**基于区块链的教育证书系统**

目录

[一、前言 2](#_Toc58780877)

[1、电子证书的发展背景 2](#_Toc58780878)

[2、电子证书结合区块链技术需求十分迫切 2](#_Toc58780879)

[二、基于区块链教育证书系统的思维方案： 3](#_Toc58780880)

[1、基于区块链教育证书系统的架构设计 3](#_Toc58780881)

[2、基于区块链教育证书系统的运作流程： 6](#_Toc58780882)

[3、基于区块链教育证书系统的业务流程： 7](#_Toc58780883)

[三、基于区块链教育证书系统的优点： 8](#_Toc58780884)

[1、不可伪造： 8](#_Toc58780885)

[2、不可篡改： 9](#_Toc58780886)

[3、节约证书成本： 9](#_Toc58780887)

[4、信息可追溯： 9](#_Toc58780888)

[5、确权问题： 9](#_Toc58780889)

[6、隐私安全问题： 10](#_Toc58780890)

[四、基于区块链教育证书系统的部分代码实现： 10](#_Toc58780891)

[1、基于区块链教育证书的智能合约代码 10](#_Toc58780892)

[①证书被授予者信息（Recipient.sol） 11](#_Toc58780893)

[②教育证书信息（certmain.sol） 11](#_Toc58780894)

[2、基于Web3调用教育证书智能合约部署并校验代码 12](#_Toc58780895)

[①web调用certmain.sol实现信息上链（deploy\_cert3.js） 12](#_Toc58780896)

[②利用swarm将多个文件上传至区块链，返回的哈希回执写入证书，完成教育证书文件签名。（swarm\_server.js，swarm.js） 16](#_Toc58780897)

[Swarm\_server.js 16](#_Toc58780898)

[Babel-node 安装.txt 18](#_Toc58780899)

[Swarm.js 19](#_Toc58780900)

[③web调用Recipient.sol实现信息校验（upload.ejs） 22](#_Toc58780901)

[encrypt.js 26](#_Toc58780902)

[decrypt.js 28](#_Toc58780903)

[cert\_bxy.json 28](#_Toc58780904)

[五、基于区块链教育证书系统的有关问题： 30](#_Toc58780905)

[1、教育证书的个人数据在区块链上安全吗？ 30](#_Toc58780906)

[2、为什么区块链教育证书系统要使用区块链技术？ 31](#_Toc58780907)

[3、教育证书能否打印后使用？ 31](#_Toc58780908)

[4、个人能否查询他人的教育证书？ 31](#_Toc58780909)

[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？ 32](#_Toc58780910)

[6、教育证书可以被吊销吗？ 32](#_Toc58780911)

[7、教育证书如何实现个性化设计？ 32](#_Toc58780912)

[六、基于区块链教育证书系统的拓展用途： 33](#_Toc58780913)

# 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、基于区块链教育证书系统的思维方案：

## 1、基于区块链教育证书系统的架构设计

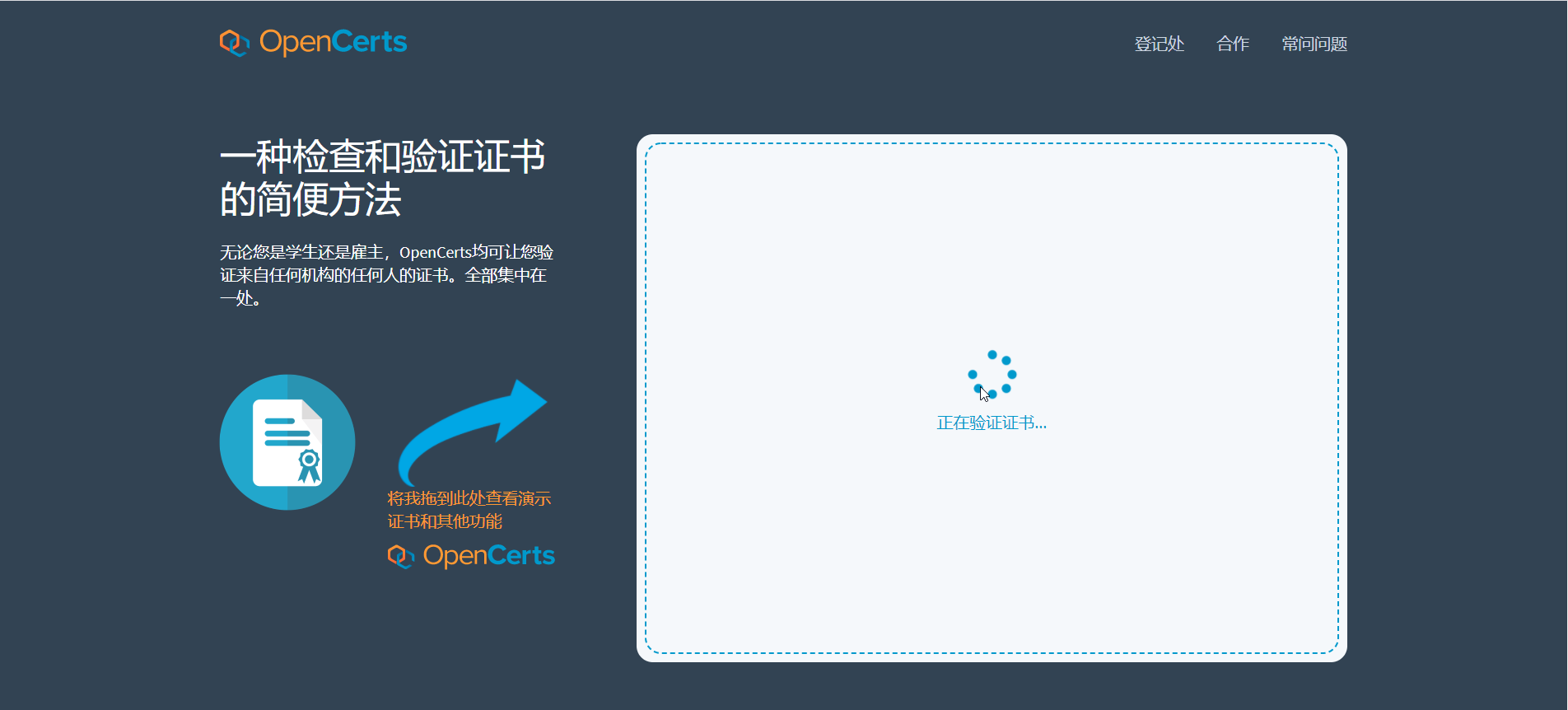
基于区块链的教育证书系统是一个用于发布教育证书的开源架构，也是一套用于为教育证书生成密码保护的工具。该系统大致由三个部分组成：视图、检查、校验。



