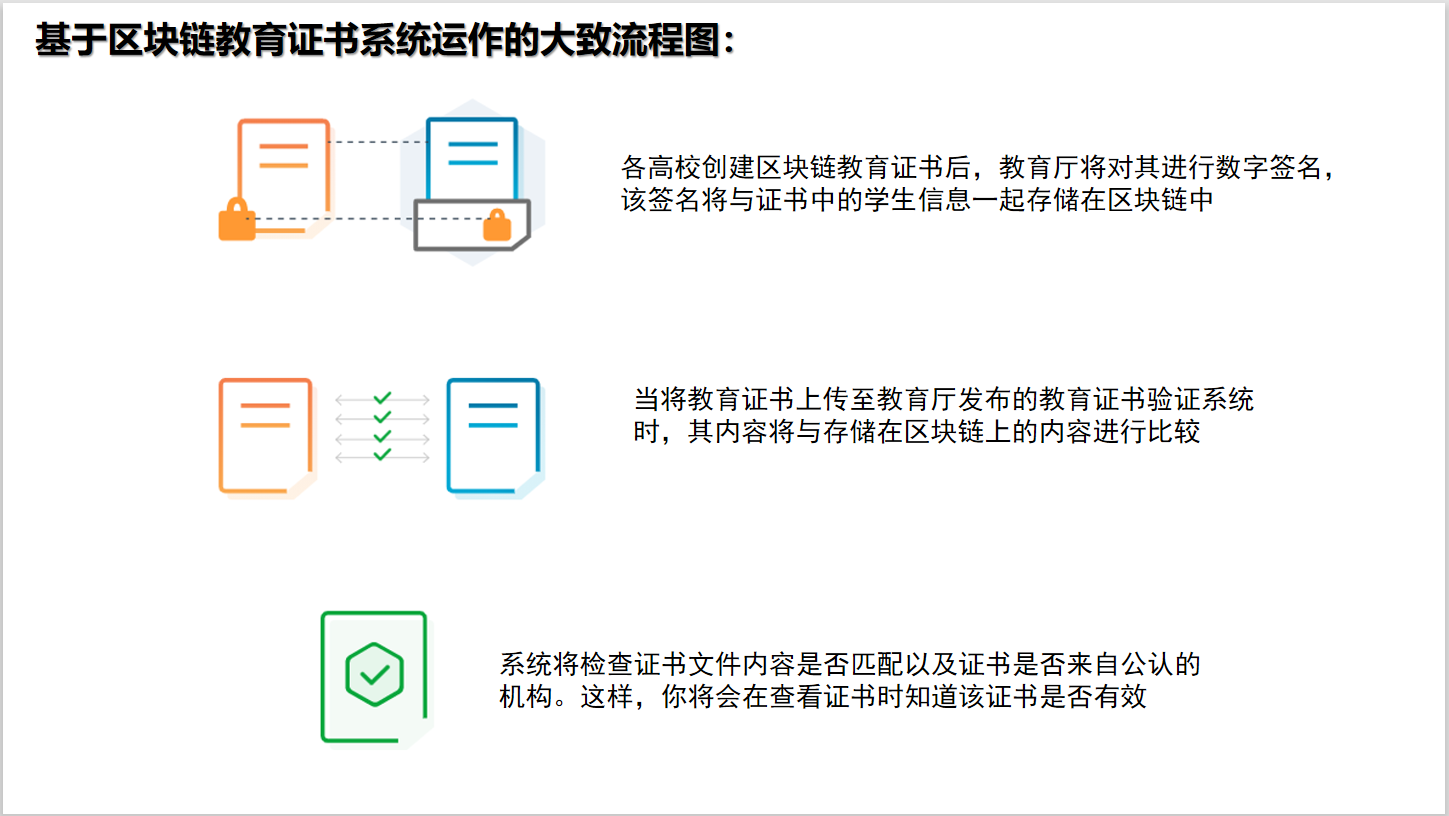
**检查教育证书文件是否被篡改**：基于区块链教育证书系统可根据内置在教育证书文件的高级密码保护自动检查拖拉上传至系统检验框的教育证书文件，将教育证书文件内的哈希回执和文件信息与上链信息相比较，从而达到检查教育证书文件是否有效的目的。



**校验**：通过教育证书的数字签名来判断该上传教育证书文件是否被公认机构认证，如若该教育证书的认证机构不是在公认机构范围内，该教育证书即是无效的。若该教育证书既未被篡改，又来自于公认的机构，基于区块链的教育证书系统将读取教育证书文件内的信息，将教育证书渲染并显示于页面。



## 2、基于区块链教育证书系统的运作流程：



**①各高校创建区块链教育证书后，教育厅将对其进行数字签名，该签名将与证书中的学生信息一起存储在区块链中。**

各高校均需要申请一个被教育厅所认可的钱包账户（只有通过教育厅认证成功后的钱包账户才可部署基于区块链教育证书的智能合约）。此后各高校将会在每一学年将学生的基本信息和学习记录上链。当毕业季来临时，各高校将用申请成功后的钱包账户部署基于区块链教育证书的智能合约。基于区块链教育证书的合约成功部署后，系统将自动生成包含学生基本信息、学生在校学习记录、对应学生合约回执和生成教育文件时间戳的教育证书文件。各高校的学生教育证书信息录入成功后，基于教育证书审核的智能合约将自动执行。教育厅通过接口对学生教育证书进行数字签名，并且该签名将与证书中的学生信息一起存储在区块链中。

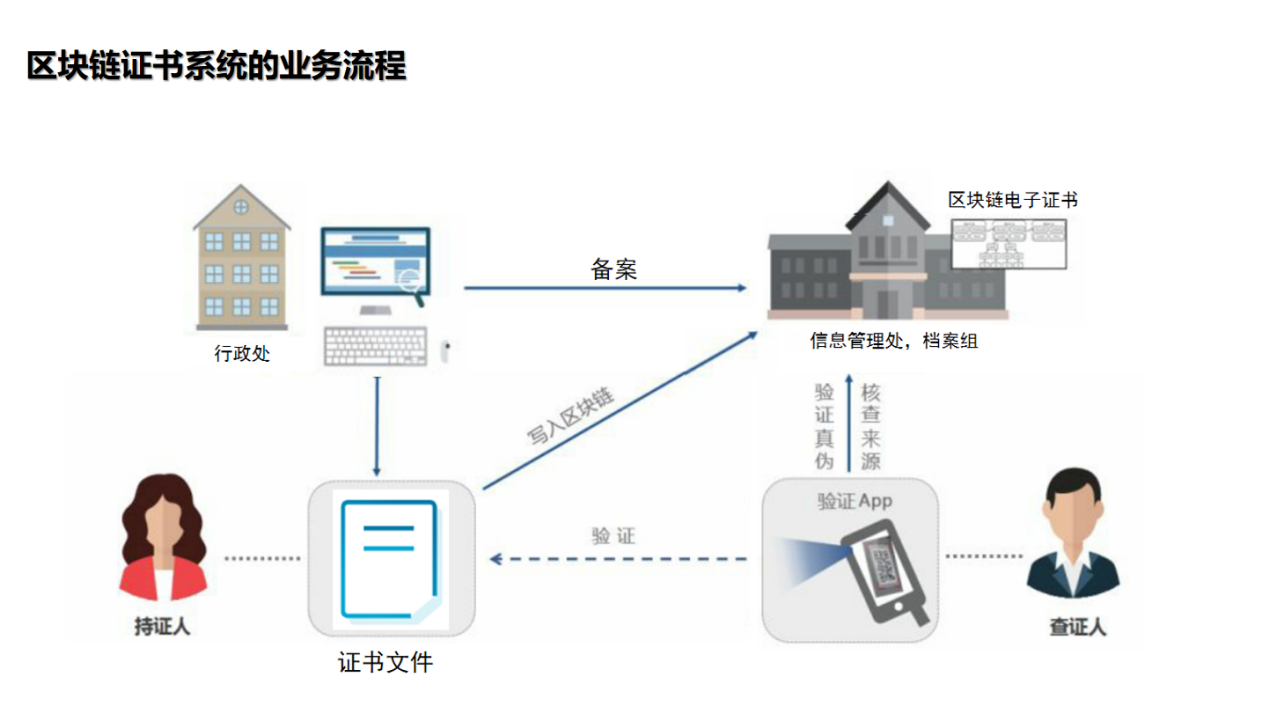
**②当将教育证书上传至教育厅发布的教育证书查询系统后，学生可通过区块链钱包账户登入系统得到教育证书文件，实现教育证书确权。**

各高校每名学生都需要创建一个属于自己的区块链钱包账户，教育厅也将搭建一个基于各高校的教育证书验证系统，在教育证书验证系统中有一个教育证书查询系统窗口。教育证书查询系统是用于各高校学生区块链钱包账户登入后查询并获得区块链教育证书文件，教育证书验证系统是用于教育证书文件的防伪验证。在上一步中教育厅对学生教育证书进行数字签名完成后，教育厅随后会将包含此签名的区块链学生教育证书上传至教育证书查询系统中。并且当教育厅将区块链学生教育证书上传至教育证书查询系统成功后，教育厅将无权限对学生区块链教育证书进行任何操作，换句话来说，此时的学生区块链教育证书已被确权，所有权都归与教育证书所关联的私钥所有者也就是学生本人所有。

**③当将教育证书上传至教育厅发布的教育证书验证系统时，其内容与存储在区块链上的内容进行比较，系统将检查证书文件内容是否匹配，验证证书是否有效。**

当学生毕业后想查询个人毕业证书时，只需进入教育厅的教育证书查询系统，用第二步创建的区块链钱包账户登录教育证书查询系统查询证书，如若证书已发布，学生可手动将教育证书文件下载至本地。当需要验证区块链教育证书是否被篡改时，只需将下载后的教育证书文件拖拉上传至教育证书验证系统的验证框中。此时教育证书验证系统将会将自动读取上传的教育证书文件，通过读取上传教育证书文件中的合约回执进行区块链信息溯源。合约回执若为无效哈希，系统将无法通过验证，并显示为已被篡改。若合约回执正确，系统会将上链信息与上传教育证书文件中所有信息进行逐一比对。如若上传教育证书文件中的信息与上链信息一致，系统将会通过自动读取上传教育证书文件中的信息，从而渲染并显示出区块链教育证书页面。如若上传教育证书文件中的信息与上链信息不一致，则表明教育证书文件已被篡改，系统将会显示“教育证书已被篡改，无法读取证书信息”的错误提示。

## 3、基于区块链教育证书系统的业务流程：



各高校将学生的基本信息和在校学习记录上链并部署智能合约生成教育证书，教育厅对学生教育证书进行数字签名并发布于教育证书查询系统中。当学生毕业后出去应聘公司时，难免需要出示毕业证书以表明学历成绩是否符合公司招聘标准。如果使用区块链教育证书系统，只需使用个人区块链钱包账户登入系统，即可得到个人的区块链教育证书文件，再将区块链教育证书文件拖拉至系统验证框，即可得到区块链教育证书页面。如若教育证书中有应聘要求之外的学习科目并且学生该成绩恰好分数较低，此时学生可能认为此成绩对应聘将会造成影响，区块链教育证书系统完美地解决了这一问题。该系统支持隐藏部分学习成绩的功能，学生可根据自身形势的需求，调整证书的显示信息，实现个性化需求。隐藏部分信息后的教育证书文件可下载保存。当把此文件拖拉上传至教育证书验证系统的验证框时，教育证书页面将显示学生设计后的证书信息页面。如若他人想查询学生的教育证书信息，首先需要得到学生同意，学生将教育证书文件通过邮件或者微信等传输方式发送给他人，他人即可拿着这份教育证书文件进入教育证书验证系统查询此教育证书信息。但由于查询的教育证书和登入教育证书验证系统的区块链钱包账户不匹配，故每次查询他人教育证书信息都将支付一笔费用给教育证书所有者。并且教育证书所有者可得知自己教育证书被查询的次数和查询方账户，实现查询的信息透明。

所以如果使用区块链教育证书系统，对于证书发布方来说：可节省大量打印成本、纸张成本、信息整理的人工成本；对于证书被授予方来说：无需随身携带各类教育信息，也无需担心教育证书信息丢失，可省略从一个部门到另外部门去盖章或者证书丢失补办证书的相应流程，节省大量时间和精力，还可以在他人查询自身教育证书信息时得到相应的支付报酬并且可得知教育证书所有的被查询信息记录，实现证书确权和查询信息透明的功能；对于证书检验方（就比如上面所说的公司招聘检查官）来说：不必担心证书是否篡改等不信任因素，可在需要的时候多次查询公司成员教育信息。

# 三、基于区块链教育证书系统的优点：



## 1、不可伪造：

1、各高校部署合约所用区块链账户钱包需被教育厅所认可。

2、各高校将学生的基本信息和学习记录上链（例如：高校可在每一学年将学生学习成绩和相应学习记录上链，上链后信息不可篡改并具有对应时间戳，为之后的信息溯源提供保障），毕业季高校部署合约后系统将自动生成包含学生基本信息、学生在校学习记录、对应学生合约回执和生成教育文件时间戳的教育证书文件。

3、各高校的学生教育证书信息录入成功后，基于教育证书审核的智能合约将自动执行，通过接口教育厅对学生教育证书进行数字签名并发布于对应学生账户系统中。

4、当教育厅签名学生教育证书并且发布教育证书后，智能合约将撤销各高校和教育厅对学生教育证书的所有权力。

## 2、不可篡改：

1、由于各高校学生信息的上链是由相关部门在每一定时间段上链，上链信息将会成为历史记录，记录成为一个事实。如若有人篡改证书文件信息，溯源信息时间戳将无法对上。

2、教育厅发布的教育证书验证系统将自动读取上传的教育证书文件，通过读取教育证书文件中的合约回执进行区块链信息溯源。合约回执若为无效哈希，系统无法通过验证，系统将会显示“教育证书已被篡改，无法读取证书信息”的错误提示。若合约回执正确，系统会将上链信息与上传的教育证书文件信息进行逐一比对，信息一旦被更改，与上链信息不符，系统也将无法通过验证。

## 3、节约证书成本：

1. 相比于传统证书，基于区块链的教育证书可节省大量打印成本、纸张成本、信息整理的人工成本。
2. 发布区块链教育证书可省略从一个部门到另外部门去盖章或者证书丢失补办证书的相应流程，为证书发布者和证书被授予者节约时间和精力。

3、持证人需要展示教育信息时只需用自身的区块链钱包账户登录教育证书查询系统即可查阅包含学生信息和学生学习记录的电子证书，无需随身携带各类教育信息。

## 4、信息可追溯：

区块链教育证书在上链过程中的每一笔记录都是有对应哈希回执和时间戳记录的，学生基本信息、学生在校学习记录、对应学生合约回执、生成教育证书文件和教育厅数字签名等信息全部记录于区块链中，所以区块链教育证书中的信息和时间戳可随时通过哈希回执在区块链上查询得到。

## 5、确权问题：

教育证书一旦发布，教育证书所有权归证书被授予者一人所有。授予者用区块链钱包账户（自身私钥）登入教育证书验证系统中的教育证书查询系统查询自己的教育证书文件，教育证书文件可手动下载保存至本地。此时将下载的教育证书文件拖拉上传至教育证书验证系统的验证框中，教育证书验证系统将自动处理上传的证书文件，此时方可查询被授予者自身的教育证书信息。若想要访问他人教育证书信息，首先需要获得他人的教育证书文件，然后用自身的区块链钱包登入教育证书验证系统，由于上传的教育证书所关联的区块链钱包账户和登入教育证书验证系统的区块链钱包账户不匹配，故每一次都需要支付相应数字货币才可查询他人教育证书文件，并且教育证书所有者可查询自身教育证书被访问的一切记录。

