Audit Smart Contract项目文档

> **AUTHOR:**
>
> **刘长硕 3170104593 工作占比33.3%**
>
> **章雨婷 3170102582 工作占比33.3%**
>
> **金一博 3170103634 工作占比33.3%**

> **Date : 2020-1-03**

Contents

Chapter 1 : 创意简介

1.1 项目简介

1.1.1 项目背景

项目名称：智能合约审计平台

用户：智能合约撰写用户和审核者

- **相关背景介绍**

  智能合约作为在没有第三方参与的情况下进行可信交易的保障，其安全性、隐私性和逻辑性需要非常缜密的判断证明。现有的形式化验证等自动化证明工具还不是非常成熟，不足以找出智能合约的所有潜在漏洞，无法完全确保智能合约的安全性，所以智能合约的审计工作仍需要人工参与。

  为了确保人工审计的结果安全性、透明性，区块链作为一个去中心化的应用，能够保证人工审计报告的及时公示和不被篡改，通过将审计报告上链和身份确认的形式，确保审计报告的来源可靠性。最终实现一个公开公正透明的智能合约审计平台。

1.1.2 项目概述

智能合约的审计平台是一个基于以太坊的智能合约提交、审计、结果生成系统。该系统模拟了一个智能合约审计公司的实际场景：

智能合约审计平台用户分为两类：智能合约上传用户，智能合约审计人员。当平台用户在DAPP上提交了智能合约后，抵押一定数量的以太币，系统会自动将智能合约分发给5个在线的智能合约审计人员。公司工作人员（智能合约审计人员）上班后，将自己的状态调整为在线，表示可以接收审计报告并审核，当完成审计后，审计人员获得一定积分（积分和其审核的效果相关），将审计人员的审计报告汇总起来之后，用户将支付抵押的以太币获得审计报告。用户和审计人员在提交智能合约和审计合约之间进行利益交换，实现各自的目标。

1.1.3 项目创新点

- **提高报告可信度和不可篡改性**

  项目主要的创新点是让审计人员的报告可信度得到提高，现今市场上的各项智能合约审计平台大多是通过专业安全团队咨询审计，报告提交者无法实时具体了解审计过程以及审计提交报告的可信度。而采用审计报告上链的形式，能够用户实时监测审计人员的报告审计进度。

- **去中心化防止审计人员对结果的操作**

  审计人员无法观测到其他审计人员的审计结果，审计人员也无法了解其余审计人员的真实身份，防止审计人员中出现对审计报告结果的操纵情况。

- **激励措施机制**

  用户提交报告进行审计需要支付一定数量的以太币，作为一种激励措施，使审计人员能够积极参与到报告审计过程中。

- **消极审计惩罚机制**

其次，在该项目的智能合约审计平台上，审计人员的审计工作关系到其收益，通过将所有审计人员提交的审计报告进行综合，得到最终的审计报告，与最终审计报告结果相差较大的审计人员将会被认定为消极审计，扣除其收益。通过将最终审计结果与审核人员的审计结果进行异或得到审核人员的积分，而公司可以将其工资与积分挂钩。这一举措可以提高审计人员的工作积极性和审计质量。

1.1.4 项目应用范围

智能合约审计平台的用户是社会中具有智能合约编写能力的社会成员和具有智能合约开发经验的专业平台审计人员。智能合约提交者具有智能合约安全性审计的需求，审计人员具有专业经验和开发经历，能够满足提交者的需求。

该项目面向广大需要检测并提高其智能合约安全性的用户，能够适用于各种专业研究方向和各种语言的智能合约，同时支持审计人员的审计工作并，运用以太坊支付机制作为审计人员的工作量回报。

1.2 项目意义

- **社会效益**

  满足了社会上部分智能合约书写者对其智能合约安全性的需求，提供了智能合约审计平台来满足实时的审计工作。减少了智能合约安全漏洞对社会经济造成的重大损失，众所周知，智能合约安全漏洞的存在将对企业项目造成毁灭性打击，通过智能合约审计，提前避免合约中可能出现的漏洞并进行修改，规避因合约安全问题导致的财产损失。