**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。



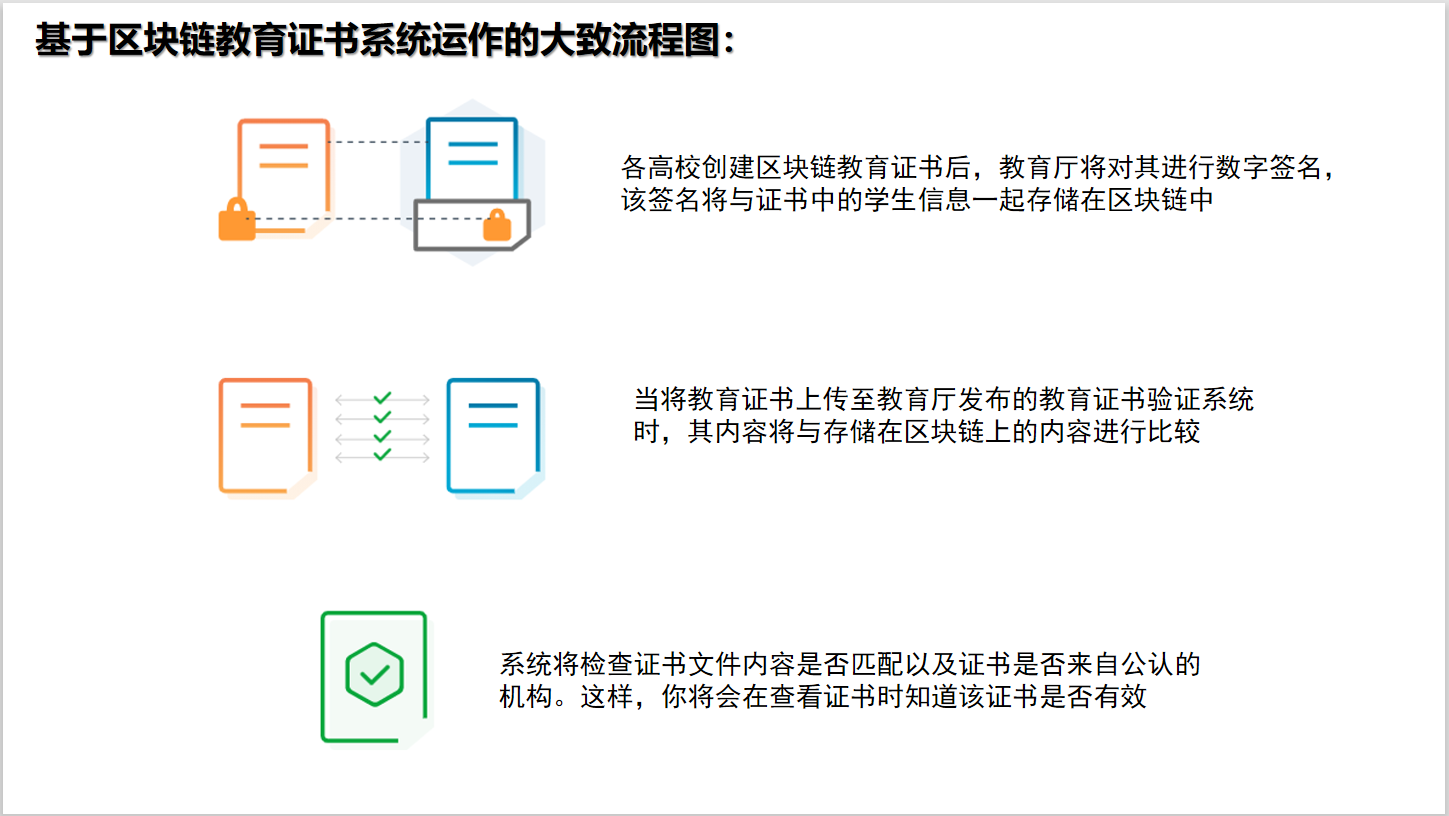
**检查教育证书文件是否被篡改**：基于区块链教育证书系统可根据内置在教育证书文件的高级密码保护自动检查拖拉上传至系统检验框的教育证书文件，将教育证书文件内的哈希回执和文件信息与上链信息相比较，从而达到检查教育证书文件是否有效的目的。



**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。

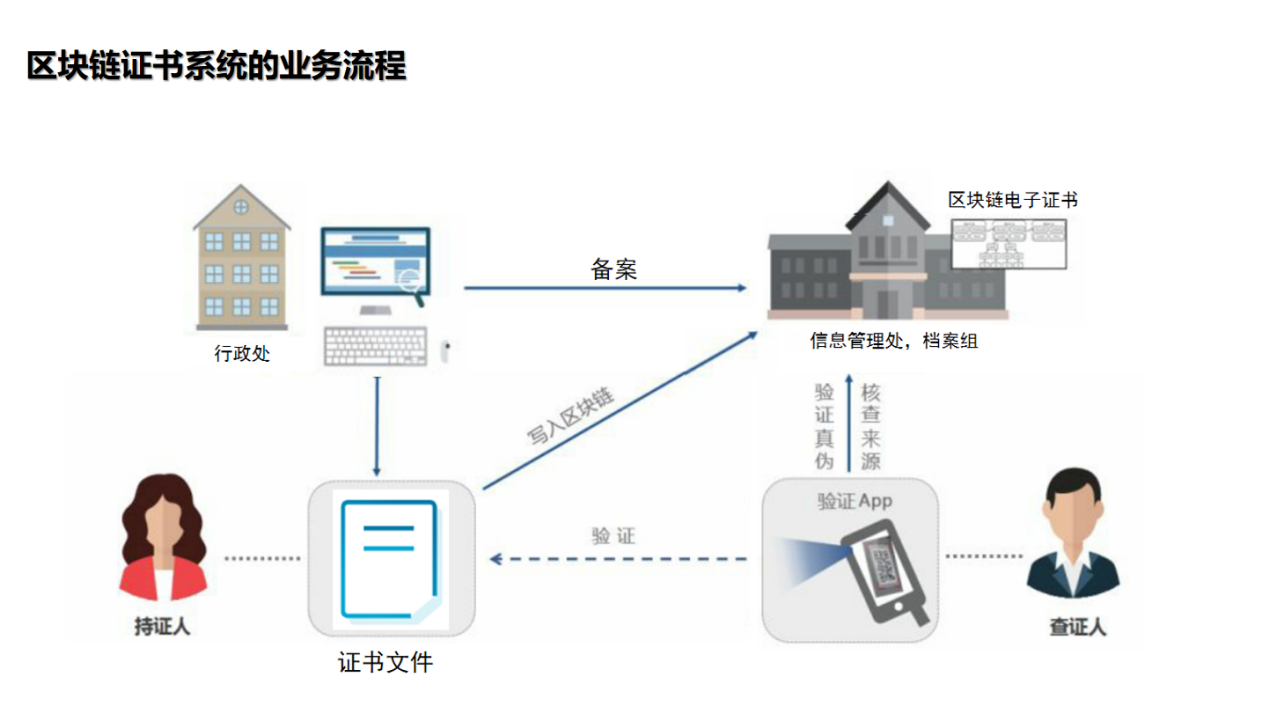
**②当将教育证书上传至教育厅发布的教育证书查询系统后，学生可通过区块链钱包账户登入系统得到教育证书文件，实现教育证书确权。**