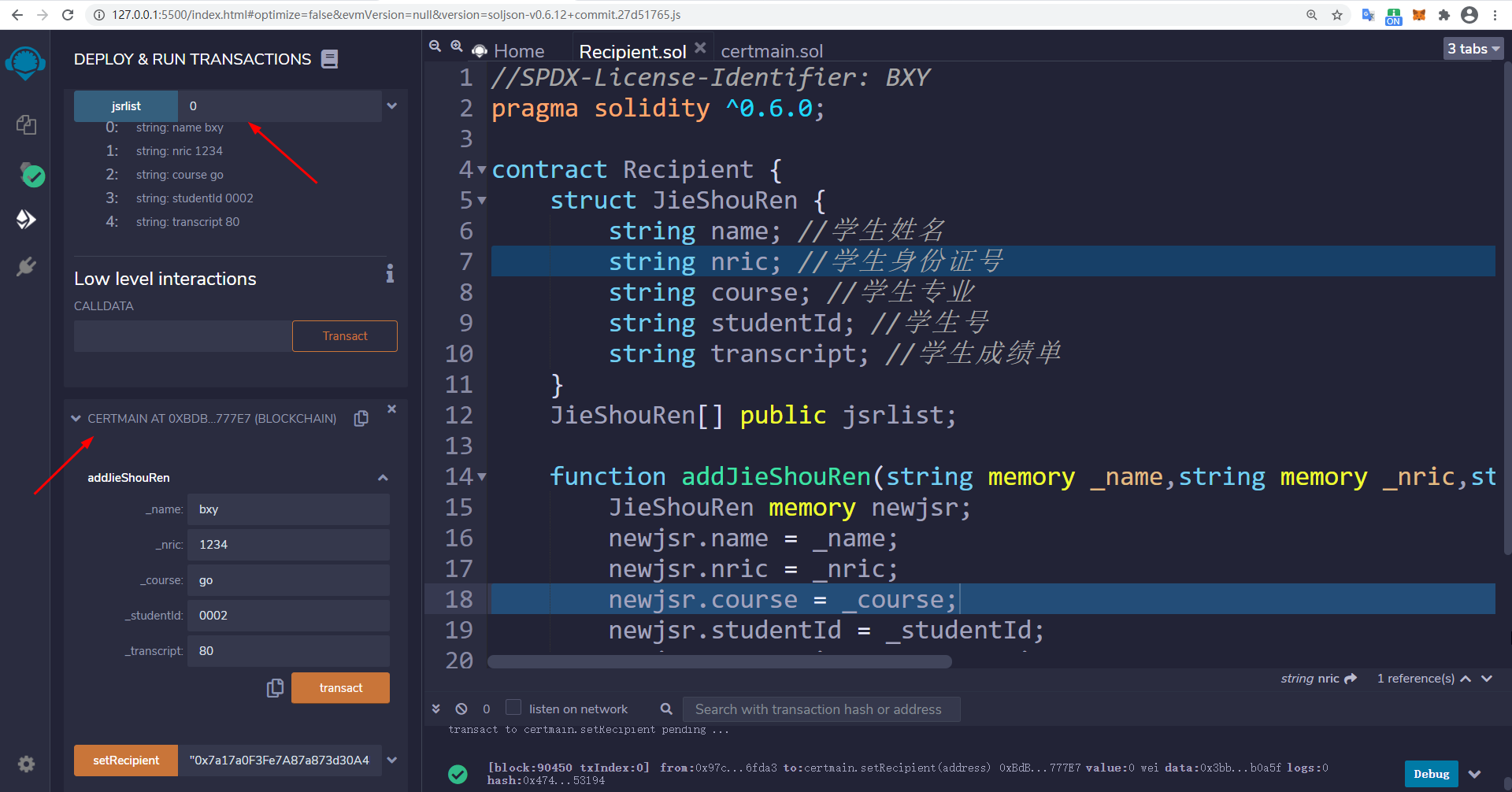
## 6、隐私安全问题：

1. 一旦教育证书存储在区块链上，它们将永远不会丢失。
2. 教育证书查询系统是由学生区块链钱包账户私钥登录，私钥是学生自己保存，所以一般他人无法查看自身的教育证书文件。若想让他人查看自身的教育证书文件，学生可使用教育证书查询系统中查看证书时可见的共享按钮或者通过电子邮件和微信等其他传输方式将教育证书文件发送给其他人。
3. 教育厅发布的教育证书验证系统支持学生对自己教育证书的个性化设计，可实现学生有选择性地展示证书信息的需求。例如：某公司招聘人员的要求中无需具备外语基础，此时学生可以选择不展示有关外语成绩，从而得到更有利于自己的教育证书格式。隐藏部分信息后的教育证书文件可下载保存。当把此文件上传至教育证书验证系统时，教育证书页面将显示学生设计后的证书信息，实现隐私安全功能。

# 四、基于区块链教育证书系统的代码实现和系统界面截图：

## 1、基于区块链教育证书的智能合约代码

## 教育证书信息上链（合约调合约—certmain.sol调用Recipient.sol）



Recipient.sol：

//SPDX-License-Identifier: BXY

pragma solidity ^0.6.0;

contract Recipient {

    struct JieShouRen {

        string name; //学生姓名

        string nric; //学生身份证号

        string course; //学生专业

        string studentId; //学生号

        string transcript; //学生成绩单

    }

    JieShouRen[] public jsrlist;

    function addJieShouRen(string memory \_name,string memory \_nric,string memory \_course,string memory \_studentId,string memory \_transcript) public {

        JieShouRen memory newjsr;

        newjsr.name = \_name;

        newjsr.nric = \_nric;

        newjsr.course = \_course;

        newjsr.studentId = \_studentId;

        newjsr.transcript = \_transcript;

        jsrlist.push(newjsr);

    }

}

certmain.sol：

//SPDX-License-Identifier: BXY

pragma solidity ^0.6.0;

pragma experimental ABIEncoderV2;

import "./Recipient.sol";

contract certmain{

    Recipient \_recipient;

    struct JieShouRen {

        string name; //学生姓名

        string nric; //学生身份证号

        string course; //学生专业

        string studentId; //学生号

        string transcript; //学生成绩

    }

    function setRecipient(Recipient addr) public {

        \_recipient = addr;

    }

    function addJieShouRen(string memory \_name,string memory \_nric,string memory \_course,string memory \_studentId,string memory \_transcript) public {

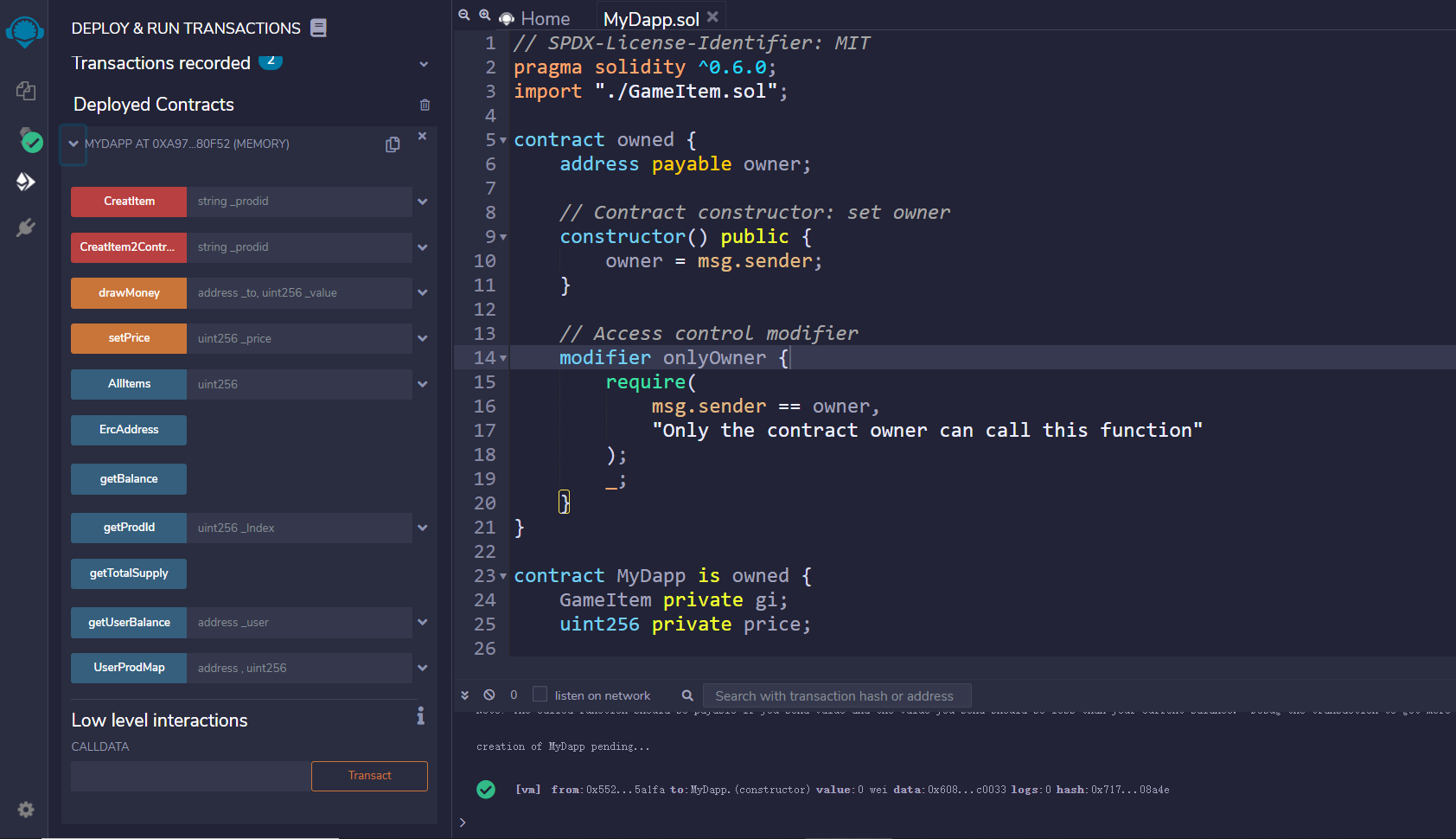
       return \_recipient.addJieShouRen(\_name,\_nric,\_course,\_studentId,\_transcript);

    }

}

­­­­­­

## 教育证书发布系统（合约调合约—MyDapp.sol调用GameItem.sol）



MyDapp.sol：

// SPDX-License-Identifier: MIT

pragma solidity ^0.6.0;

import "./GameItem.sol";

contract owned {

    address payable owner;

    // Contract constructor: set owner

    constructor() public {

        owner = msg.sender;

    }

    // Access control modifier

    modifier onlyOwner {

        require(

            msg.sender == owner,

            "Only the contract owner can call this function"

        );

        \_;

    }

}

contract MyDapp is owned {

    GameItem private gi;

    uint256 private price;

    // struct ProudItem {

    //     string \_proudid;

    // }

    mapping(address => string[]) public UserProdMap;

    string[] public AllItems;

    constructor() public {

        gi = new GameItem();

        price = 0.01 ether;

    }

    function setPrice(uint256 \_price) public {

        price = \_price;

    }

    function CreatItem(string memory \_prodid) public payable {

        require(msg.value >= price, "at least 0.1 Eth");

        gi.awardItem(msg.sender, \_prodid);

        AllItems.push(\_prodid);

        string[] storage user\_Prod;

        user\_Prod = UserProdMap[msg.sender];

        user\_Prod.push(\_prodid);

        UserProdMap[msg.sender] = user\_Prod;

    }

    function CreatItem2Contract(string memory \_prodid) public payable {

        require(msg.value >= price, "at least 0.1 Eth");

        gi.awardItem(address(this), \_prodid);

        AllItems.push(\_prodid);

        string[] storage user\_Prod;

        user\_Prod = UserProdMap[msg.sender];

        user\_Prod.push(\_prodid);

        UserProdMap[msg.sender] = user\_Prod;

    }

    function getTotalSupply() public view returns (uint256) {

        return gi.totalSupply();

    }

    function getUserBalance(address \_user) public view returns (uint256) {

        return gi.balanceOf(\_user);

    }

    function getProdId(uint256 \_Index)

        public

        view

        returns (string memory r\_token, uint256 r\_index)

    {

        uint256 \_token = gi.tokenOfOwnerByIndex(msg.sender, \_Index);

        return (gi.tokenURI(\_token), \_token);

    }

    function ErcAddress() public view returns (GameItem) {

        return gi;

    }

    // 合约往账户转账

    function drawMoney(address \_to, uint256 \_value) public onlyOwner {

        address(uint160(\_to)).transfer(\_value);

        return;

    }

    // 得到合约里面的eth余额

    function getBalance() public view returns (uint256) {

        return address(this).balance;

    }

    //   function getLast10Item(address \_to,string memory \_proudid) public {

    //     uint256 totleitem = gi.totalSupply();

    //         for (uint i = 0; i < totleitem; i++) {

    //     }

    // }

}

GameItem.sol：

// SPDX-License-Identifier: MIT

pragma solidity ^0.6.0;

import "https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/token/ERC721/ERC721.sol";

import "https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/utils/Counters.sol";

contract GameItem is ERC721 {

    using Counters for Counters.Counter;

    Counters.Counter private \_tokenIds;

    constructor() public ERC721("GameItem", "ITM") {}

    function awardItem(address player, string memory tokenURI)

        public

        returns (uint256)

    {

        \_tokenIds.increment();

        uint256 newItemId = \_tokenIds.current();

        \_mint(player, newItemId);

        \_setTokenURI(newItemId, tokenURI);

        return newItemId;

    }

}

## 2、各高校调用教育证书智能合约代码实现信息上链

## web调用合约实现证书信息上链和存储地址的发送（index.js）

console.log("Hello")

let accounts = [];

if (typeof window.ethereum !== 'undefined') {

    console.log('MetaMask is installed!');

}

console.log("isMetaMask:" + ethereum.isMetaMask)

$(".enableEthereumButton").click(function () {

    getAccount()

}

)

async function getAccount() {

    accounts = await ethereum.request({ method: 'eth\_requestAccounts' });

    const account = accounts[0];

    $(".showAccount").html(account);

}

ethereum.on('accountsChanged', function (accounts) {

    console.log("accountsChanged");

    getAccount()

});

ethereum.on('chainChanged', (chainId) => {

    console.log("chainId ", chainId);

});

let web3 = new Web3(Web3.givenProvider || "ws://localhost:8546");

console.log("web3", web3)

var certmain\_abi = [

    {

        "inputs": [

            {

                "internalType": "string",

                "name": "\_name",

                "type": "string"

            },

            {

                "internalType": "string",

                "name": "\_nric",

                "type": "string"

            },

            {

                "internalType": "string",

                "name": "\_course",

                "type": "string"

            },

            {

                "internalType": "string",

                "name": "\_studentId",

                "type": "string"

            },

            {

                "internalType": "string",

                "name": "\_transcript",

                "type": "string"

            }

        ],

        "name": "addJieShouRen",

        "outputs": [],

        "stateMutability": "nonpayable",

        "type": "function"

    },

    {

        "inputs": [

            {

                "internalType": "contract Recipient",

                "name": "addr",

                "type": "address"

            }

        ],

        "name": "setRecipient",

        "outputs": [],

        "stateMutability": "nonpayable",

        "type": "function"

    }

];

const certmain\_address = '0xf056c3EeC006EecDF209449ad351a9AbC587b034'

const Recipient\_address = '0x5739Dd0b6554A34CBa33CE8bf10f57ECE1608884'

const certmain\_contract = new web3.eth.Contract(certmain\_abi, certmain\_address)

$(".submit").click(function () {

    $("#load\_ci\_img").show();

    let name = $("#prodid").val()

    let nric = $("#nric").val()

    let course = $("#course").val()

    let studentId = $("#studentId").val()

    let transcript = $("#transcript").val()

    certmain\_contract.methods.setRecipient(Recipient\_address).send({ from: accounts[0] }).then(function (result1) {

        certmain\_contract.methods.addJieShouRen(name, nric, course, studentId, transcript).send({ from: accounts[0] }).then(function (result2) {

            console.log("主合约连接副合约==>", result1);

            console.log("向副合约添加信息==>", result2);

            if (result1.status && result2.status) {

                $(".txt\_CreatItem").html("信息上链成功")

                // 写入证书

                const schema = "opencerts/v2.0";

                const deploy = Recipient\_address;

                const transaction\_hash = result2.transactionHash;

                const data = { recipient: { name: name, nric: nric, course: course, studentId: studentId, transcript: transcript } }

                var content, btn;

                content = document.querySelector("#content");

                btn = document.querySelector("#save-btn");

                content.addEventListener("change", function () {

                    var f = function () {

                        btn.setAttribute("href", "data:text/paint; utf-8," + JSON.stringify({ schema, deploy, transaction\_hash, data }));