- **经济效益**

  智能合约审计平台占用资金少，成本支出少，主要通过虚拟货币以太币进行项目交互，能够解决大量智能合约的安全性问题，能够产生的大量的有用成果。

1.3 计划目标

实现一个基于以太坊的智能合约审计平台。在用户方面，审计平台做到实时上传，并在固定的时间内获得优质的智能合约审计报告，进而降低智能合约的风险。在审计人员方面，审计平台做到实时显示用户上传的智能合约，提醒审计人员在固定的时间完成相应审计并提交。在以太坊方面，审计平台实现审计人员新的积分值和审计报告hash值的上链。

Chapter 2 : 代码部署

底层环境搭建

搭建环境

os: ubuntu 16.04

tool: wget,nodejs,ganache,web3,solc,truffle,webpack

ganache

通过在虚拟机上安装ganache自动搭建私有网络进行区块链网络的搭建。

MetaMask

通过在chrome浏览器上使用MetaMask钱包应用与ganache的私有网络进行联合使用，获取钱包应用接口

webpack

然后用webpack模版初始化项目骨架结构

```
~/demo$ truffle unbox webpack
Downloading…
Unpacking…
Setting up…
Unbox successful. Sweet!
```

truffle

```
module.exports = {
  networks:{
    development: {
      …
      port: 8545
      …
    }
  }
}
```

将truffle的端口改为8545

2.1 前端

2.1.1 环境配置

前端使用nodejs、bootstrap、express和react框架

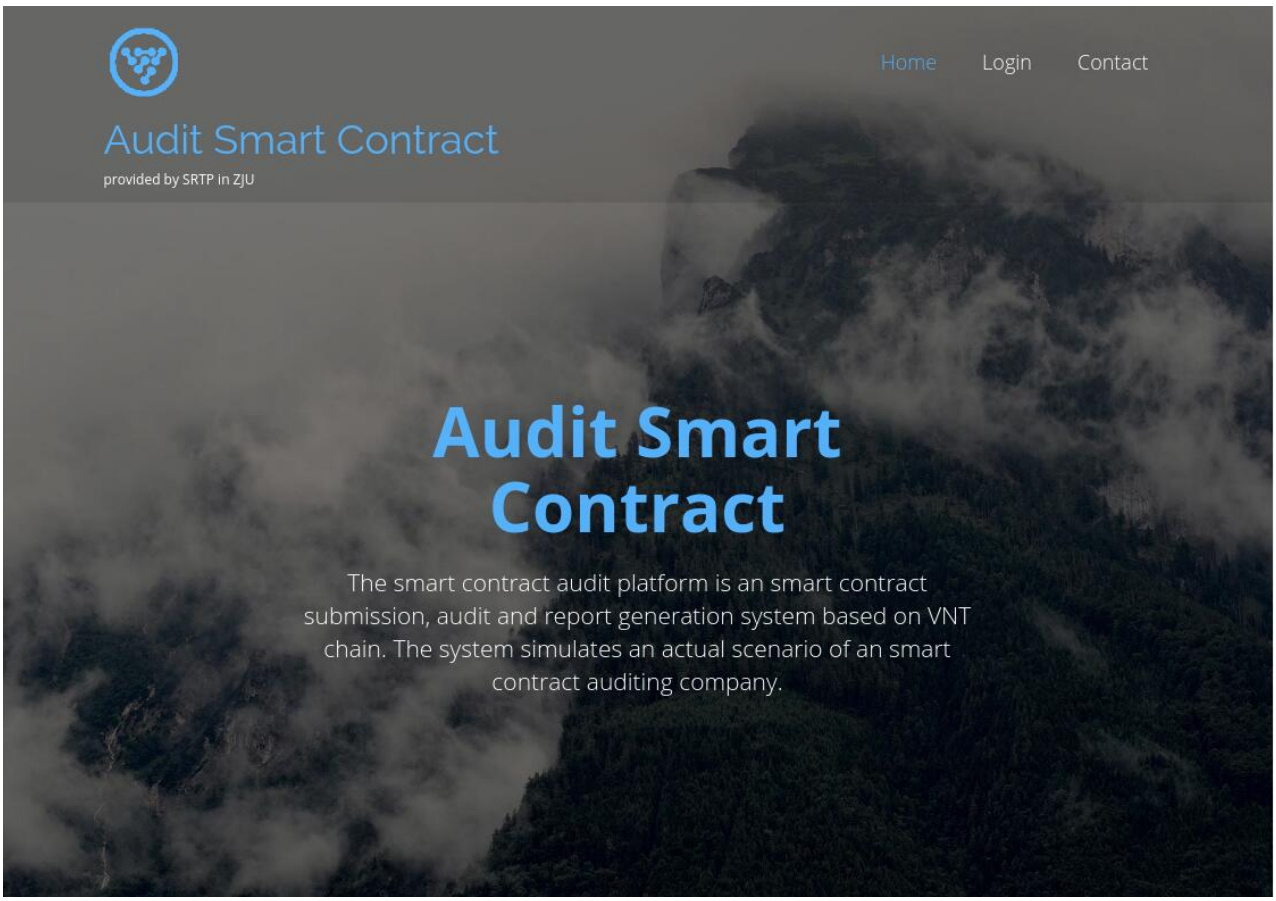
- npm和nodejs：安装完成最新的npm和nodejs。
- Bootstrap：在terminal内输入bower install bootstrap完成安装bootstrap。
- Go语言：在terminal内输入sudo apt-get install golang-go即可安装Go。
- Express和react框架：在terminal中输入npm install express -g 安装Express框架
  输入npm install -g create-react-app安装react框架