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：

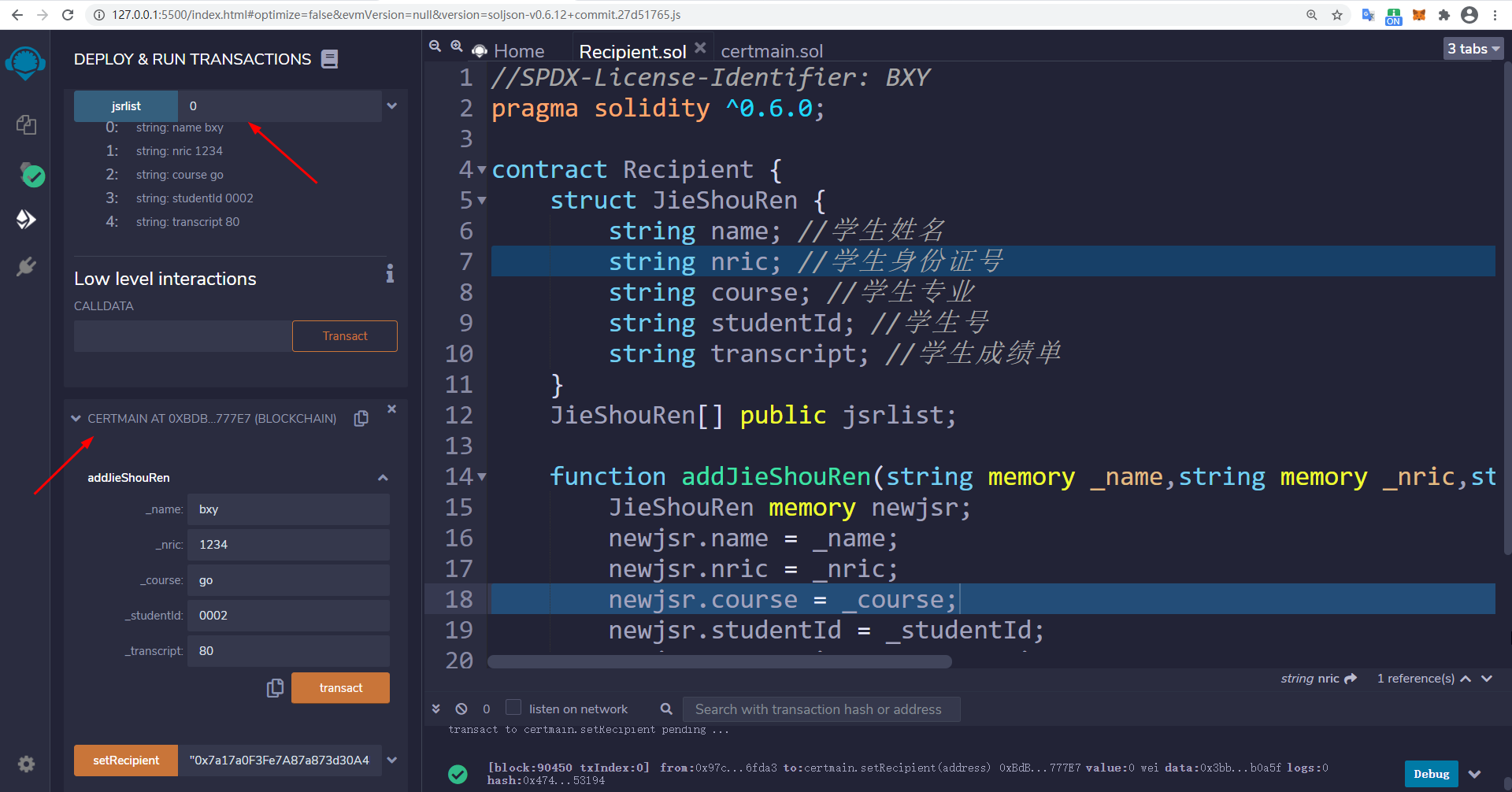
教育证书一旦发布，教育证书所有权归证书被授予者一人所有。授予者用区块链钱包账户（自身私钥）登入教育证书验证系统中的教育证书查询系统查询自己的教育证书文件，教育证书文件可手动下载保存至本地。此时将下载的教育证书文件拖拉上传至教育证书验证系统的验证框中，教育证书验证系统将自动处理上传的证书文件，此时方可查询被授予者自身的教育证书信息。若想要访问他人教育证书信息，首先需要获得他人的教育证书文件，然后用自身的区块链钱包登入教育证书验证系统，由于上传的教育证书所关联的区块链钱包账户和登入教育证书验证系统的区块链钱包账户不匹配，故每一次都需要支付相应数字货币才可查询他人教育证书文件，并且教育证书所有者可查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

## 1、基于区块链教育证书的智能合约代码



## **①**证书被授予者信息（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);

    }

}

## ②教育证书信息（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);

    }

}

­­­­­­

## 2、基于Web3调用教育证书智能合约部署并校验代码

## ①web调用certmain.sol实现信息上链（deploy\_cert3.js）

const express = require('express')

const app = express()

const path = require("path");

let Web3 = require('web3')

let url\_geth = 'ws://127.0.0.1:8546'

let web3 = new Web3(url\_geth)

// 部署

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 = '0xBdB7cbAEefD6838a9F903423aA6F18ff910777E7'

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

const Recipient\_address = '0xBdB7cbAEefD6838a9F903423aA6F18ff910777E7'

const My\_address = '0xBdB7cbAEefD6838a9F903423aA6F18ff910777E7'

app.get('/submit', function (req, res) {

    const name = req.query.name

    const nric = req.query.nric

    const course = req.query.course

    const studentId = req.query.studentId

    const transcript = req.query.transcript

    certmain\_contract.methods.setRecipient(Recipient\_address).send({ from: My\_address }, (error, result) => {

        certmain\_contract.methods.addJieShouRen(name, nric, course, studentId, transcript).send({ from: My\_address }, (error, result) => {

            if (!error) {

                console.log('回执1:', result)

                res.render('submit.ejs', {

                    msg: { code: "00", txt: "信息上链成功，哈希回执为：" + result +"\n教育证书文件已保存在public目录下" }

                });

            }

        })

    })

    const fs = require("fs");

    const schema = "opencerts/v2.0";

    const deploy = Recipient\_address;

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

    fs.writeFile('./public/cert3.json', JSON.stringify({ schema, deploy, data }), 'utf8', function (error) {

        if (error) {

            console.log(error);

            return false;

        }

        console.log('写入成功');

    })

})

app.use(express.static(path.join(\_\_dirname, "public")));

app.listen(1113)

app.get("/submit3", (req, res, next) => {

    res.render("submit.ejs")

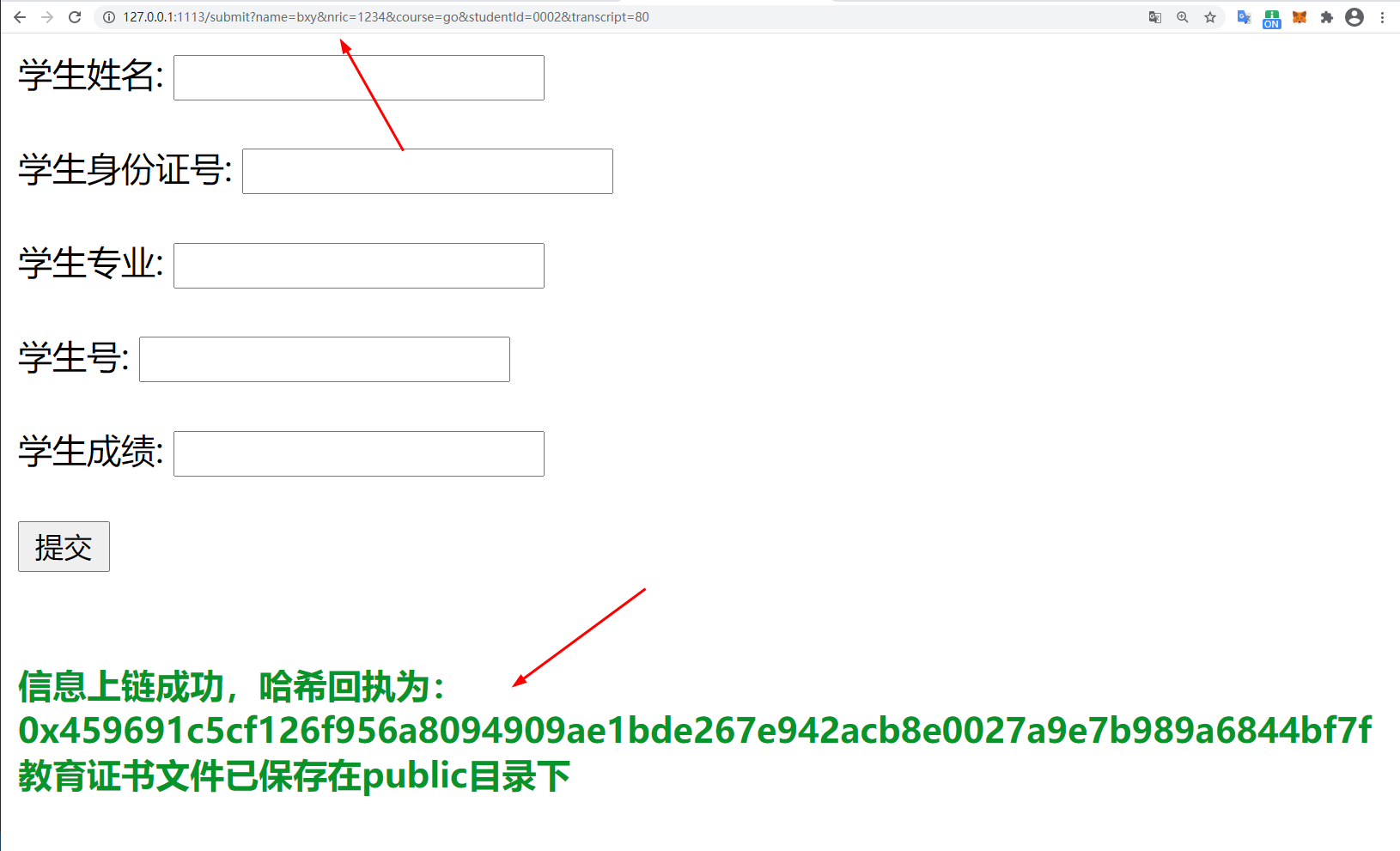
});

app.get("/upload", (req, res, next) => {

    res.render("upload.ejs")