                    }

                    return f(), f;

                }());

                $("#save-btn").show();

                let \_txt = result2.transactionHash

                contract.methods.CreatItem(\_txt).send({ from: accounts[0], value: 0.02 \* Math.pow(10, 18) }).then(

                    function (result) {

                        console.log("result", result);

                        if (result.status) {

                            $(".txt\_CreatItem").html("文件地址绑定钱包成功")

                            getBalance();

                            // window.location.href = "http://127.0.0.1:5502/%E5%90%84%E9%AB%98%E6%A0%A1%E4%BB%A3%E7%A0%81/%E4%B8%BB%E9%A1%B5.html";

                        } else {

                            hide

                            $(".txt\_CreatItem").html("文件地址绑定钱包失败")

                        }

                        // $("#load\_ci\_img").hide();

                    }

                )

            } else {

                hide

                $(".txt\_CreatItem").html("信息上链失败")

            }

            $("#load\_ci\_img").hide();

        })

    })

}

)

async function getProdID(\_index) {

    contract.methods.getProdId(\_index).call({ from: accounts[0] }).then(

        function (result) {

            console.log("getProdID", result)

            $("#pitem" + \_index).html(result.r\_token);

            $("#tokenIndex" + \_index).html(result.r\_index);

        }

    );

}

async function getAccount() {

    accounts = await ethereum.request({ method: 'eth\_requestAccounts' });

    const account = accounts[0];

    // showAccount.innerHTML = account;

    $(".showAccount").html(account);

    getBalance();

    ErcAddress();

}

async function ErcAddress() {

    contract.methods.ErcAddress().call({ from: accounts[0] }).then(

        function (result) {

            console.log("ErcAddress==》", result)

            $(".ErcAddress").html(result)

        }

    )

}

function getBalance() {

    contract.methods.getUserBalance(accounts[0]).call({ from: accounts[0] }).then(

        function (result) {

            $('#box').children().remove();

            console.log('getUserBalance', result)

            $("#MyBalance").html(result)

            for (var i = 0; i < result; i++) {

                console.log(i + "========================")

                // var el = $('<h4>证书地址:<span id="pitem' + i + '">-- </span>#<span id="tokenIndex' + i + '"></span>' + '<br>现将证书授权于Address:<input id="s\_address' + i + '" value="" /><button id="Btn\_send' + i + '"  onClick="sendBtn(' + i + ')">发送</button></h4>');

                var el = $('<div>证书地址:<span id="pitem' + i + '">-- </span>#<span id="tokenIndex' + i + '"></span>' + '|接收人Address:<input id="s\_address' + i + '" value="" /><button id="Btn\_send' + i + '"  onClick="sendBtn(' + i + ')">发送</button></div>');

                getProdID(i);

                $('#box').append(el);

            }

        }

    );

}

function sendBtn(i) {

    \_address = $('#s\_address' + i).val();

    p\_index = $('#tokenIndex' + i).html();

    alert('send' + i + "#" + \_address + "#" + p\_index);

    // transferProd

    getErc721($(".ErcAddress").html());

    contractERC721.methods.transferFrom(accounts[0], \_address, p\_index).send({ from: accounts[0] }).then(

        function (result) {

            console.log("transferProd==>", result);

            if (result.status) {

                $(".txt\_CreatItem").html("添加成功")

                getBalance();

            } else {

                hide

                $(".txt\_CreatItem").html("添加失败")

            }

            $("#load\_ci\_img").hide();

        }

    )

}

ethereum.on('accountsChanged', function (accounts) {

    console.log("accountsChanged");

    getAccount()

});

ethereum.on('chainChanged', (chainId) => {

    console.log("chainId", chainId);

});

var contractERC712Abi = [

    {

        "inputs": [],

        "stateMutability": "nonpayable",

        "type": "constructor"

    },

    {

        "anonymous": false,

        "inputs": [

            {

                "indexed": true,

                "internalType": "address",

                "name": "owner",

                "type": "address"

            },

            {

                "indexed": true,

                "internalType": "address",

                "name": "approved",

                "type": "address"

            },

            {

                "indexed": true,

                "internalType": "uint256",

                "name": "tokenId",

                "type": "uint256"

            }

        ],

        "name": "Approval",

        "type": "event"

    },

    {

        "anonymous": false,

        "inputs": [

            {

                "indexed": true,

                "internalType": "address",

                "name": "owner",

                "type": "address"

            },

            {

                "indexed": true,

                "internalType": "address",

                "name": "operator",

                "type": "address"

            },

            {

                "indexed": false,

                "internalType": "bool",

                "name": "approved",

                "type": "bool"

            }

        ],

        "name": "ApprovalForAll",

        "type": "event"

    },

    {

        "anonymous": false,

        "inputs": [

            {

                "indexed": true,

                "internalType": "address",

                "name": "from",

                "type": "address"

            },

            {

                "indexed": true,

                "internalType": "address",

                "name": "to",

                "type": "address"

            },

            {

                "indexed": true,

                "internalType": "uint256",

                "name": "tokenId",

                "type": "uint256"

            }

        ],

        "name": "Transfer",

        "type": "event"

    },

    {

        "inputs": [

            {

                "internalType": "address",

                "name": "to",

                "type": "address"

            },

            {

                "internalType": "uint256",

                "name": "tokenId",

                "type": "uint256"

            }

        ],

        "name": "approve",

        "outputs": [],

        "stateMutability": "nonpayable",

        "type": "function"

    },

    {

        "inputs": [

            {

                "internalType": "address",

                "name": "player",

                "type": "address"

            },

            {

                "internalType": "string",

                "name": "tokenURI",

                "type": "string"

            }

        ],

        "name": "awardItem",

        "outputs": [

            {

                "internalType": "uint256",

                "name": "",

                "type": "uint256"

            }

        ],

        "stateMutability": "nonpayable",

        "type": "function"

    },

    {

        "inputs": [

            {

                "internalType": "address",

                "name": "owner",

                "type": "address"

            }

        ],

        "name": "balanceOf",

        "outputs": [

            {

                "internalType": "uint256",

                "name": "",

                "type": "uint256"

            }

        ],

        "stateMutability": "view",

        "type": "function"

    },

    {

        "inputs": [],

        "name": "baseURI",

        "outputs": [

            {

                "internalType": "string",

                "name": "",

                "type": "string"

            }

        ],

        "stateMutability": "view",

        "type": "function"

    },

    {

        "inputs": [

            {

                "internalType": "uint256",

                "name": "tokenId",

                "type": "uint256"

            }

        ],

        "name": "getApproved",

        "outputs": [

            {

                "internalType": "address",

                "name": "",

                "type": "address"

            }

        ],

        "stateMutability": "view",

        "type": "function"

    },

    {

        "inputs": [

            {

                "internalType": "address",

                "name": "owner",

                "type": "address"

            },

            {

                "internalType": "address",

                "name": "operator",

                "type": "address"

            }

        ],

        "name": "isApprovedForAll",

        "outputs": [

            {

                "internalType": "bool",

                "name": "",

                "type": "bool"

            }

        ],

        "stateMutability": "view",

        "type": "function"

    },

    {

        "inputs": [],

        "name": "name",

        "outputs": [

            {

                "internalType": "string",

                "name": "",

                "type": "string"

            }

        ],

        "stateMutability": "view",

        "type": "function"

    },

    {

        "inputs": [

            {

                "internalType": "uint256",

                "name": "tokenId",

                "type": "uint256"

            }

        ],

        "name": "ownerOf",

        "outputs": [

            {

                "internalType": "address",

                "name": "",

                "type": "address"

            }

        ],

        "stateMutability": "view",

        "type": "function"

    },

    {

        "inputs": [

            {

                "internalType": "address",

                "name": "from",

                "type": "address"

            },

            {

                "internalType": "address",

                "name": "to",

                "type": "address"

            },

            {

                "internalType": "uint256",

                "name": "tokenId",

                "type": "uint256"

            }

        ],

        "name": "safeTransferFrom",

        "outputs": [],

        "stateMutability": "nonpayable",

        "type": "function"

    },

    {

        "inputs": [

            {

                "internalType": "address",

                "name": "from",

                "type": "address"

            },

            {

                "internalType": "address",

                "name": "to",

                "type": "address"

            },

            {

                "internalType": "uint256",

                "name": "tokenId",

                "type": "uint256"

            },

            {

                "internalType": "bytes",

                "name": "\_data",

                "type": "bytes"

            }

        ],

        "name": "safeTransferFrom",

        "outputs": [],

        "stateMutability": "nonpayable",

        "type": "function"

    },

    {

        "inputs": [

            {

                "internalType": "address",

                "name": "operator",

                "type": "address"

            },

            {

                "internalType": "bool",

                "name": "approved",

                "type": "bool"

            }

        ],

        "name": "setApprovalForAll",

        "outputs": [],

        "stateMutability": "nonpayable",

        "type": "function"

    },

    {

        "inputs": [

            {

                "internalType": "bytes4",

                "name": "interfaceId",

                "type": "bytes4"

            }

        ],

        "name": "supportsInterface",

        "outputs": [

            {

                "internalType": "bool",

                "name": "",

                "type": "bool"

            }

        ],

        "stateMutability": "view",

        "type": "function"

    },

    {

        "inputs": [],

        "name": "symbol",

        "outputs": [

            {

                "internalType": "string",

                "name": "",

                "type": "string"

            }

        ],

        "stateMutability": "view",

        "type": "function"

    },

    {

        "inputs": [

            {

                "internalType": "uint256",

                "name": "index",

                "type": "uint256"

            }

        ],

        "name": "tokenByIndex",

        "outputs": [

            {

                "internalType": "uint256",

                "name": "",

                "type": "uint256"

            }

        ],

        "stateMutability": "view",

        "type": "function"

    },

    {

        "inputs": [

            {

                "internalType": "address",

                "name": "owner",

                "type": "address"

            },

            {

                "internalType": "uint256",

                "name": "index",

                "type": "uint256"

            }

        ],

        "name": "tokenOfOwnerByIndex",

        "outputs": [

            {

                "internalType": "uint256",

                "name": "",

                "type": "uint256"

            }

        ],

        "stateMutability": "view",

        "type": "function"

    },

    {

        "inputs": [

            {

                "internalType": "uint256",

                "name": "tokenId",

                "type": "uint256"

            }

        ],

        "name": "tokenURI",

        "outputs": [

            {

                "internalType": "string",

                "name": "",

                "type": "string"

            }

        ],

        "stateMutability": "view",

        "type": "function"

    },

    {

        "inputs": [],

        "name": "totalSupply",

        "outputs": [

            {

                "internalType": "uint256",

                "name": "",

                "type": "uint256"

            }

        ],

        "stateMutability": "view",

        "type": "function"

    },

    {

        "inputs": [

            {

                "internalType": "address",

                "name": "from",

                "type": "address"

            },

            {

                "internalType": "address",

                "name": "to",

                "type": "address"

            },

            {

                "internalType": "uint256",

                "name": "tokenId",

                "type": "uint256"

            }

        ],

        "name": "transferFrom",

        "outputs": [],

        "stateMutability": "nonpayable",

        "type": "function"

    }

];

var contractERC721;

function getErc721(conAddress) {

    contractERC721 = new web3.eth.Contract(contractERC712Abi, conAddress);

    console.log('contractERC721==>', contractERC721);

}

// MyApp.sol abi

var contractAbi = [

    {

        "inputs": [],

        "stateMutability": "nonpayable",

        "type": "constructor"

    },

    {

        "inputs": [

            {

                "internalType": "uint256",

                "name": "",

                "type": "uint256"

            }

        ],

        "name": "AllItems",

        "outputs": [

            {

                "internalType": "string",

                "name": "",

                "type": "string"

            }

        ],

        "stateMutability": "view",

        "type": "function"

    },

    {

        "inputs": [

            {

                "internalType": "string",

                "name": "\_prodid",

                "type": "string"

            }

        ],

        "name": "CreatItem",

        "outputs": [],

        "stateMutability": "payable",

        "type": "function"

    },

    {

        "inputs": [

            {

                "internalType": "string",

                "name": "\_prodid",

                "type": "string"

            }

        ],

        "name": "CreatItem2Contract",

        "outputs": [],

        "stateMutability": "payable",

        "type": "function"

    },

    {

        "inputs": [],

        "name": "ErcAddress",

        "outputs": [

            {

                "internalType": "contract GameItem",

                "name": "",

                "type": "address"

            }

        ],

        "stateMutability": "view",

        "type": "function"

    },

    {

        "inputs": [

            {

                "internalType": "address",

                "name": "",

                "type": "address"

            },

            {

                "internalType": "uint256",

                "name": "",

                "type": "uint256"

            }

        ],

        "name": "UserProdMap",

        "outputs": [

            {

                "internalType": "string",

                "name": "",

                "type": "string"

            }

        ],

        "stateMutability": "view",

        "type": "function"

    },

    {

        "inputs": [

            {

                "internalType": "address",

                "name": "\_to",

                "type": "address"

            },

            {

                "internalType": "uint256",

                "name": "\_value",

                "type": "uint256"

            }

        ],

        "name": "drawMoney",

        "outputs": [],

        "stateMutability": "nonpayable",

        "type": "function"

    },

    {

        "inputs": [],

        "name": "getBalance",

        "outputs": [

            {

                "internalType": "uint256",

                "name": "",

                "type": "uint256"

            }

        ],

        "stateMutability": "view",

        "type": "function"

    },

    {

        "inputs": [

            {

                "internalType": "uint256",

                "name": "\_Index",

                "type": "uint256"

            }

        ],

        "name": "getProdId",

        "outputs": [

            {

                "internalType": "string",

                "name": "r\_token",

                "type": "string"

            },

            {

                "internalType": "uint256",

                "name": "r\_index",

                "type": "uint256"

            }

        ],

        "stateMutability": "view",

        "type": "function"

    },

    {

        "inputs": [],

        "name": "getTotalSupply",

        "outputs": [

            {

                "internalType": "uint256",

                "name": "",

                "type": "uint256"

            }

        ],

        "stateMutability": "view",

        "type": "function"

    },

    {

        "inputs": [

            {

                "internalType": "address",

                "name": "\_user",

                "type": "address"

            }

        ],

        "name": "getUserBalance",

        "outputs": [

            {

                "internalType": "uint256",

                "name": "",

                "type": "uint256"

            }

        ],

        "stateMutability": "view",

        "type": "function"

    },

    {

        "inputs": [

            {

                "internalType": "uint256",

                "name": "\_price",

                "type": "uint256"

            }

        ],

        "name": "setPrice",

        "outputs": [],

        "stateMutability": "nonpayable",

        "type": "function"

    }

];

// 0x0b23c2e1e574B301FdB2b36f7Cdeac31A8DC2893  本地

//0x7a9901a9667297b5744cC1fb3CFf075d76db95d3 rinkeby

var contract = new web3.eth.Contract(contractAbi, "0x52D64956cf1B513Df13A24947baEBe78BD458ca8");

console.log("contract", contract)

$("#Btn\_getTotalSupply").click(function () {

    console.log("Btn\_getTotalSupply");

    contract.methods.getTotalSupply().call({ from: accounts[0] }).then(

        function (result) {

            console.log(result)

            $(".txt\_getTotalSupply").html(result)

        }

    );

}

)

//

$("#Btn\_getErc721").click(function () {

    console.log("Btn\_getErc721");