2.1.2 代码部署

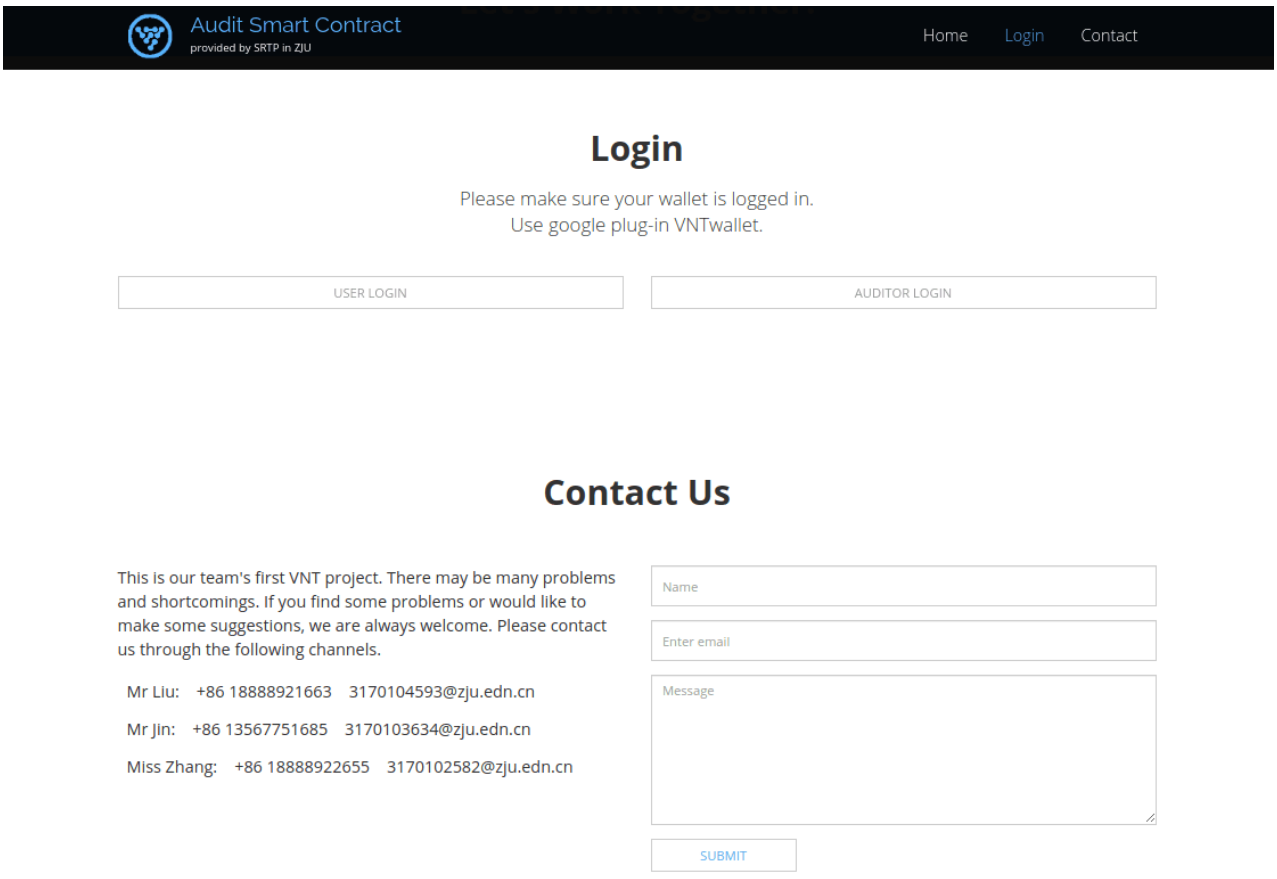
前端分为主页面、用户页面和审计人员页面三部分。

- 主页面(index.ejs)

  （1）主页显示


Audit Smart Contract
provided by SRTP in ZJU
Home
Login
Contact
Audit Smart Contract
The smart contract audit platform is an smart contract submission, audit and report generation system based on VNT chain. The system simulates an actual scenario of an smart contract auditing company.