});

启动本地以太坊私链，执行nodemon deploy\_cert3.js命令，打开<http://127.0.0.1:1113/submit3>网址出现以下页面，然后高校可以输入学生在校信息，确认无误后点击提交按钮，后台web调用certmain.sol合约将学生信息记录于区块链，一段时间后若信息上链成功，将会出现成功提示，显示上链信息哈希回执于该页面，并利用fs将教育证书文件自动写入public文件目录下（附下图参考）。



## ②利用swarm将多个文件上传至区块链，返回的哈希回执写入证书，完成教育证书文件签名。（swarm\_server.js，swarm.js）

## Swarm\_server.js

const http = require('http');

var moment = require('moment');

moment.locale('zh-cn');

moment().format('LLL');

const socketio = require('socket.io');

var onlineClientCount = 0;

var ClientMap = new Map();

const server = http.createServer(

    (req, res) => {

        res.end("I am connected!SOcket.IO");

    }

);

const io = socketio(server);

io.on('connection', (clientsocket, req) => {

    onlineClientCount++;

    console.log('客户端上线:', clientsocket.id, '当前在线客户端', onlineClientCount);

    ClientMap.set(clientsocket.id, "User" + onlineClientCount);

    console.log(ClientMap)

    io.emit('onlineClientCount',  onlineClientCount)

    clientsocket.emit('welcome', 'Hello 你好@ Socketio!我来自服务器');

    setInterval(function () {

        clientsocket.emit('msg\_p2p', '我是P2P消息:' + clientsocket.id + '==>' + moment.parseZone(new Date()).format('YYYY/MM/DD  hh:mm:ss a'))

    }

        , 3000);

    clientsocket.on('msg\_p2p', (msg) => {

        console.log('接收到客户端的消息 data:', msg);

        clientsocket.emit('returnmessage', '我已经收到消息：' + msg + new Date())

    })

    clientsocket.on('disconnect', () => {

        onlineClientCount--;

        console.log('客户端下线:', clientsocket.id,'当前在线客户端' , onlineClientCount);

        io.emit('onlineClientCount',  onlineClientCount)

        ClientMap.delete(clientsocket.id)

        console.log(ClientMap)

    })

    clientsocket.on('from\_hash', (msg) => {

        console.log('用户上传Hash:', msg);

        clientsocket.emit('to\_go', '本地消息====>>>写入文件hash：' + msg )

    })

});

setInterval(function () {

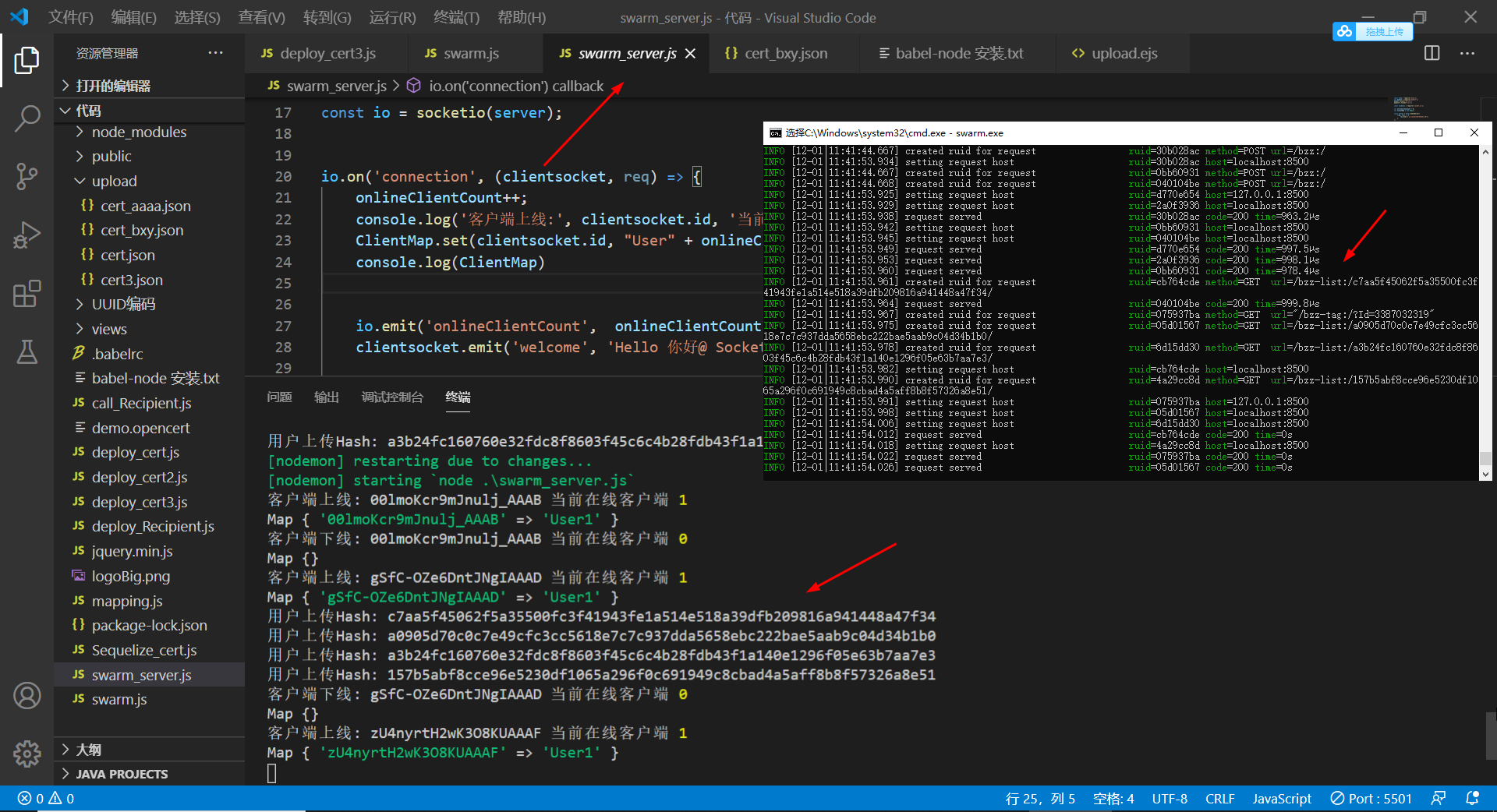
    io.emit('msg\_broadcast', '我是广播消息,3秒钟发送一次' + new Date())

}

    , 3000);

server.listen(8000);

启动搭建的swarm链，执行nodemon swarm\_server.js命令使其与swarm端口连接，终端输出信息来反应接收到的swarm信息。当连接swarm成功时，终端将输出“客户端上线…，当前在线客户端 …”的指示信息。当swarm上传文件后，待swarm\_server信息接收成功，终端输出“用户上传Hash…”的指示信息。（附下图参考）



## Babel-node 安装.txt

参考页面 ：https://blog.csdn.net/ll837448792/article/details/103307796

babel-node 5erebos.js

报错，SyntaxError: Cannot use import statement outside a module===

babel的安装

============

npm install --save babel-core

npm install --save babel-preset-env 或者 npm install --save babel-preset-es2015

npm install babel-cli -g

=============

接着在项目根目录下创建一个名字为.babelrc的文件， 文件内容入如下

===

{

    "presets": [

      "es2015"