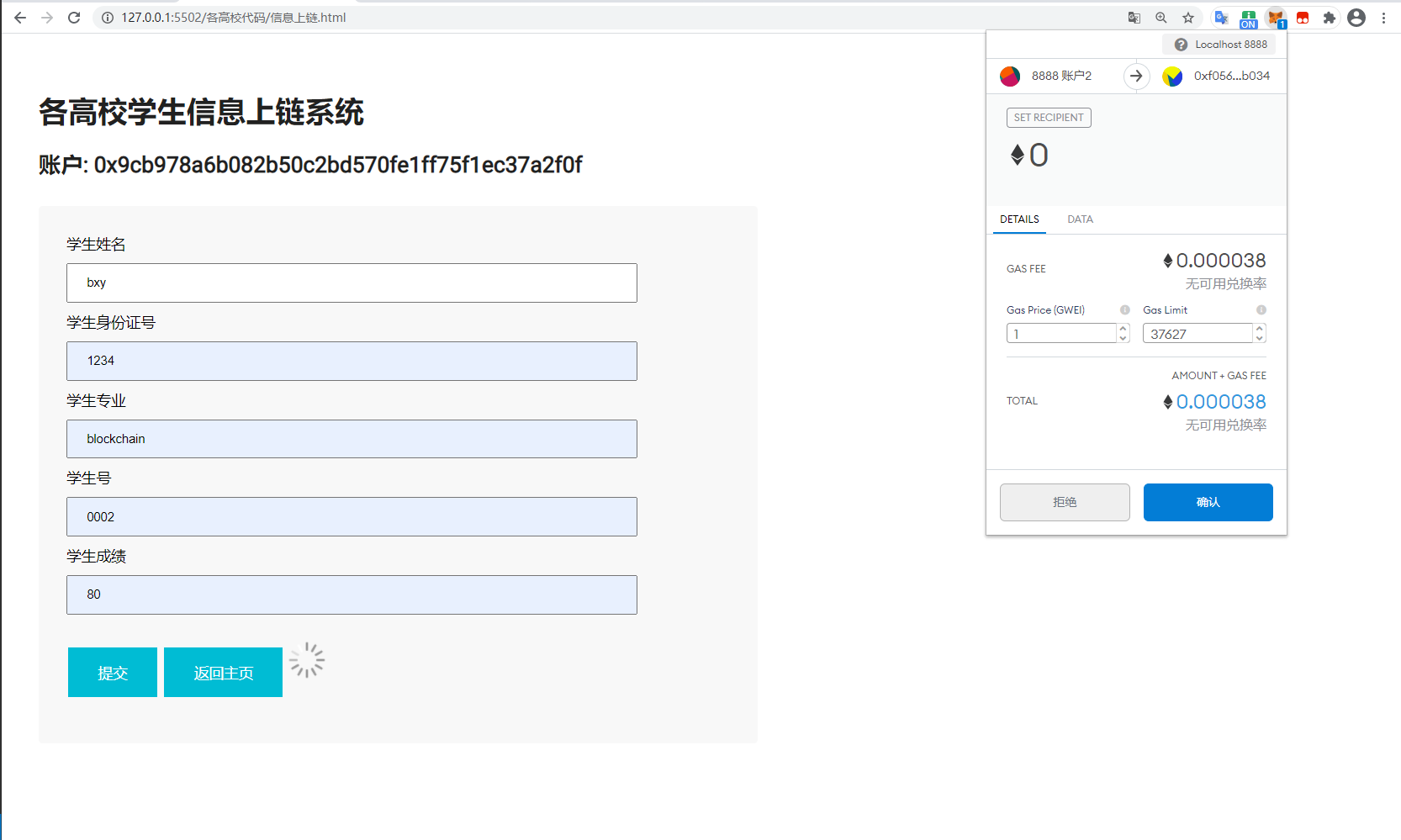
    getErc721($(".ErcAddress").html())

}

)

启动本地以太坊私链，登录MetaMask账户。打开本地Remix，连接injected Web3（MetaMask），编译部署Recipient.sol合约和certmain.sol合约。待编译部署成功后，拷贝certmain.sol合约部署地址至index.js文件84行certmain\_address处，拷贝Recipient.sol合约部署地址至index.js文件85行Recipient\_address处。

打开信息上链.html文件显示以下页面，此时各高校可以按照学生在校记录输入学生在校信息，确认无误后点击提交按钮，后台web调用certmain.sol合约连接Recipient.sol合约，实现合约交互。MetaMask弹出交易信息，点击确认按钮，等待合约交互成功。（附下图参考）



当满足MetaMask钱包账户与合约部署账户相对应、index.js文件中写入的certmain.sol合约地址和Recipient.sol合约地址无误、钱包余额足够等条件时，合约交互将成功。

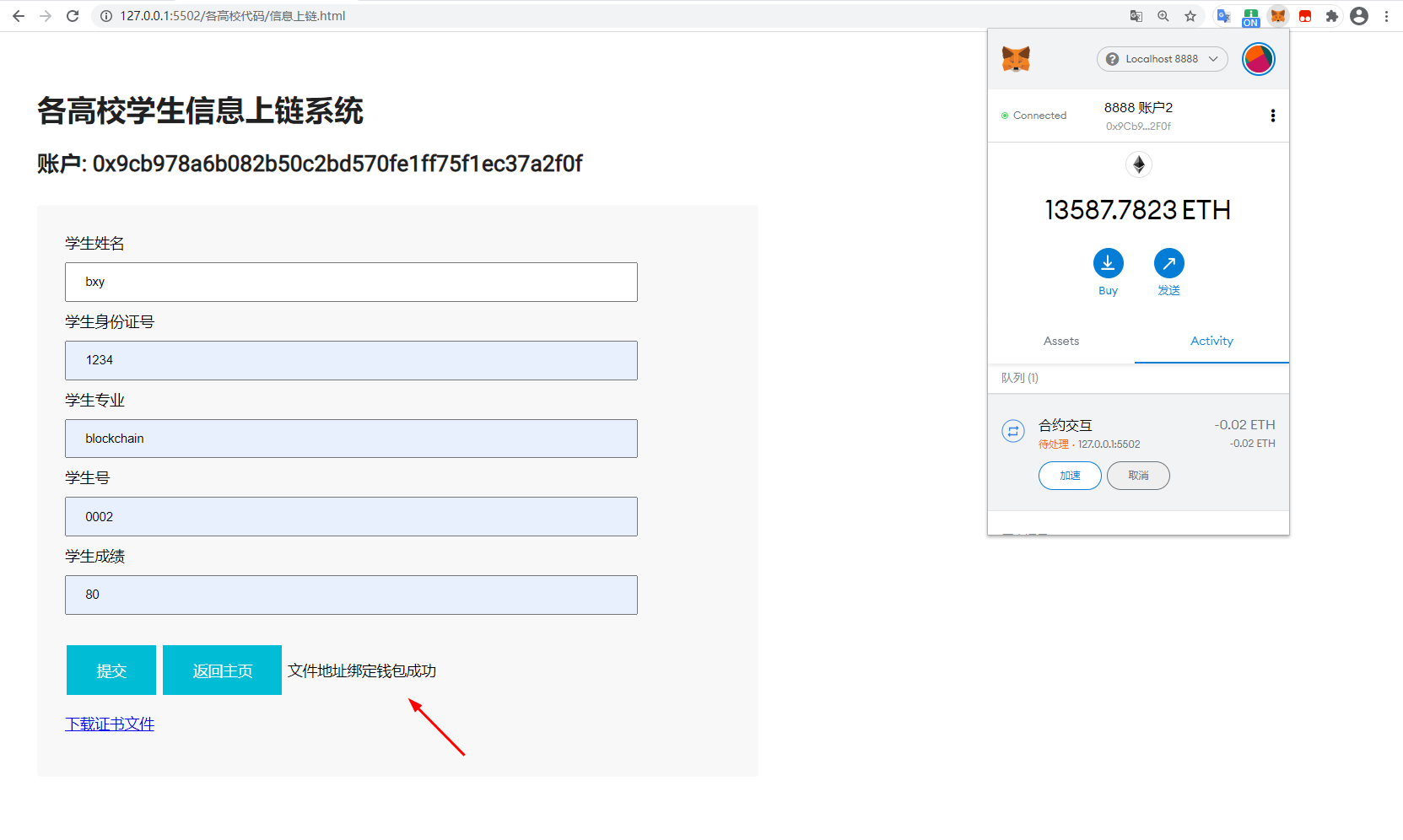
当合约交互成功后，MetaMask又将自动弹出，此时是学生信息上链过程。继续点击确认按钮，在满足上一过程条件的基础下，合约将在一定时间段后部署成功。

页面附着合约交互和合约部署进度条，在出现进度条隐藏、显示信息上链成功并且可下载证书文件的情况下，表示教育证书信息上链成功。（附下图参考）

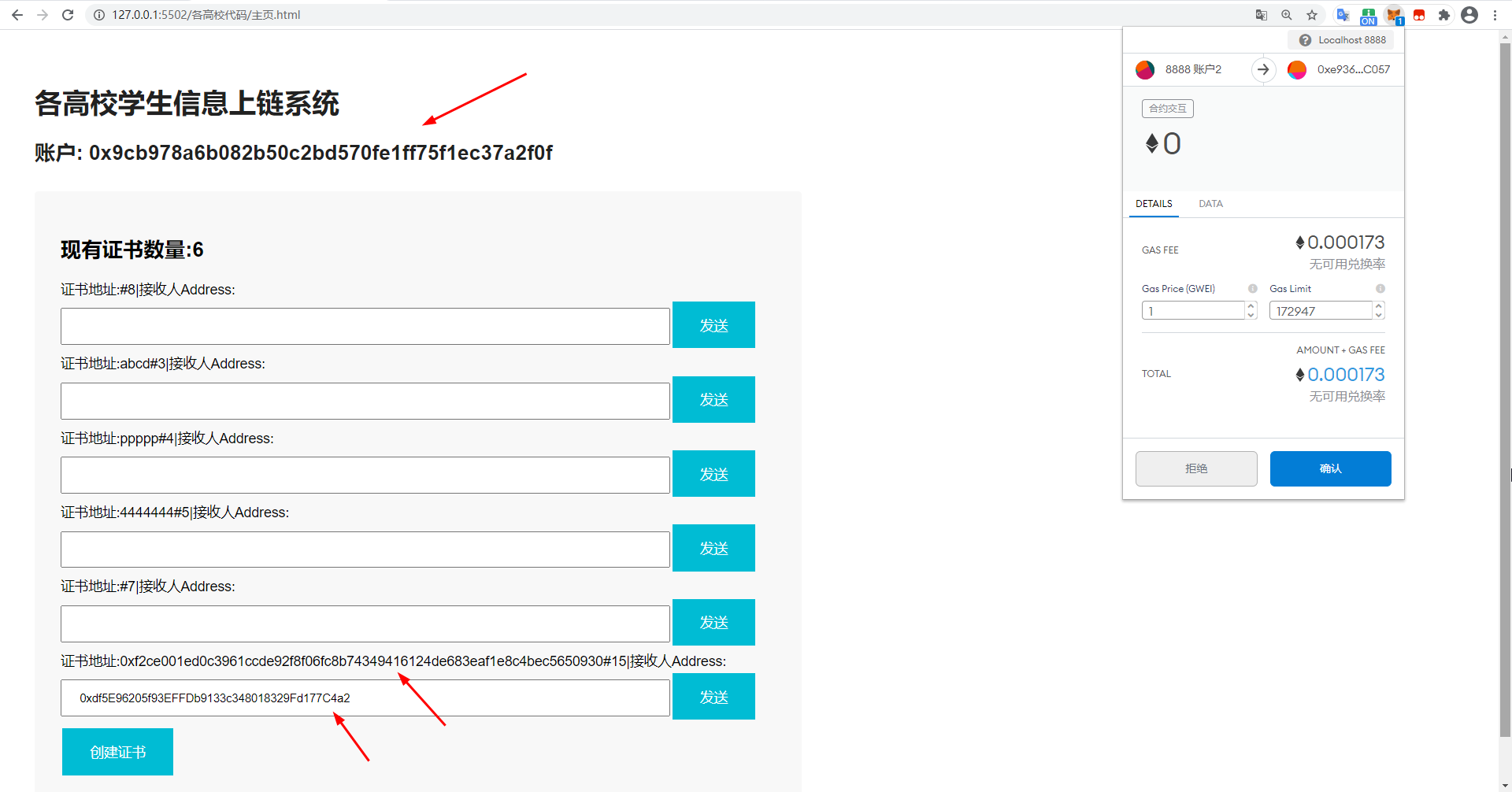


当教育证书信息上链成功后，下载证书文件。此时各高校将证书文件上传至以教育证书信息上链的交易哈希为地址的网址上（该网址较难被他人猜出，使得教育证书文件存储方便安全），并且将教育证书信息上链的交易哈希写入MetaMask账户中储存。（此过程利于将教育证书文件存储的网址安全便携地发送给教育厅）

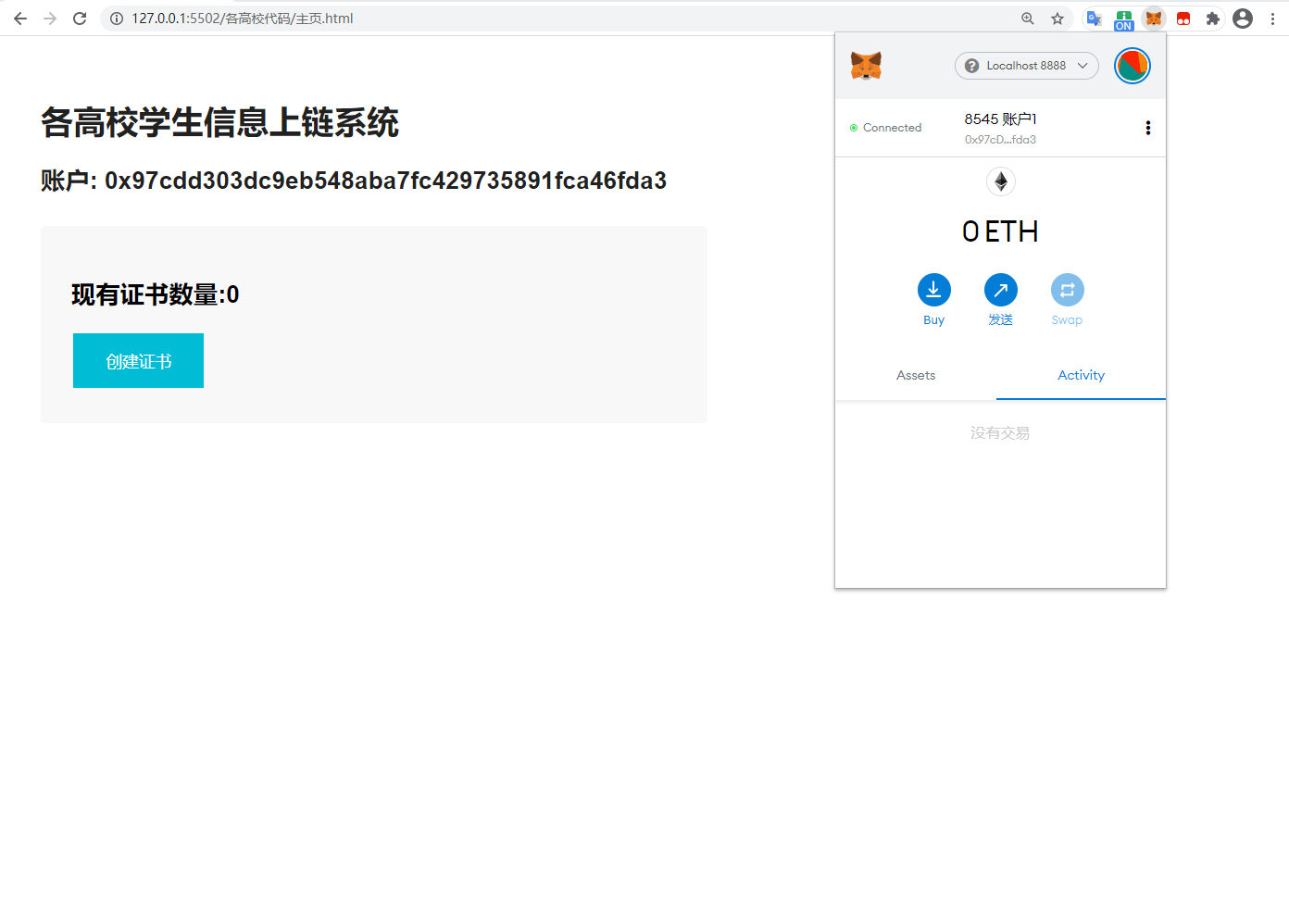
MetaMask钱包自动弹出，点击确认按钮，实现将教育证书信息上链的交易哈希写入MetaMask账户中储存过程。满足第一步条件的前提下，一段时间合约交互成功，页面“信息上链成功”提示信息变成“文件地址绑定钱包成功”提示信息，教育证书文件地址存储成功。（附下图参考）