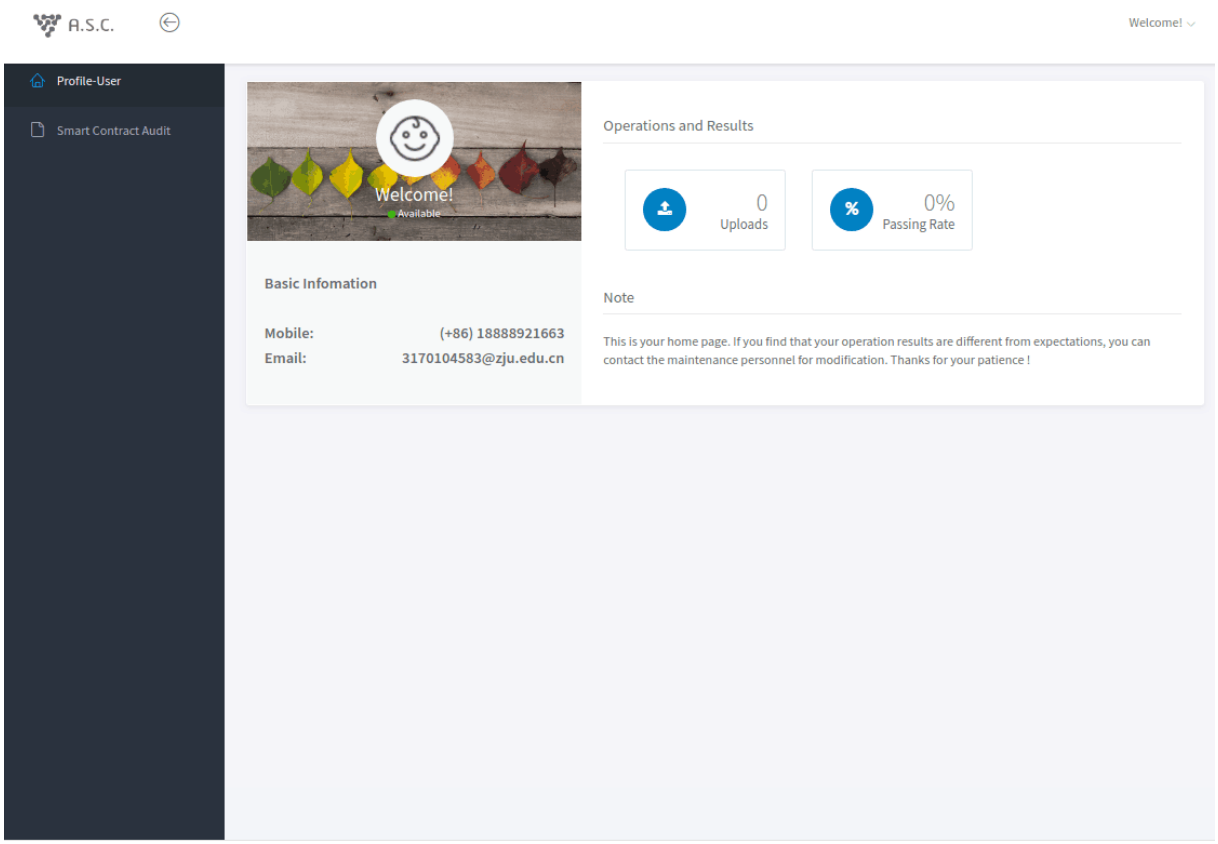
（2） 登录界面


Audit Smart Contract
provided by SRTP in ZJU
Home
Login
Contact
Login
Please make sure your wallet is logged in.
Use google plug-in VNTwallet.
USER LOGIN
AUDITOR LOGIN
Contact Us
This is our team's first VNT project. There may be many problems and shortcomings. If you find some problems or would