    ],

    "plugins": []

}

====

或者

{

    "presets": [

         "env"

     ],

    "plugins": []

}

接下来可以用npm start和babel-node app.js运行项目。

运行nodejs项目，npm start启动项目import报错，SyntaxError: Cannot use import statement outside a module，可能原因是nodejs不支持import语法，故使用babel来支持，解决报错问题。（附上图Babel-node 安装.txt文件代码参考）

## Swarm.js

//命令行运行命令babel-node .\s.js

import { BzzFS } from '@erebos/bzz-fs'

import { BzzNode } from '@erebos/bzz-node'

var sd = require('silly-datetime');

var time=sd.format(new Date(), 'YYYY-MM-DD HH:mm');

const fs = require("fs");

const io = require('socket.io-client');

const socket = io("http://127.0.0.1:8000");

const bzz = new BzzNode({ url: 'http://localhost:8500' })

const bzzFS = new BzzFS({ bzz })

var readDir = fs.readdirSync("./upload");

console.log(readDir);

console.log("总共上传" + readDir.length + "个文件，分别是" + readDir)

for(var i=0; i < readDir.length; i++) {

        (function(i) {

            fs.readFile(readDir[i], function(err, contents) {

    //             console.log(readDir[i]);

                var s1 = readDir[i]

                var filename = './upload/'+s1

            fs.stat(filename,function(err,stats){

                // console.log("文件信息是:",stats);

                console.log(readDir[i]+"文件大小是:",stats.size);

            fs.exists(filename, (exists) => {

                 if (exists) {

                    upload(filename)

                } else {

                    console.log("文件不存在");

            }

        });

function upload(filename) {

    bzzFS

        .uploadFileFrom(filename, { contentType: 'text/plain' })

        .then(hash => {

            console.log(readDir[i] + "的上传hash是:" ,hash)

            console.log(readDir[i] + "的上传时间戳是:" ,time)

                socket.emit('from\_hash',hash);

            return bzz.list(hash)

        })

}

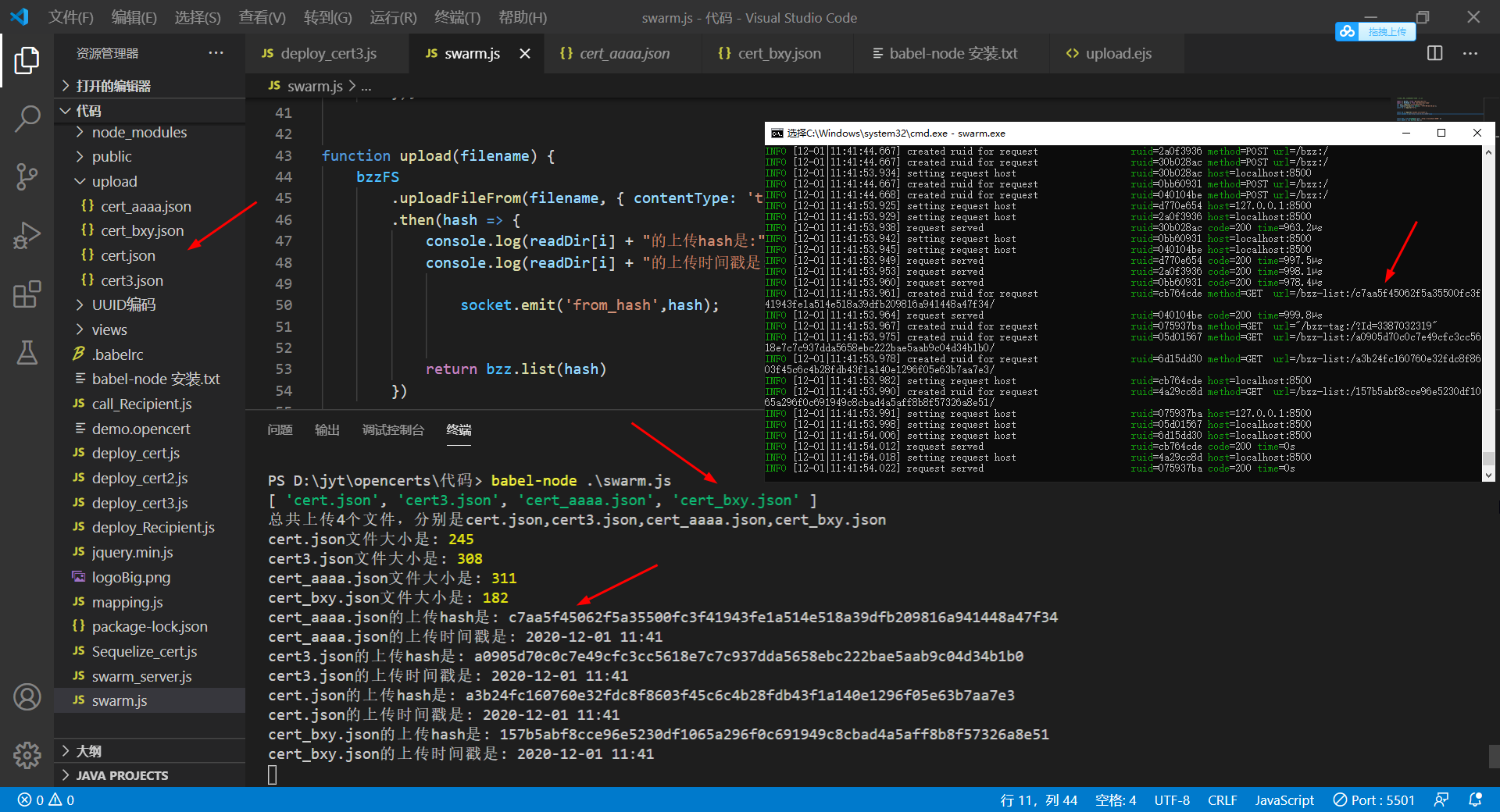
}) // Manifest contents describing the uploaded files