当教育证书文件信息上链成功和教育证书文件地址存储成功后，点击返回主页按钮，页面跳转至“主页.html”文件页面，此时可以看到上过程教育证书文件存储的证书地址，实现了证书的查询功能。（该页面显示前提是登录的MetaMask账户是高校被认证的账户，也就是与上述过程合约交互和合约部署的账户地址一致，附下图参考）



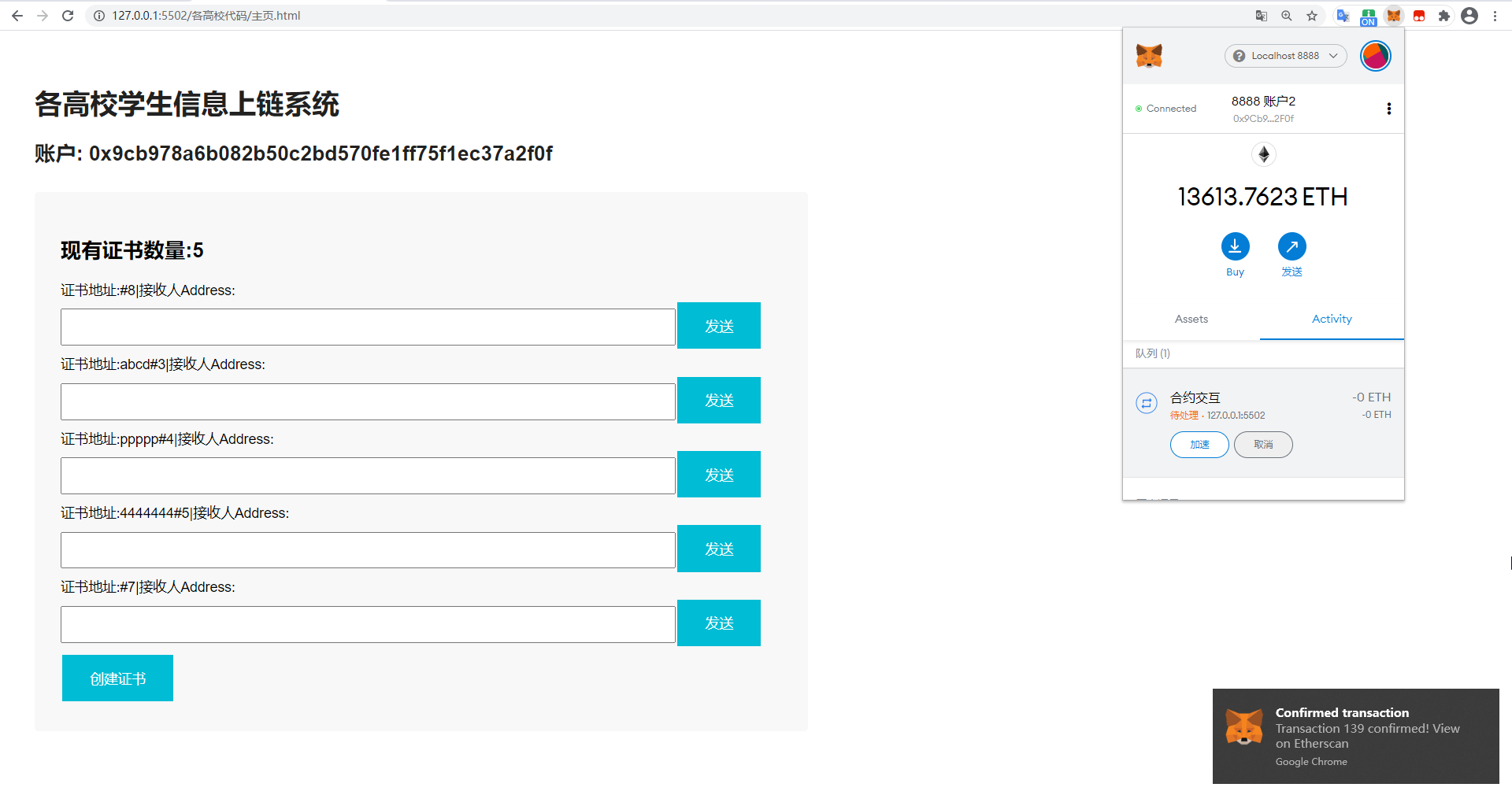
如若该页面登录的MetaMask账户不是高校被认证的账户，也就是说不是与上述过程合约交互和合约部署的账户地址一致，则无法显示证书地址，查询功能无法实现。（附下图参考）



当教育证书文件被创造成功，教育证书文件存储地址绑定钱包账户功能也成功实现后，各高校即可将教育证书文件存储地址通过MyDapp.sol合约文件发送给教育厅账户。

各高校使用MetaMask账户进入各高校学生信息上链系统页面，选择需要发送的证书地址选项栏，将教育厅的MetaMask账户地址输入文本框，点击右侧对应的发送按钮，合约开始交互。

地址发送过程一旦交互成功，高校使用MetaMask进入各高校学生信息上链系统页面时发送成功的证书地址将会消失。（附下图参考）



## 教育厅基于各高校的教育证书实现签名

## 实现教育证书文件的下载（0xfd3ac0e515d4867b6aa6c3947244d1e05953a78515c1aa28d695a8747b4cd5d0.html）

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

    <script type="text/javascript" src="web3.min.js"></script>

    <script type="text/javascript" src="jquery.js"></script>

</head>

<body>

<img id="girlImg" src="./signature/cert\_bxy.json" alt="">

<button class="downloadBtn" type="button" onclick="downloadImg()">下载文件</button>

</body>

<script>

    function downloadImg() {

        var img = document.getElementById('girlImg'); // 获取要下载的图片

        var url = img.src;                            // 获取图片地址

        var a = document.createElement('a');          // 创建一个a节点插入的document

        var event = new MouseEvent('click')           // 模拟鼠标click点击事件

        a.download = 'cert\_bxy'                  // 设置a节点的download属性值

        a.href = url;                                 // 将图片的src赋值给a节点的href

        a.dispatchEvent(event)                        // 触发鼠标点击事件

    }

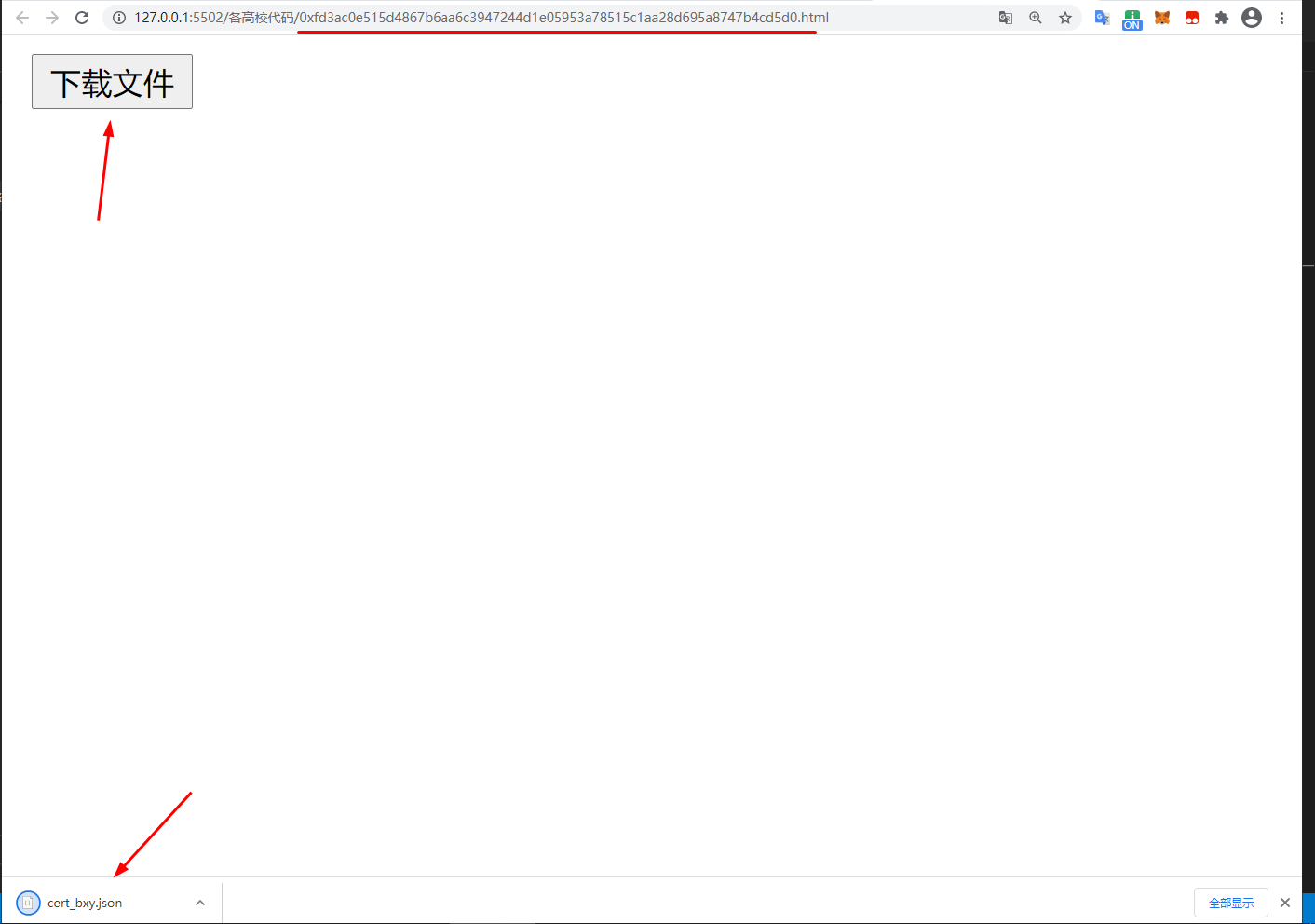
</script>

</html>

教育厅登录MetaMask账户，打开签名.html文件，进入教育厅加密签名证书文件系统即可显示接收到的证书存储地址。（附下图参考）



当教育厅打开以教育证书信息上链的交易哈希为地址的网址时，即可下载教育证书文件。（附下图参考）



## 下载至本地的教育证书文件（cert\_bxy.json）

{

    "schema": "opencerts/v2.0",

    "deploy": "0x7a17a0F3Fe7A87a873d30A4F6Fd4D07Fb28b0a5F",

    "transaction\_hash": "0xfd3ac0e515d4867b6aa6c3947244d1e05953a78515c1aa28d695a8747b4cd5d0",

    "data": {

        "recipient": {

            "name": "bxy",

            "nric": "1234",

            "course": "blockchain",

            "studentId": "0002",

            "transcript": "80"

        }

    }

}

## 教育厅加密签名区块链教育证书文件（encrypt.js）

//加密签名

const ethers = require('ethers');

const fs = require("fs");

// 账号0x97cDD303dc9EB548aba7fc429735891FcA46fda3

let privateKey = "0xaaxxx4cb7e8eb629879"; //私钥

let wallet = new ethers.Wallet(privateKey);

// console.log('wallet=>', wallet);

let accountAddress = wallet.address;

console.log('账户地址=>', accountAddress);

// fs.readFile

var data = fs.readFileSync('./cert\_bxy.json', 'utf8');

let json = JSON.parse(data);

// console.log("data",data)

console.log(json)

console.log("交易哈希:",json.transaction\_hash);

let hash = json.transaction\_hash;

let binaryData = ethers.utils.arrayify(hash);

let signPromise = wallet.signMessage(binaryData)

signPromise.then((signature) => {

    console.log('签署结果:', signature);

    let sig = ethers.utils.splitSignature(signature);

    console.log('展开:', sig);

    //插入签名

    var j =json;

    j.signature=signature

    // console.log(j)

    fs.writeFile('./signature/cert\_' + json.data.recipient.name + '.json', JSON.stringify(j), 'utf8', function (error) {

        if (error) {

            console.log(error);

            return false;

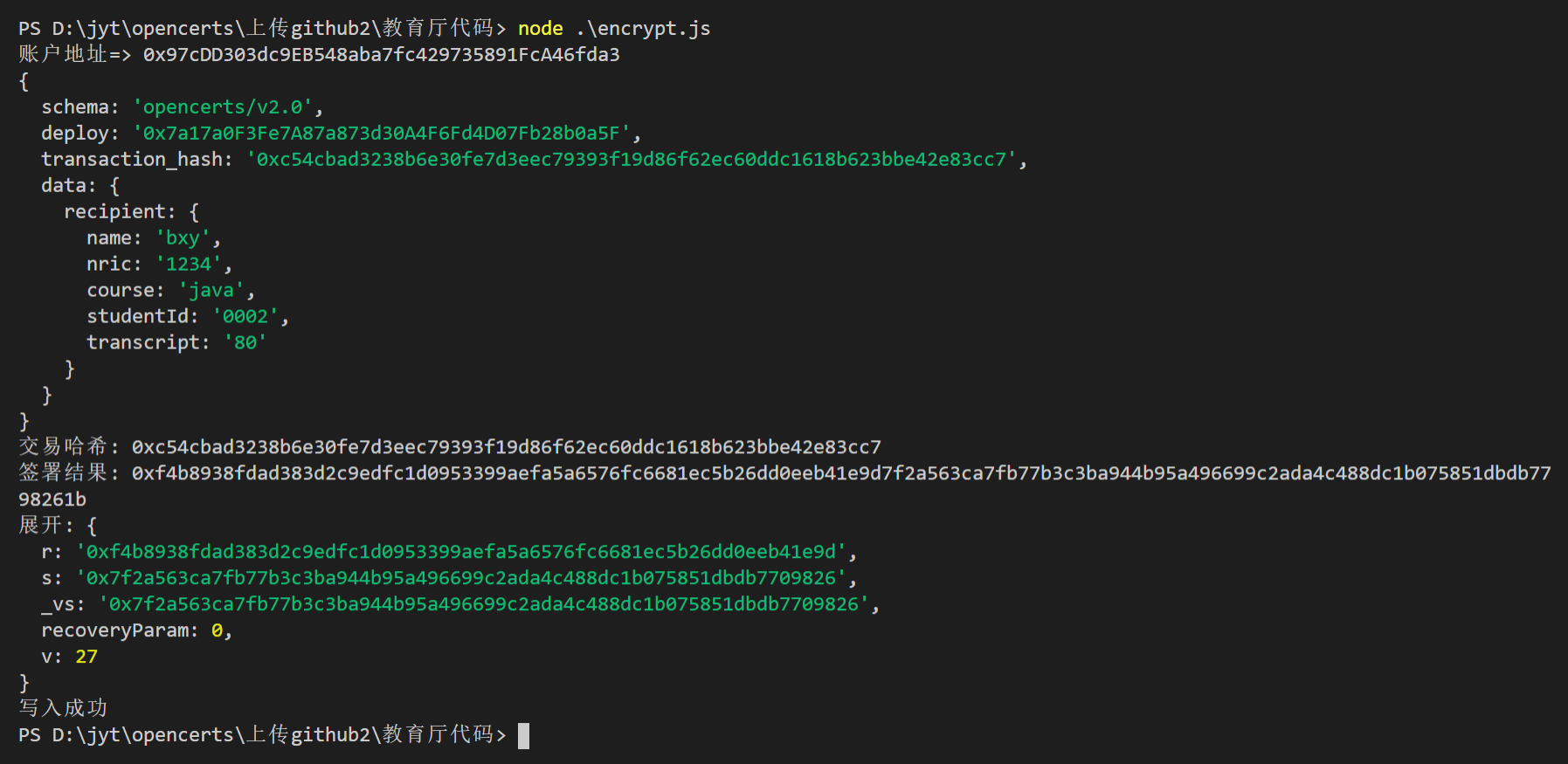
        }

        console.log('写入成功');

    })

});

教育厅得到各高校教育证书文件后，执行node encrypt.js命令。encrypt.js利用ethers和教育厅账户私钥对教育证书文件进行加密签名，并利用fs将签名后的教育证书文件自动写入signature文件下，该文件根据教育证书所有者姓名命名，例如cert\_bxy.json。（运行结果附下图参考）