like to make some suggestions, we are always welcome. Please contact us through the following channels.
Mr Liu: +86 18888921663 3170104593@zju.edn.cn
Mr Jin: +86 13567751685 3170103634@zju.edn.cn
Miss Zhang: +86 18888922655 3170102582@zju.edn.cn
Name
Enter email
Message
SUBMIT

- 用户页面

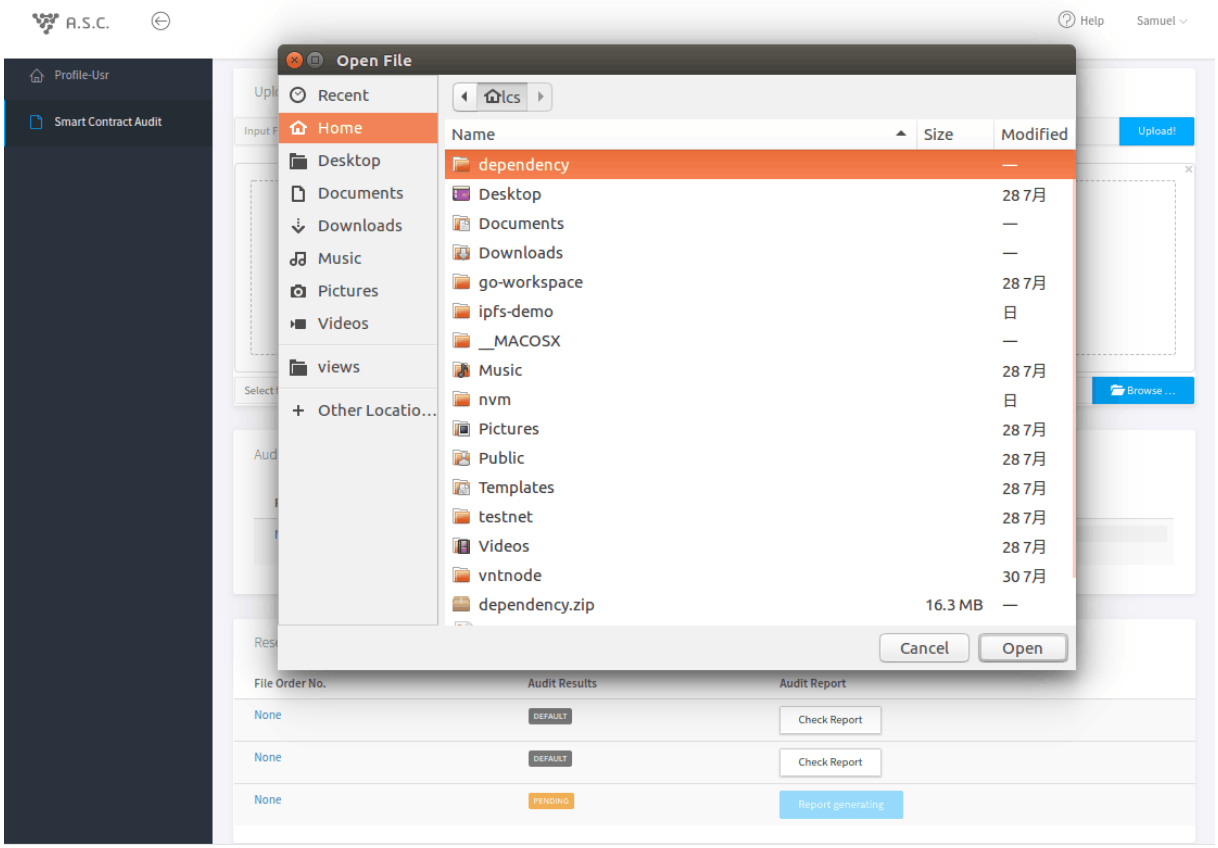
- 用户主页面(profileuser.ejs)