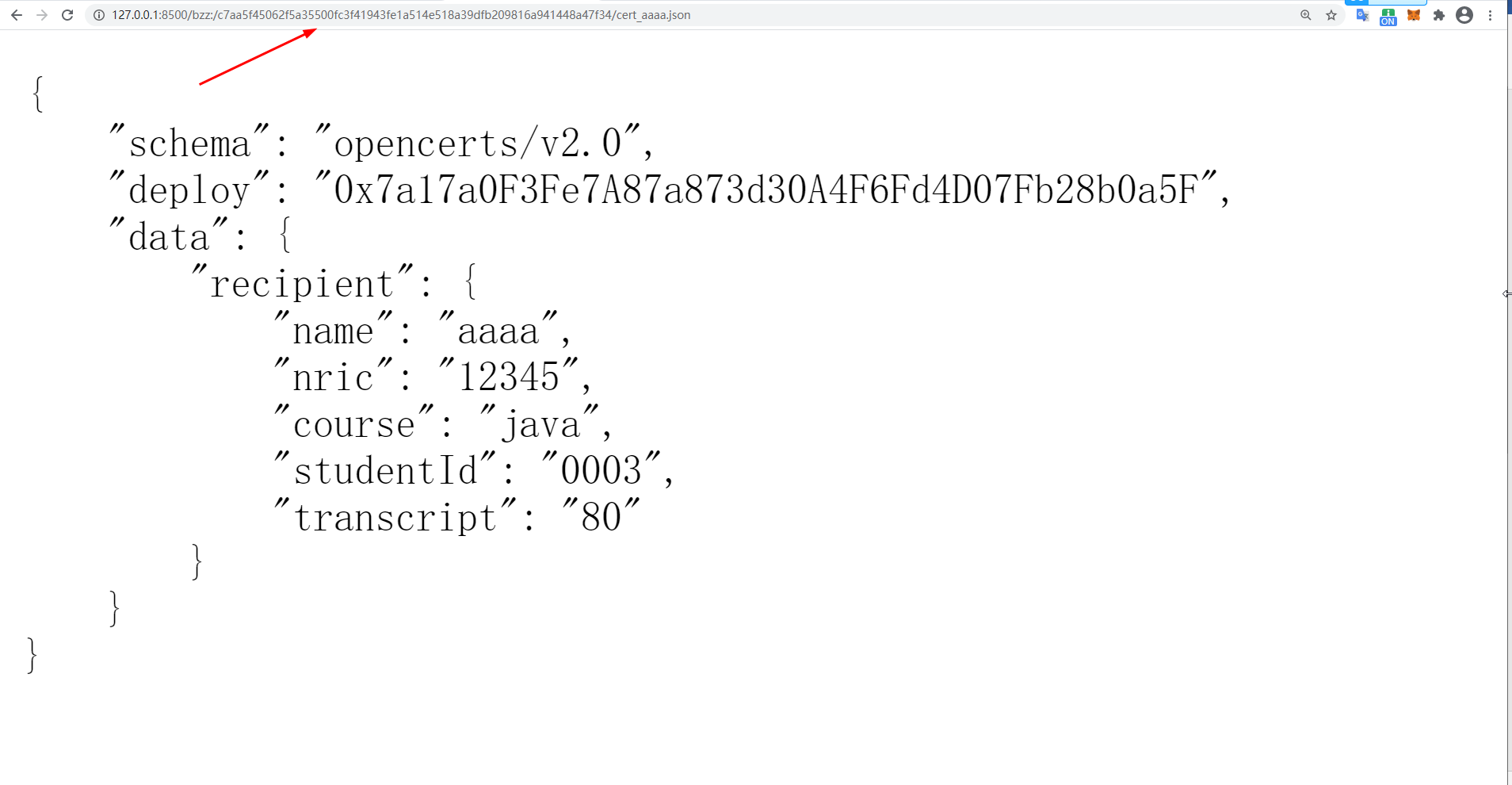
        });

    })(i);

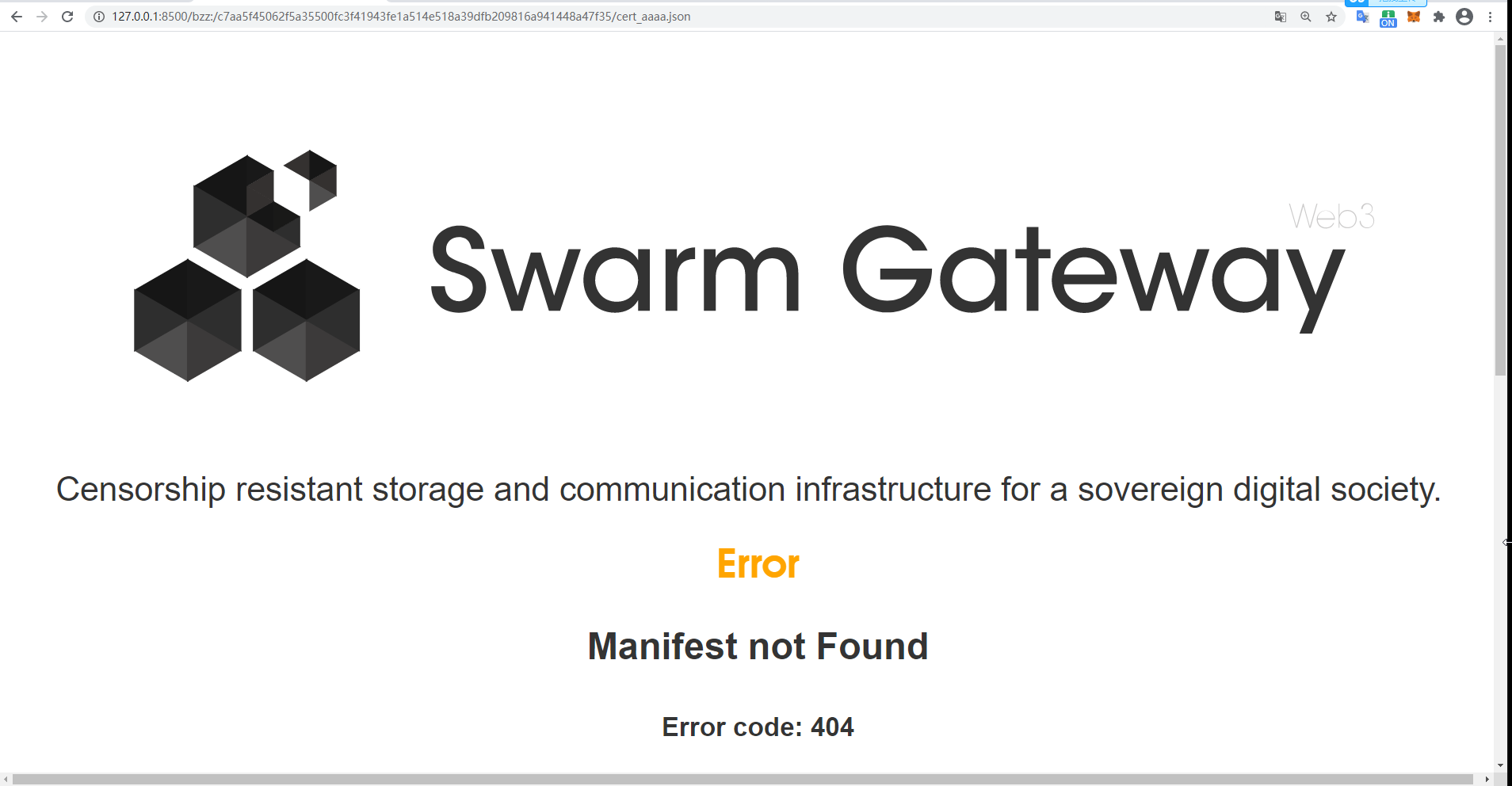
}

启动搭建的swarm链，执行nodemon swarm\_server.js命令使其与swarm端口连接，终端输出信息来反应接收到的swarm信息。当连接swarm成功时，终端将输出“客户端上线…，当前在线客户端 …”的指示信息。当swarm上传文件后，待swarm\_server信息接收成功，终端输出“用户上传Hash…”的指示信息。（附下图参考）





选择文件，利用nodejs将教育证书文件上传至swarm，进度条拉完后上传成功



## ③web调用Recipient.sol实现信息校验（upload.ejs）

<!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>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="/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) {

                                $('#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>' +

                                    "签名：" + json.deploy + '</p>'

                                    + '</p>');

                            } 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>

# encrypt.js

//加密签名

const ethers = require('ethers');

const fs = require("fs");

// 账号0x97cDD303dc9EB548aba7fc429735891FcA46fda3

let privateKey = "0xaxxxeb629879";

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);