## 签名后的教育证书文件（cert\_bxy.json）

{

    "schema": "opencerts/v2.0",

    "deploy": "0x7a17a0F3Fe7A87a873d30A4F6Fd4D07Fb28b0a5F",

    "transaction\_hash": "0xc54cbad3238b6e30fe7d3eec79393f19d86f62ec60ddc1618b623bbe42e83cc7",

    "data": {

        "recipient": {

            "name": "bxy",

            "nric": "1234",

            "course": "blockchain",

            "studentId": "0002",

            "transcript": "80"

        }

    },

    "signature": "0xf4b8938fdad383d2c9edfc5a6576fc6681ec5b26dd0eeb41e9d7f2a563ca7fb77b3c3ba944b95a496699c2ada4c488dc1b075851dbdb77098261b"

}

当教育厅对教育证书文件签名完成后，将签名后的教育证书文件上传至以教育证书信息上链的交易哈希为地址的网址上（与上述各高校使用MetaMask发送给教育厅的教育证书文件存储地址一致，只需将各高校的域名改成教育厅的域名）。随后教育厅便可将签名完成后的教育证书文件存储地址通过MetaMask发送给证书接收者。（附下图参考）



## 学生基于学生教育证书发布查询系统查询证书是否发布

## 教育证书的证书查询代码（证书查询.html，index.js部分方法）

证书查询.html

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="utf-8">

    <script type="text/javascript" src="jquery.js"></script>

    <script type="text/javascript" src="web3.min.js"></script>

    <link rel="stylesheet" href="style.css" />

    <meta name="viewport"

        content="width=device-width,initial-scale=1,minimum-scale=1,maximum-scale=1,user-scalable=no" />

    <title>区块链学院 Dapp

    </title>

</head>

<body>

    <h1>学生教育证书发布查询系统</h1>

    <button class="enableEthereumButton" hidden>

        <!-- 连接以太坊钱包 -->

    </button>

    <h2>账户: <span class="showAccount"></span></h2>

    <div id="ftc-make">

        <div class=form-break></div>

        <button id="Btn\_getErc721" hidden>

        </button>

        <span class="ErcAddress" hidden></span>

        <div class="form-group">

            <h2>您的教育证书是否发布:<span id="MyBalance"></span></h2>

            <div id="box"></div>

            <script type="text/javascript" src="index.js"></script>

</body>

</html>

index.js部分方法

$(".enableEthereumButton").click(function () {

    getAccount()

}

)

async function getAccount() {

    accounts = await ethereum.request({ method: 'eth\_requestAccounts' });

    const account = accounts[0];

    $(".showAccount").html(account);

}

ethereum.on('accountsChanged', function (accounts) {

    console.log("accountsChanged");

    getAccount()

});

ethereum.on('chainChanged', (chainId) => {

    console.log("chainId ", chainId);

});

let web3 = new Web3(Web3.givenProvider || "ws://localhost:8546");

console.log("web3", web3)

async function getProdID(\_index) {

    contract.methods.getProdId(\_index).call({ from: accounts[0] }).then(

        function (result) {

            console.log("getProdID", result)

            $("#pitem" + \_index).html(result.r\_token);

            $("#tokenIndex" + \_index).html(result.r\_index);

        }

    );

}

async function getAccount() {

    accounts = await ethereum.request({ method: 'eth\_requestAccounts' });

    const account = accounts[0];

    // showAccount.innerHTML = account;

    $(".showAccount").html(account);

    getBalance();

    ErcAddress();

}

async function ErcAddress() {

    contract.methods.ErcAddress().call({ from: accounts[0] }).then(

        function (result) {

            console.log("ErcAddress==》", result)

            $(".ErcAddress").html(result)

        }

    )

}

function getBalance() {

    contract.methods.getUserBalance(accounts[0]).call({ from: accounts[0] }).then(

        function (result) {

            $('#box').children().remove();

            console.log('getUserBalance', result)

            $("#MyBalance").html(result)

            for (var i = 0; i < result; i++) {

                console.log(i + "========================")

                // var el = $('<h4>证书地址:<span id="pitem' + i + '">-- </span>#<span id="tokenIndex' + i + '"></span>' + '<br>现将证书授权于Address:<input id="s\_address' + i + '" value="" /><button id="Btn\_send' + i + '"  onClick="sendBtn(' + i + ')">发送</button></h4>');

                var el = $('<h3>证书地址:<span id="pitem' + i + '">-- </span><br><br><img src="./logoBig.png" /></h3>');

                getProdID(i);

                $('#box').append(el);

            }

        }

    );

}

ethereum.on('accountsChanged', function (accounts) {

    console.log("accountsChanged");

    getAccount()

});

ethereum.on('chainChanged', (chainId) => {

    console.log("chainId", chainId);

});

$("#Btn\_getTotalSupply").click(function () {

    console.log("Btn\_getTotalSupply");

    contract.methods.getTotalSupply().call({ from: accounts[0] }).then(

        function (result) {

            console.log(result)

            $(".txt\_getTotalSupply").html(result)

        }

    );

}

)

$("#Btn\_getErc721").click(function () {

    console.log("Btn\_getErc721");

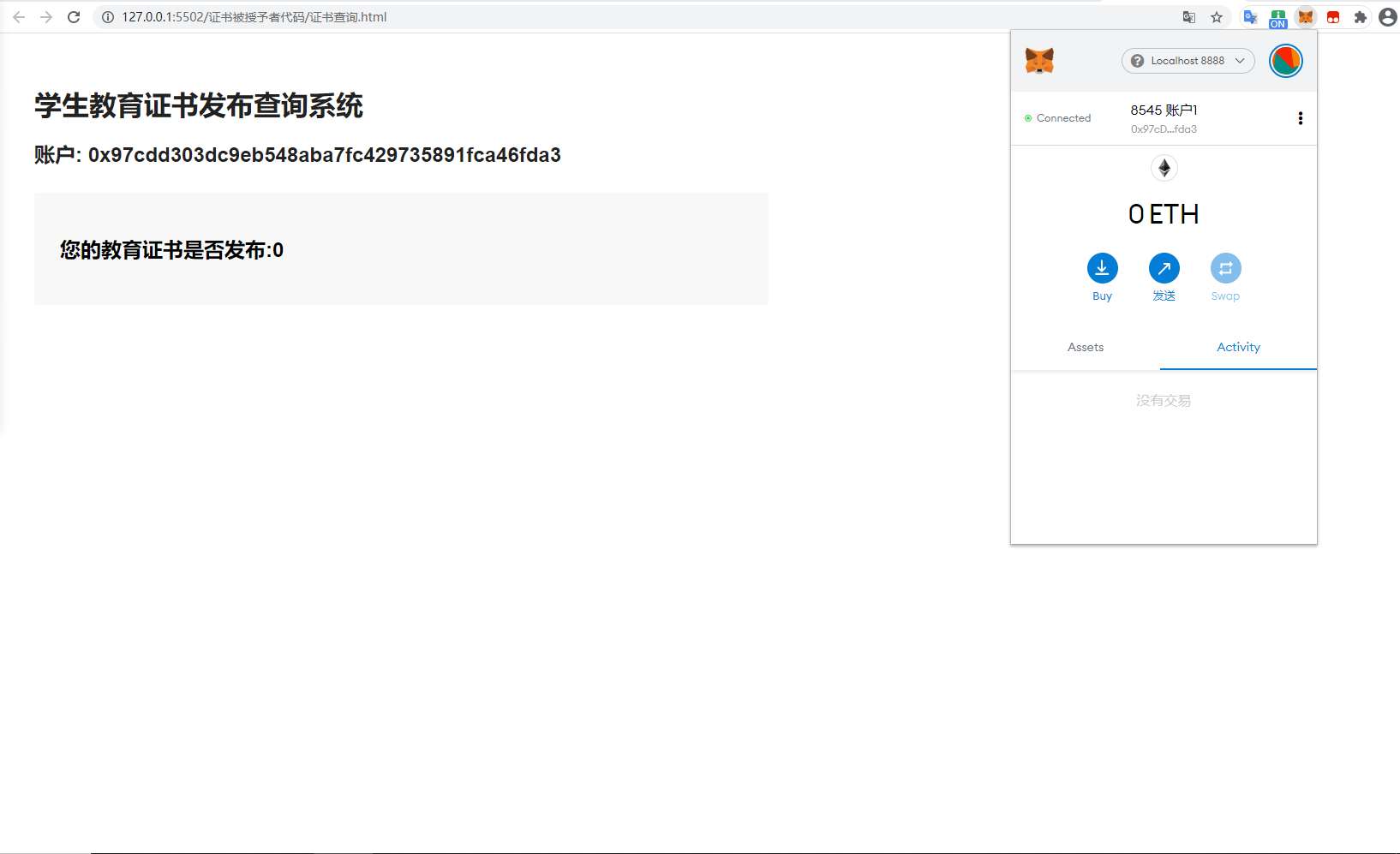
    getErc721($(".ErcAddress").html())

}

)

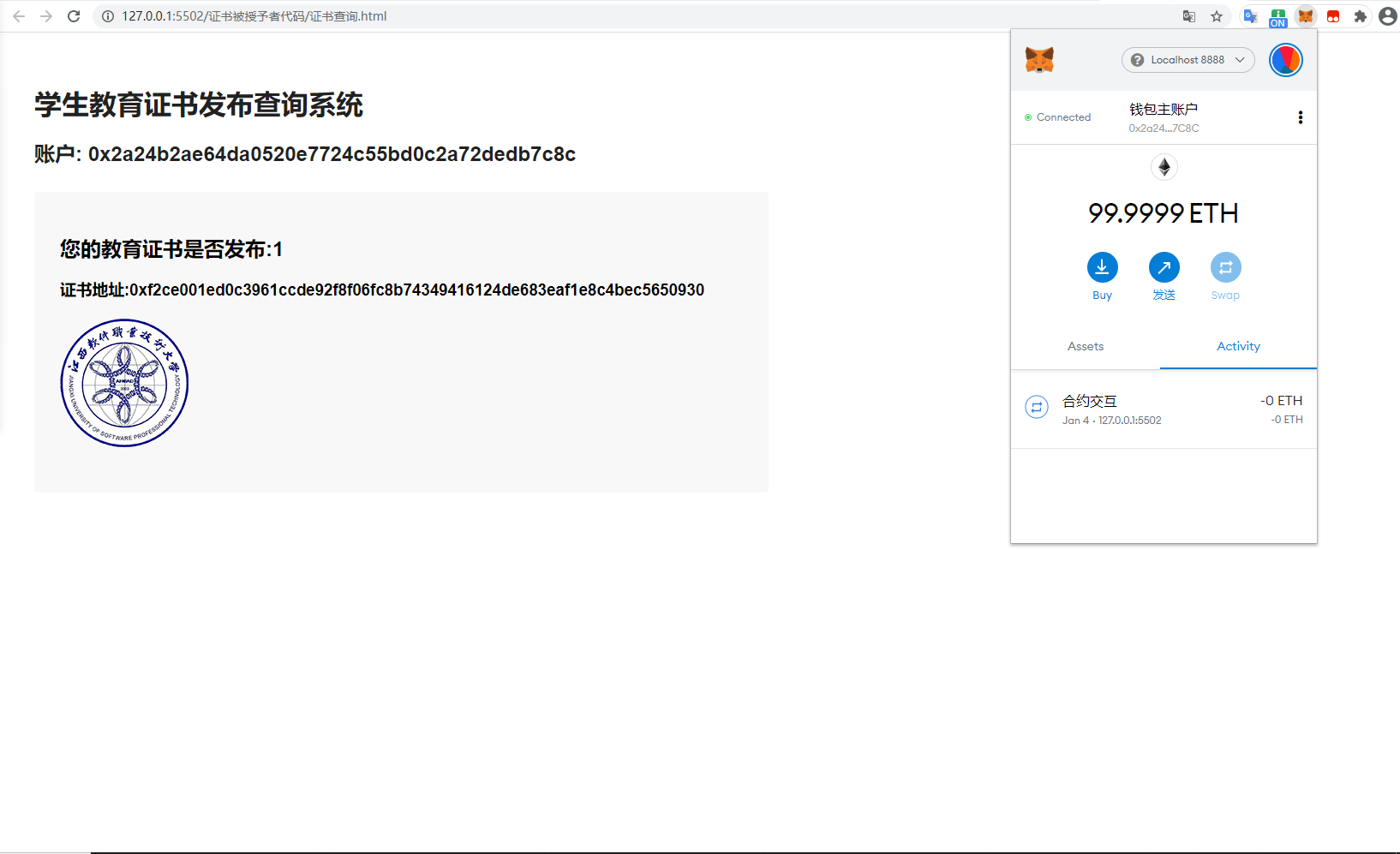
学生登录MetaMask账户，打开证书查询.html文件，进入学生教育证书发布查询系统即可显示查询教育证书是否发布。

如若该学生区块链教育证书尚未发布，则页面显示“您的教育证书是否发布:0”。（附下图参考）



如若该学生区块链教育证书已发布，则页面显示“您的教育证书是否发布:1”，并且附上教育证书文件存储地址（附下图参考）。

当学生打开以教育证书信息上链的交易哈希为地址的网址时，即可下载教育证书文件（与上述教育厅下载教育证书文件步骤相同）。



## 验证机构验证教育证书是否有效

## web调用Recipient.sol实现信息校验（upload.html）

<!-- https://www.cnblogs.com/gamedaybyday/archive/2004/01/13/9906542.html -->

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

    <script type="text/javascript" src="./jquery.min.js"></script>

    <h1>区块链教育证书校验系统</h1>

    <div>

        <input type="file" id="files" name="files" />

    </div>

    <div>

        <h3 style="text-align:center" id="test">

            </h5>

    </div>

    <script src="https://cdn.ethers.io/scripts/ethers-v4.min.js" charset="utf-8" type="text/javascript">

    </script>

    <script src="./web3.js"></script>

    <script>

        let url\_geth = 'http://127.0.0.1:8545';

        let web3 = new Web3(url\_geth);

        console.log("web3版本：", web3.version);

        //查询

        const Recipient\_abi =

            [

                {

                    "inputs": [

                        {

                            "internalType": "string",

                            "name": "\_name",

                            "type": "string"

                        },

                        {

                            "internalType": "string",

                            "name": "\_nric",

                            "type": "string"

                        },

                        {

                            "internalType": "string",

                            "name": "\_course",

                            "type": "string"

                        },

                        {

                            "internalType": "string",

                            "name": "\_studentId",

                            "type": "string"

                        },

                        {

                            "internalType": "string",

                            "name": "\_transcript",

                            "type": "string"

                        }

                    ],

                    "name": "addJieShouRen",

                    "outputs": [],

                    "stateMutability": "nonpayable",

                    "type": "function"

                },

                {

                    "inputs": [

                        {

                            "internalType": "uint256",

                            "name": "",

                            "type": "uint256"

                        }

                    ],

                    "name": "jsrlist",

                    "outputs": [

                        {

                            "internalType": "string",

                            "name": "name",

                            "type": "string"

                        },

                        {

                            "internalType": "string",

                            "name": "nric",

                            "type": "string"

                        },

                        {

                            "internalType": "string",

                            "name": "course",

                            "type": "string"

                        },

                        {

                            "internalType": "string",

                            "name": "studentId",

                            "type": "string"

                        },

                        {

                            "internalType": "string",

                            "name": "transcript",

                            "type": "string"

                        }

                    ],

                    "stateMutability": "view",

                    "type": "function"

                }

            ];

        var inputElement = document.getElementById("files");

        inputElement.addEventListener("change", handleFiles, false);

        function handleFiles() {

            var selectedFile = document.getElementById("files").files[0];//获取读取的File对象

            var name = selectedFile.name;//读取选中文件的文件名

            var size = selectedFile.size;//读取选中文件的大小

            // console.log("文件名:" + name + "大小：" + size);

            var reader = new FileReader();//这里是核心！！！读取操作就是由它完成的。

            reader.readAsText(selectedFile);//读取文件的内容

            reader.onload = function (data) {

                console.log("读取结果：", this.result);//当读取完成之后会回调这个函数，然后此时文件的内容存储到了result中。直接操作即可。

                let json = JSON.parse(this.result);

                if (web3.utils.isAddress(json.deploy)) {

                    const Recipient\_address = json.deploy;

                    const Recipient\_contract = new web3.eth.Contract(Recipient\_abi, Recipient\_address);

                    Recipient\_contract.methods.jsrlist(0).call((error, result1) => {

                        if (!error) {

                            console.log('name:', result1.name)

                            console.log('nric:', result1.nric)

                            console.log('course:', result1.course)

                            console.log('studentId:', result1.studentId)

                            console.log('transcript:', result1.transcript)

                            var str = JSON.stringify(result1);

                            var index = str.lastIndexOf('"name"');

                            var finalValue = str.substring(index + 1);

                            const aa = JSON.stringify(json.data.recipient);

                            const bb = '{"' + finalValue;

                            if (aa == bb) {

                                //解密验证

                                // https://learnblockchain.cn/docs/ethers.js/api-wallet.html

                                let accountAddress = "0x97cDD303dc9EB548aba7fc429735891FcA46fda3";

                                console.log('账户地址:', accountAddress);

                                console.log("交易哈希:", json.transaction\_hash);

                                console.log("合约签名:", json.signature);

                                let hash = json.transaction\_hash;

                                let binaryData = ethers.utils.arrayify(hash);

                                let signature = json.signature;

                                try {

                                let signingAddress = ethers.utils.verifyMessage(binaryData, signature);

                                console.log('签名人：', signingAddress);

                                if (accountAddress === signingAddress) {

                                    console.log('验证成功，是教育厅签署后的教育证书！！！')

                                    $('#test').append('<p>' +

                                        '<h2>' + "证书被授予者" + '</h2>' + '</p>' +

                                        "学生姓名：" + json.data.recipient.name + '</p>' +

                                        "学生身份证号：" + json.data.recipient.nric + '</p>' +

                                        "学生专业：" + json.data.recipient.course + '</p>' +

                                        "学生号：" + json.data.recipient.studentId + '</p>' +

                                        "学生成绩：" + json.data.recipient.transcript + '</p>' +

                                        "合约部署地址：" + json.deploy + '</p>' +

                                        "合约交易哈希：" + json.transaction\_hash + '</p>'

                                        // "合约签名：" + json.signature + '</p>'

                                        + '</p>');

                                }

                                else {

                                    $('#test').append('验证失败，非公认机构发布证书！！！')

                                }} catch (err) {

                                    $('#test').append('验证失败，证书签名无效！！！')

                                }

                            } else {

                                $('#test').append("教育证书已被篡改，无法读取证书信息！！！")

                            }

                        }

                    });

                } else {

                    $('#test').append("该证书为无效证书，请重新上传教育证书文件！！！")

                }

            };

        };

    </script>

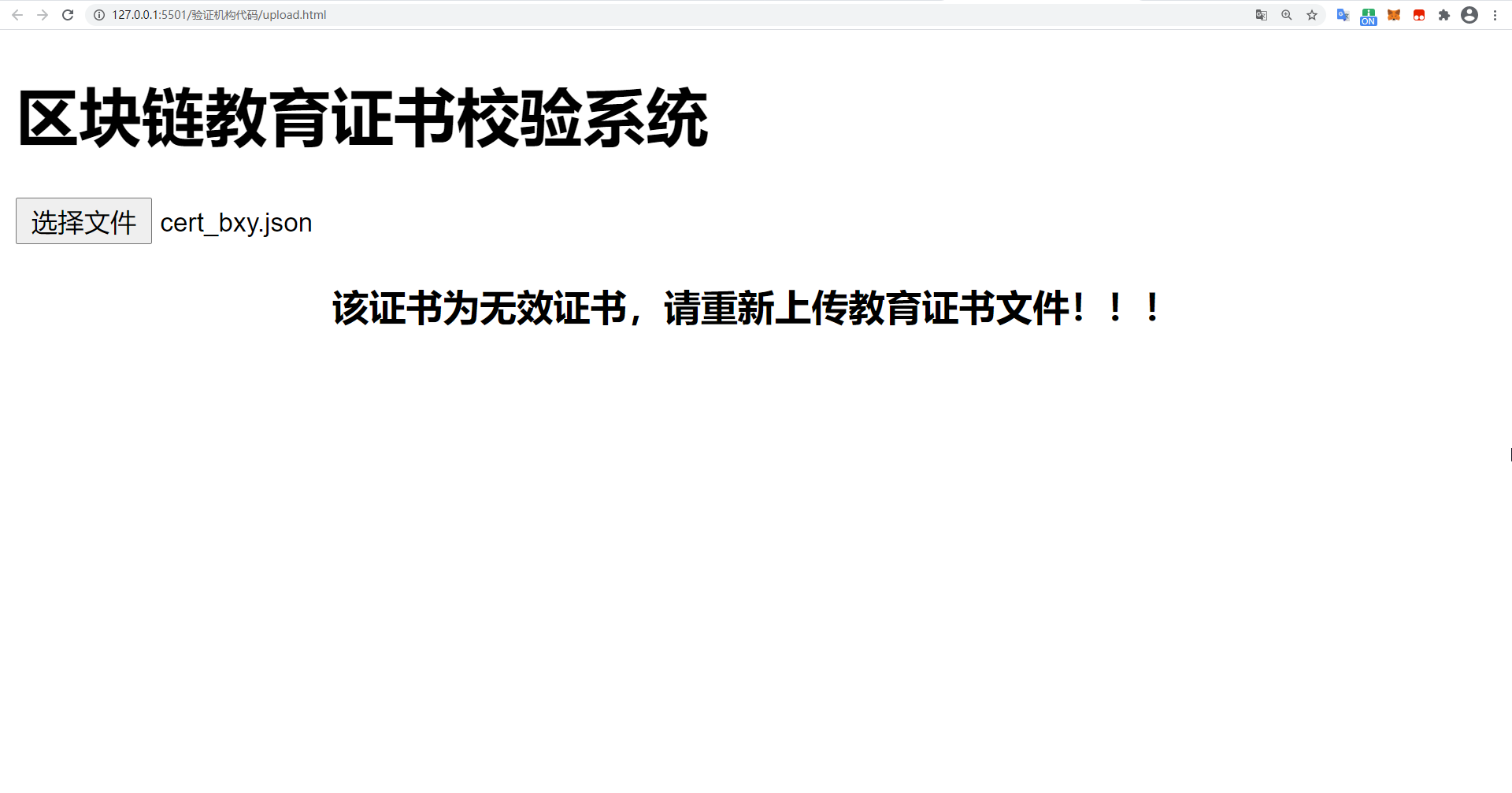
</head>

</html>

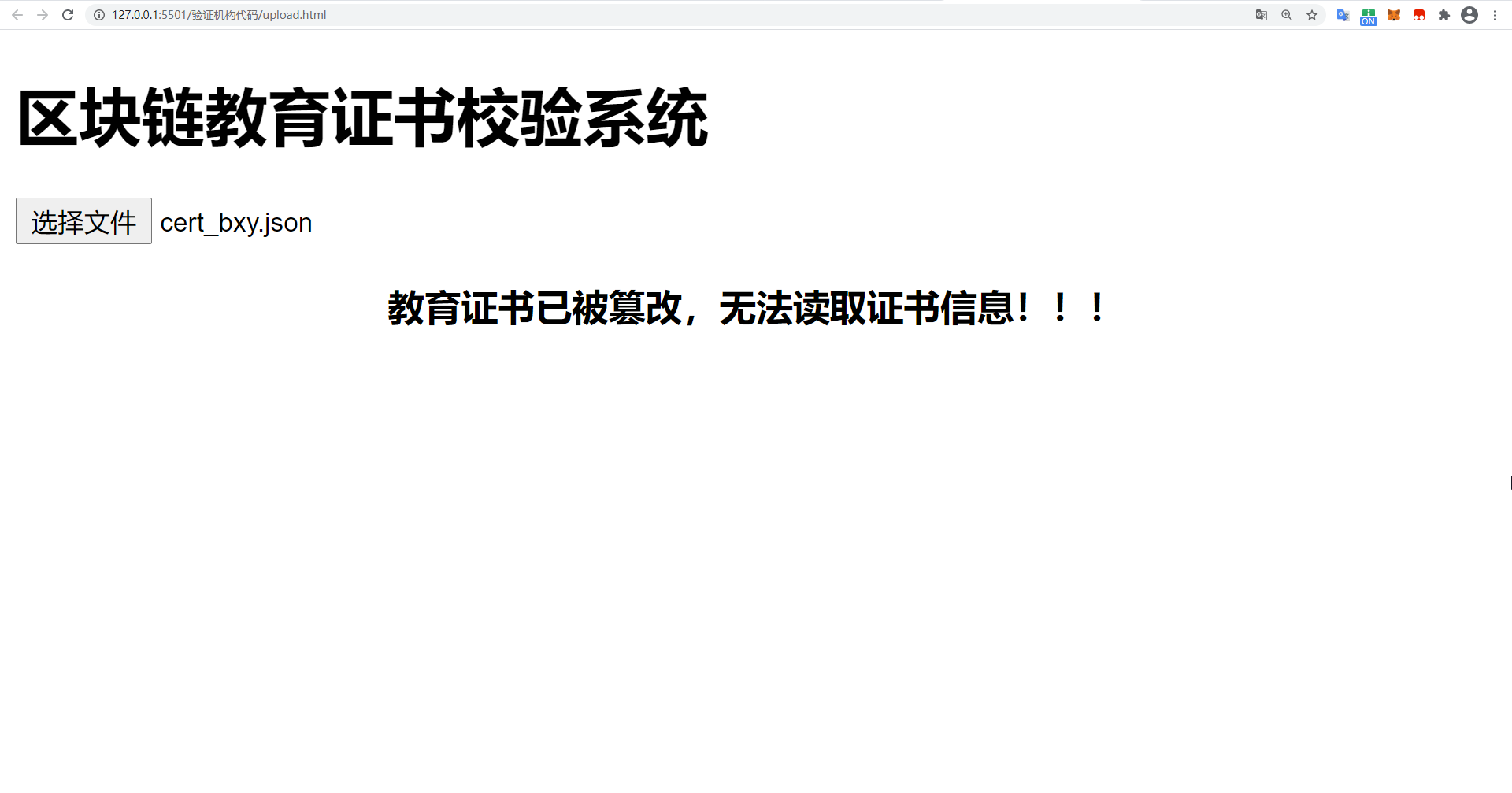
启动本地以太坊私链，点击目录下upload.html文件出现以下页面。点击选择文件按钮，选择需要验证的教育证书文件。

一、利用reader.readAsText(selectedFile)读取选择上传的教育证书文件中所有内容

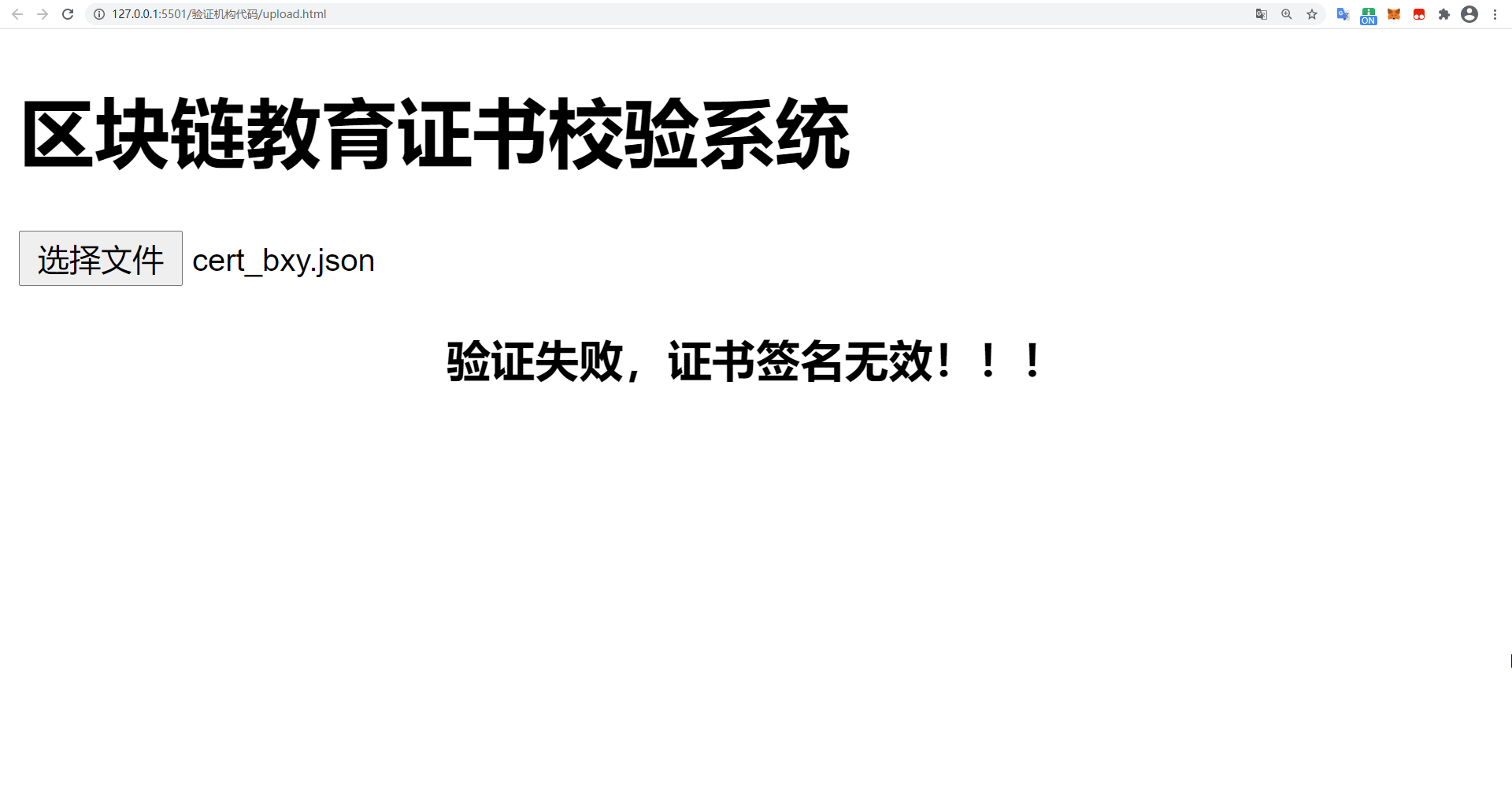
二、利用web3.utils.isAddress(deploy)方法验证上传教育证书文件中deploy（合约部署地址）是否合法。如若地址不合法，教育证书可能是被伪造或者已被篡改，系统将无法通过校验，页面将会返回“该证书为无效证书，请重新上传教育证书文件！！！”的错误提示（附下图参考）。如若合约部署地址合法，将进入下一步检验。