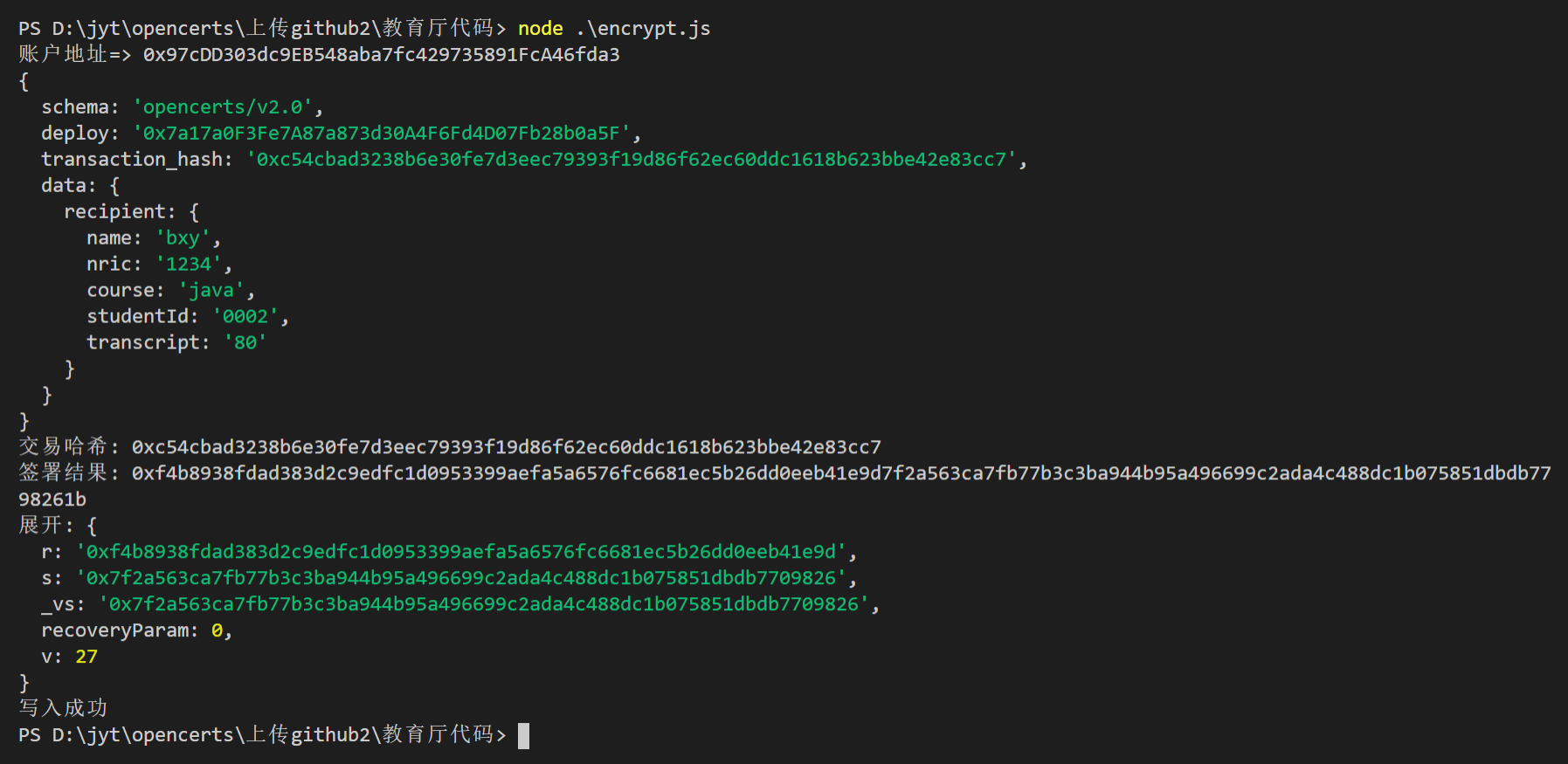
            return false;

        }

        console.log('写入成功');

    })

});



## decrypt.js

//解密验证

const fs = require("fs");

const ethers = require('ethers');

let accountAddress = "0x97cDD303dc9EB548aba7fc429735891FcA46fda3";

console.log('账户地址:', accountAddress);

var data = fs.readFileSync('./signature/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);

console.log("合约签名:",json.signature);

let hash = json.transaction\_hash;

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

let signature=json.signature;

    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！！！')

    } else {

        console.log('验证失败，非公认机构发布证书！！！')

    }

## cert\_bxy.json

{

    "schema": "opencerts/v2.0",

    "deploy": "0x7a17a0F3Fe7A87a873d30A4F6Fd4D07Fb28b0a5F",

    "transaction\_hash": "0xc54cbad3238b6e30fe7d3eec79393f19d86f62ec60ddc1618b623bbe42e83cc7",

    "data": {

        "recipient": {

            "name": "bxy",

            "nric": "1234",

            "course": "java",

            "studentId": "0002",

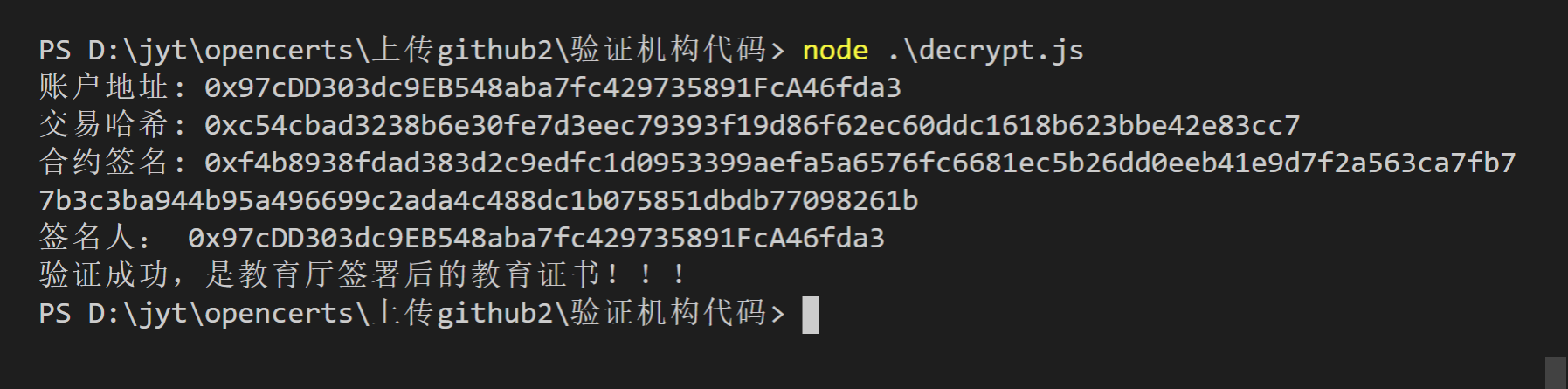
            "transcript": "80"