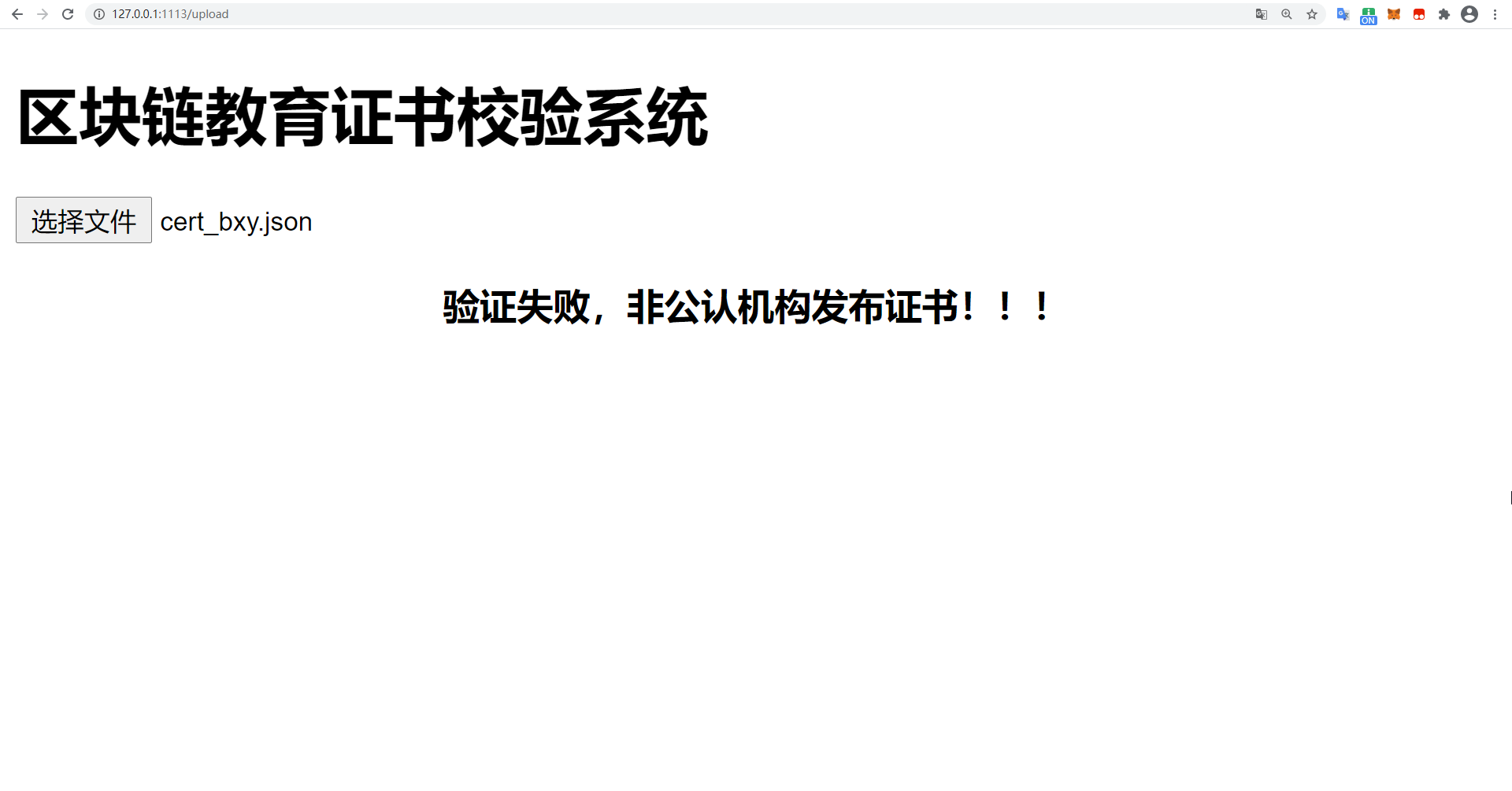
三、利用Recipient.sol合约文件的abi文件、上传教育证书读取的合约部署地址、合约的call方法等信息在区块链上查询上链记录并与上传的教育证书文件内容逐一进行对比。当上传证书文件内信息与区块链记录信息存在信息不符合情况，教育证书可能是被伪造或者已被篡改，系统将无法通过校验，页面将会返回“教育证书已被篡改，无法读取证书信息！！！”的错误提示（附下图参考）。如若上传证书文件内信息与区块链记录信息完全符合，将进入下一步检验。



四、利用ethers.utils.verifyMessage(binaryData, signature)方法将上传教育证书文件中的signature（教育厅签名）进行签名账户比对。如若上传教育证书文件内的签名为无效签名，异常将被捕获，教育证书可能是被伪造或者已被篡改，系统将无法通过校验，页面将会返回“验证失败，证书签名无效！！！”的错误提示（附下图1参考）。如若上传教育证书文件内的签名为有效签名，验证系统将会利用if (accountAddress === signingAddress)检验上传教育证书文件内签名的账户与教育厅公认账户是否相同。如若上传教育证书文件内签名的账户与教育厅公认账户不相同，教育证书可能是被伪造或者已被篡改，系统将无法通过校验，页面将会返回“验证失败，非公认机构发布证书！！！”的错误提示（附下图2参考）。如若上传教育证书文件内签名的账户与教育厅公认账户相同，则上传教育证书文件检验完毕，该上传证书文件信息将被成功打印于页面上（附下图3参考）。



如图1



如图2

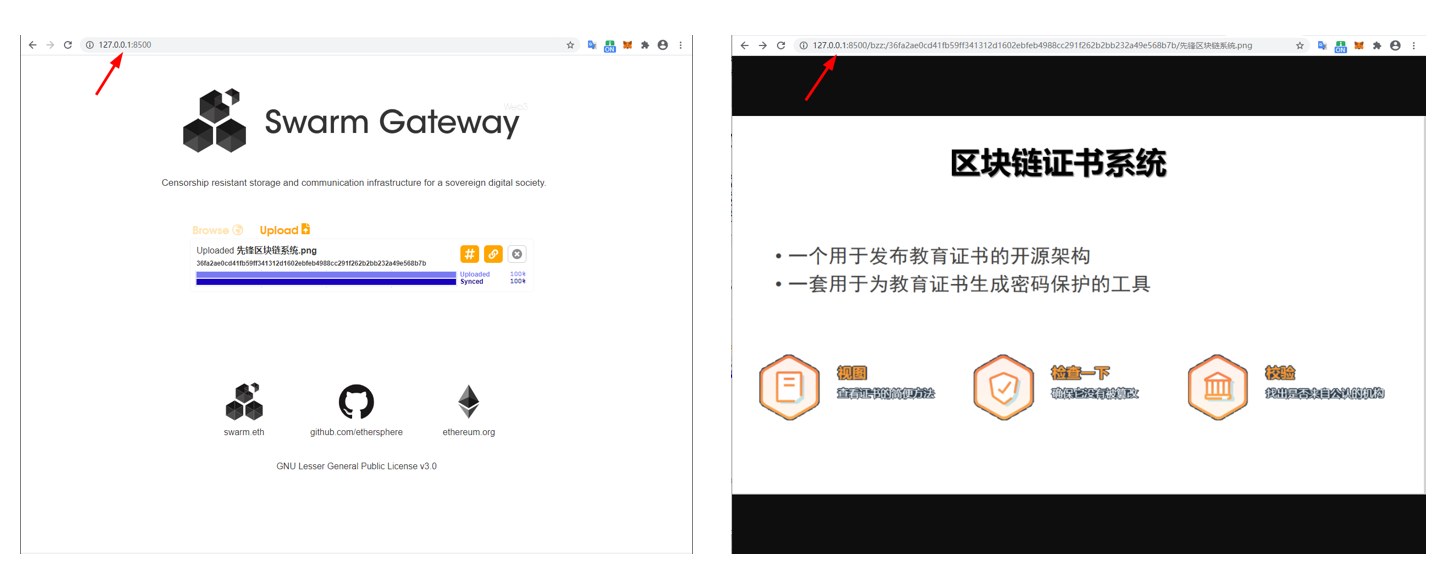


如图3

# 五、基于区块链教育证书系统的有关问题：

## 1、教育证书的个人数据在区块链上安全吗？

教育证书和个人数据的学术记录不能在区块链上被他人查询阅览。教育证书生成的哈希，用于证明该交易证书是合法的。教育证书计算哈希的过程是不可逆的。由于哈希是发布到区块链中的唯一信息，而教育证书是需要通过区块链钱包账户登录教育证书查询系统才有权查询证书信息，因此他人无法在教育证书持有人不知或者不允许的情况下从区块链上的内容获取任何个人信息。如若个人想查询他人的教育证书信息，需要经过教育证书持有人允许并且获得他人的教育证书文件，同时每次查询均需要支付给教育证书所有人一笔相应费用，并且教育证书所有人能够查询个人证书的一切查询信息记录。



## 2、为什么区块链教育证书系统要使用区块链技术？

与目前现有的昂贵的专有软件相比，使用区块链，我们可以大大减少发布受密码保护的教育证书的门槛。此外，公共区块链由社区拥有和维护，任何人都可以轻松访问。因此，无需运行或维护服务来验证区块链教育证书系统。

## 3、教育证书能否打印后使用？­­­­

不能。打印教育证书会丢弃系统内置在区块链教育证书中的所有高级密码保护，无法通过教育证书验证系统的验证。因此打印的教育证书不被视为真实证书。

## 4、个人能否查询他人的教育证书？

可以，但需要经过他人同意并且每次查询将要支付一笔费用给证书所有者。

要想查询他人的教育证书，首先需要和他人协商，经过他人同意后通过邮件或者微信等传输工具获取到他人的教育证书文件。由于教育证书验证系统需要区块链钱包账户登录，当登录的区块链钱包账户与拖拉上传的教育证书文件所关联的区块链钱包账户不一致时，系统将会要求登录的区块链钱包账户支付一定的数字货币，当支付货币成功后即可查询他人的教育证书。此时证书所有者账户将会收到查询者所支付的数字货币，并且该账户页面将会显示查询方的基本信息和查询证书的时间戳和次数。

## 5、教育证书文件使用安全吗？他人不能复制我的教育证书文件并像我验证教育证书文件一样假冒验证吗？

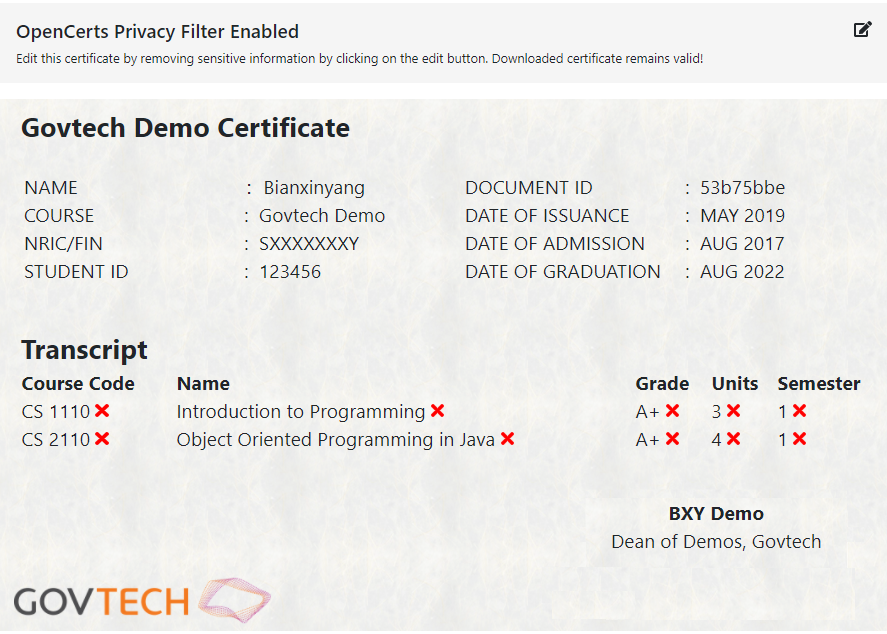
教育证书文件使用是安全的。教育证书文件可以轻松复制，但是，如果不通过教育证书验证系统的验证过程，则无法更改教育证书中的收件人姓名。因此，进行验证的人员确保教育证书中指示的接收者实际上是出示教育证书的本人，这一点极为重要。对于更高级的机构，将来可能会使用分布式ID来关联教育证书接收者的公钥，以便将来使用公钥/私钥签名进行进一步的身份验证。

## 6、教育证书可以被吊销吗？

可以。当教育证书持有者违背了基于教育证书系统的智能合约有关协议时，合约将自动实行程序设定，发布该教育证书的吊销通知，此时该教育证书将不再是有效的教育证书。

## 7、教育证书如何实现个性化设计？

使用区块链钱包账户登入教育证书验证系统后，将教育证书文件拖拉上传至系统检验框，待校验完毕后，进入教育证书成绩单一栏（附下图）。教育证书验证系统支持隐私过滤功能，通过单击编辑按钮按个人需求删除敏感信息来编辑该教育证书。编辑完成后下载的证书仍然有效！



# 六、总结和体会

在设计基于区块链的教育证书系统过程中，我碰到了许多问题。刚开始我连基本的架构流程都理不清楚，一遍一遍地规划系统流程。流程理清楚后就是合约编译部署、合约调合约、nodejs合约交互等问题。在这个过程中，代码小到页面设计，进度条显示等问题，然后利用MetaMask账户部署交易。刚开始我利用的是ejs文件类型显示页面，但后来便于使用者使用，还是改成了html，于是js文件的方法又被我大幅度修改。读取文件内容和写入教育证书信息我开始使用的是fs，后面也改成document.getElementById("files")方法。我个人认为理清楚系统流程、合约交互、签名校验这几个过程是比较难的，我花费大量时间去寻找资料和代码不断改善。

学习软件程文档写作的课本课程只能够让我得到浅显的认识，相比之下，这次软件文档写作就更加全面的说明了开发一个项目所需要的步骤以及开发项目过程中所需要注意的诸多细节。如果说学习软件工程文档写作教科书注重理论基础的话，那么这此软件工程文档写作考试，更加侧重的是我的动手操作的能力。原来我认为开发一个项目最重要的就是写代码，似乎整个软件都是编代码，因为自己动手能力不强所以就很排斥做项目。可是经过学习软件工程文档理论到尝试做项目再到这次软件文档写作考试之后，我才真正意识到实施一个软件工程项目并不是说简单的会编码就能够解决问题的,因为一个软件的生命周期分为三个时期:软件定义时期、开发时期、维护时期，而这三个时期整体又分为七个阶段，他们分别是：问题定义、可行性研究、需求分析、总体设计、详细设计、编码和单元测试、综合测试，由此可看出，当我们开发一个项目时，更多的精力不是放在编码上，编码只是一个很小的模块，而是项目的整体结构上。所以我要写的软件工程文档写作的体会是：首先要先确定项目主题，也就是这个项目用来做什么，可以解决什么问题。接着就是这个项目是否有研究的必要以及是否有解决的办法。第三步根据我们确定的项目主题进行需求分析,这一步骤当时做的不是很好，比如说画数据流图等都有考虑不周的问题，导致接下来的概要设计、详细设计进行的很困难，有些步骤甚至还需要返工。从我在需求分析中出现的问题，使我明白了软件定义阶段对于一个项目的开发是至关重要的,当软件定义阶段完成时必须要用正式的文档准确的地记录目标系统的需求。只有前期的准备工作做得好，后面的工作才能顺利进行。虽然项目最后没有实现的很完美，但是起码我已经初步体会到软件项目开发的步骤，以及每一步所需要完成的文档等内容。关于编码，我认为编码要想做的完美必备条件就是前面的软件定义和软件设计时期要按部就班的做,文档一定要按要求书写,不能偷懒也不能草草书写。对于编码也要有相应的文档书写规范，要使源程序代码的逻辑简明清晰、易读易懂。这样当看到源程序代码的时候也能容易读懂代码的意思。其次就是测试的内容，我认为测试在软件开发中同样占据了重要的地位,它主要就是尽可能多的找到问题并排除其中的潜藏的错误，最终把一个高质量的软件系统交给用户使用。在此次软件工程文档写作考试过程中，除了让我明白工作中需要能力，知识之外，更重要的是学会了如何去完成一个任务，懂得了享受工作。当遇到问题，冷静，想办法一点一点的排除障碍，到最后获取成功，一种自信心就由然而生，这应该就是工作的乐趣。有时候不懂的就需要问别人了，虚心请教，从别人的身上真的能学到自己没有的东西，每一次的挫折都会使我更接近成功。

区块链教育证书系统中的教育证书文件是由各高校将学生基本信息和学年内的学习记录上链，由教育厅认证签名后上传至教育证书查询系统，供学生以个人区块链钱包账户登录查询的高校教育证书，相当于高校毕业证书。基于学生的教育证书文件的上链原理，其他人在生活和工作中所获得的由公认机构颁发的荣誉奖项也可上链。在终生学习的过程中，如若个人获得多种荣誉证书和奖项，用自己的区块链钱包账户登入官方的证书查询系统，系统将会呈现你获得的所有的荣誉奖项。也就是说，无需随身携带多种证明，无需查询多个方向记录，无需花费大量金钱去维护荣誉证书的有效性。只需使用区块链钱包账户登录区块链教育证书系统一键查询，它将解决大量人力、时间、资源浪费问题，确保证书的真实性和有效性，解决人与人之间的不信任问题。



**B190601**

**边昕旸**

**2021/1/4**