        }

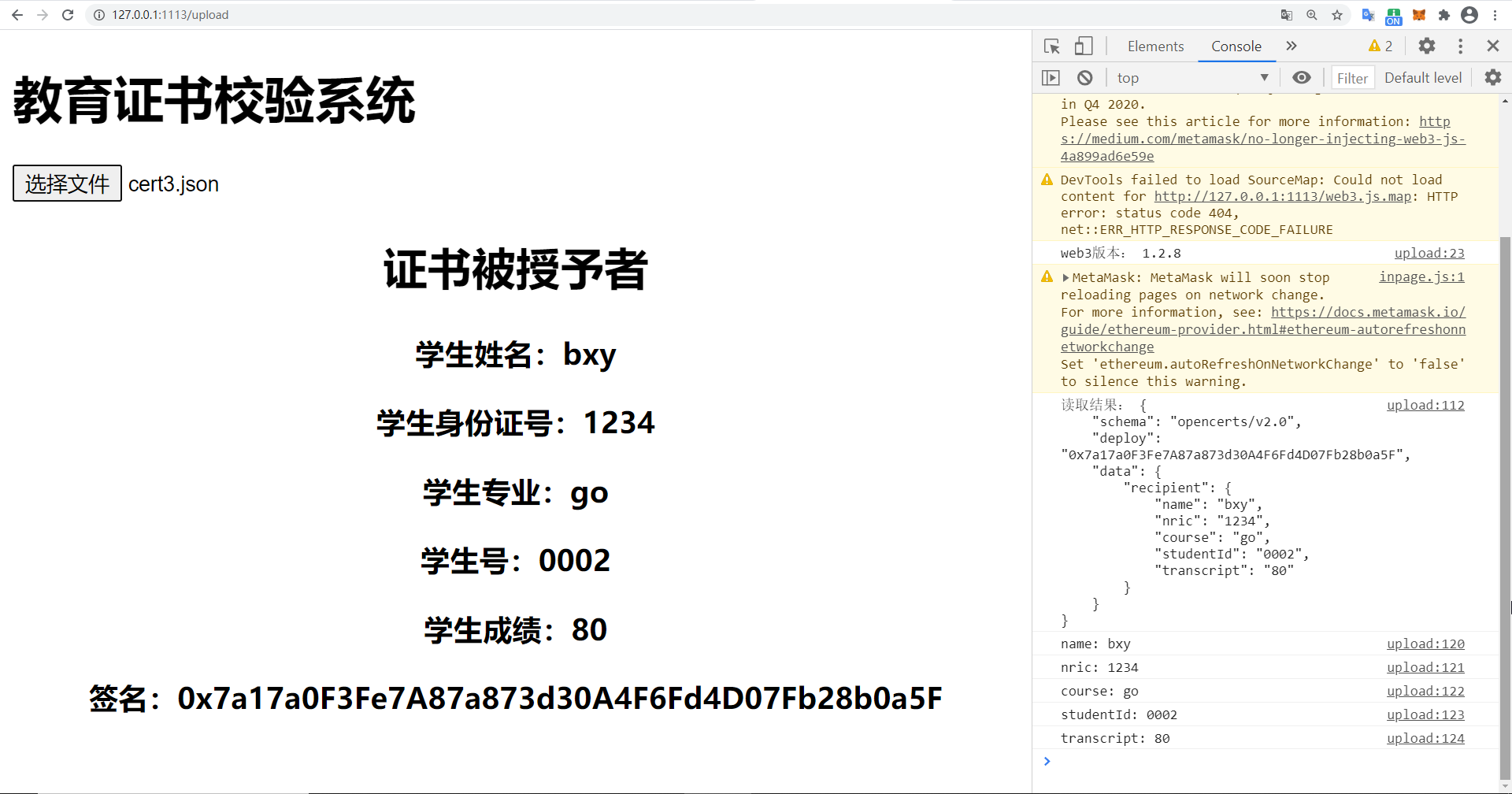
    },

    "signature": "0xf4b8938fdad383d2c9edfc1d0953399aefa5a6576fc6681ec5b26dd0eeb41e9d7f2a563ca7fb77b3c3ba944b95a496699c2ada4c488dc1b075851dbdb77098261b"

}



启动本地以太坊私链，执行nodemon deploy\_cert3.js命令，打开<http://127.0.0.1:1113/upload>网址出现以下页面，此时可以选中上一步信息上链成功的教育证书文件。Web将会根据Recipient.sol合约文件的abi文件、合约部署地址、合约的call方法在区块链上查找上链记录并与上传的教育证书文件内容逐一进行对比。当上传证书文件内所有信息都与区块链记录信息相符合，则该上传证书文件信息将被成功打印于此页面上（附下图参考）。



如果上传的教育证书文件内的签名被篡改或为无效签名，系统将无法通过校验，并且返回“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）。



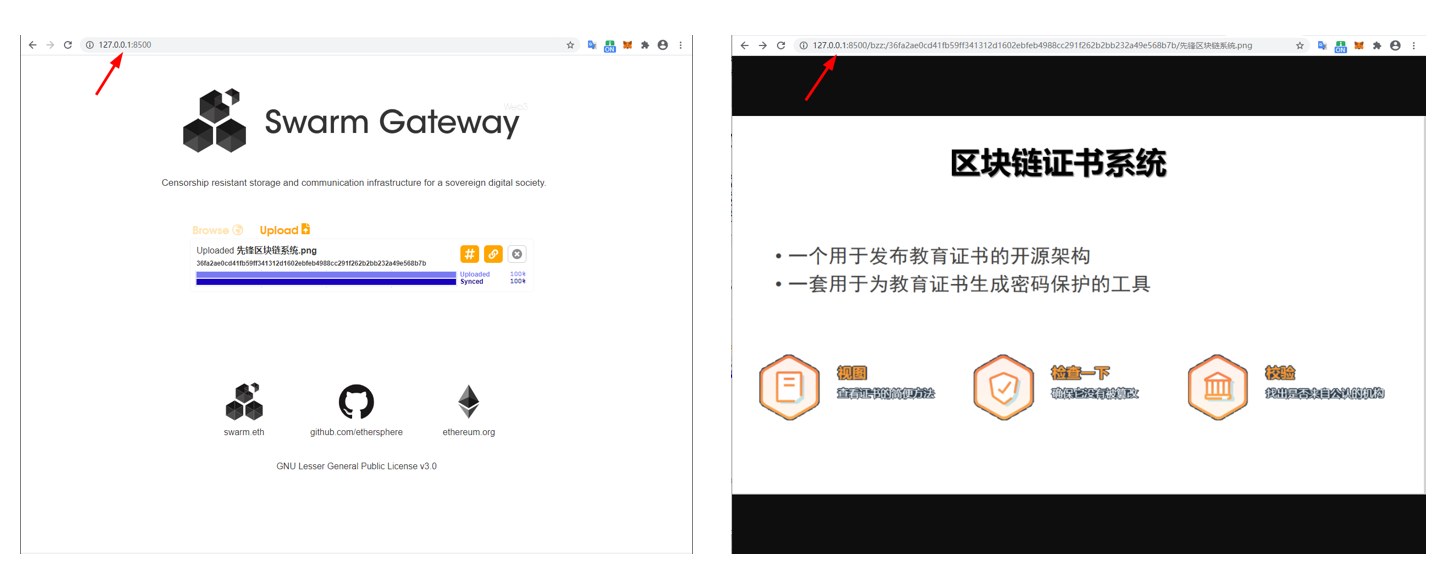
如果上传教育证书文件的签名为有效签名，但文件内容与上链信息内容不匹配，系统也将无法通过校验，并且返回“教育证书已被篡改，无法读取证书信息！！！”的错误提示（附下图参考）。



# 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？

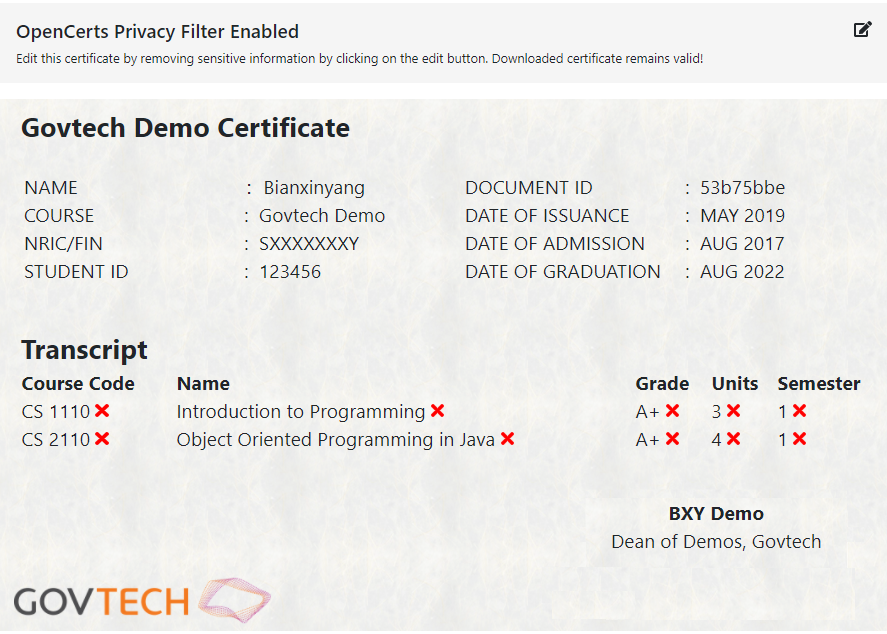
教育证书文件使用是安全的。教育证书文件可以轻松复制，但是，如果不通过教育证书验证系统的验证过程，则无法更改教育证书中的收件人姓名。因此，进行验证的人员确保教育证书中指示的接收者实际上是出示教育证书的本人，这一点极为重要。对于更高级的机构，将来可能会使用分布式ID来关联教育证书接收者的公钥，以便将来使用公钥/私钥签名进行进一步的身份验证。

## 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、基于区块链教育证书系统的拓展用途：

之前所说的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。