  （1） 个人信息简介

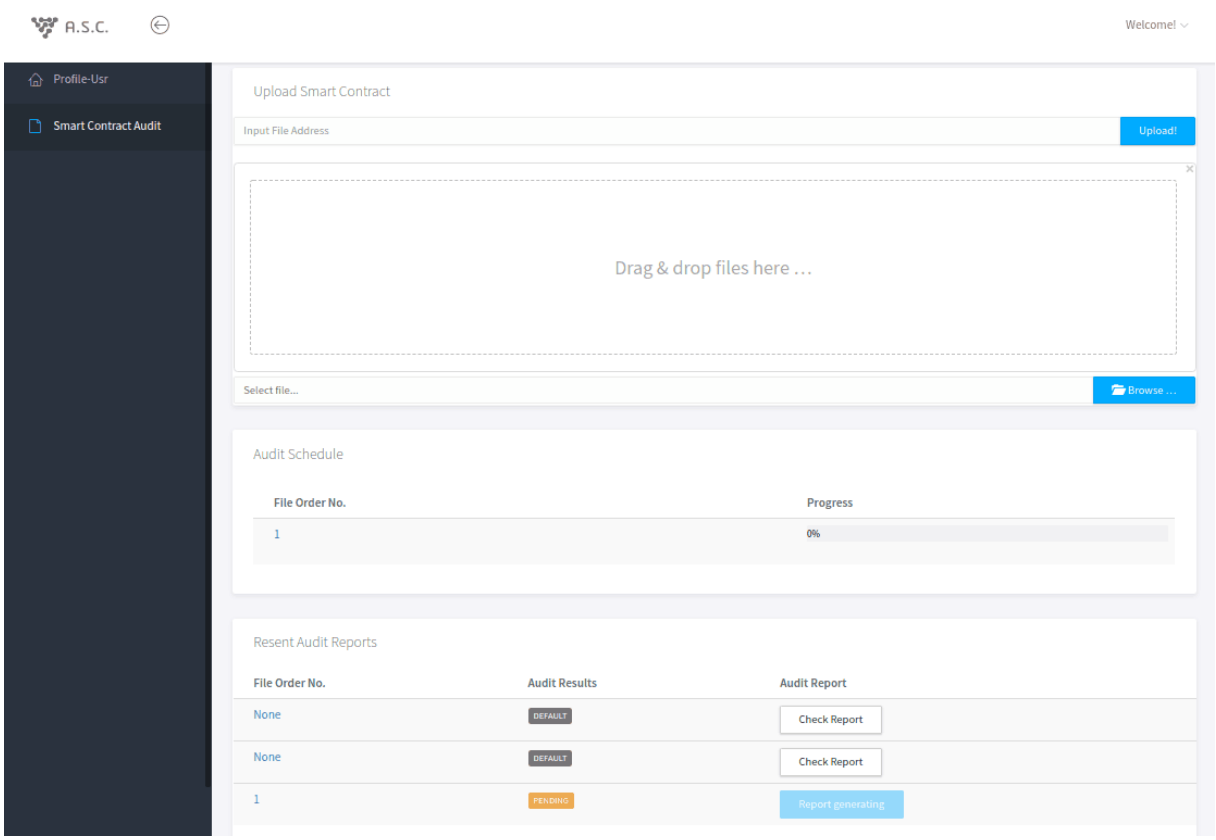


- 用户功能页面(upload.ejs)

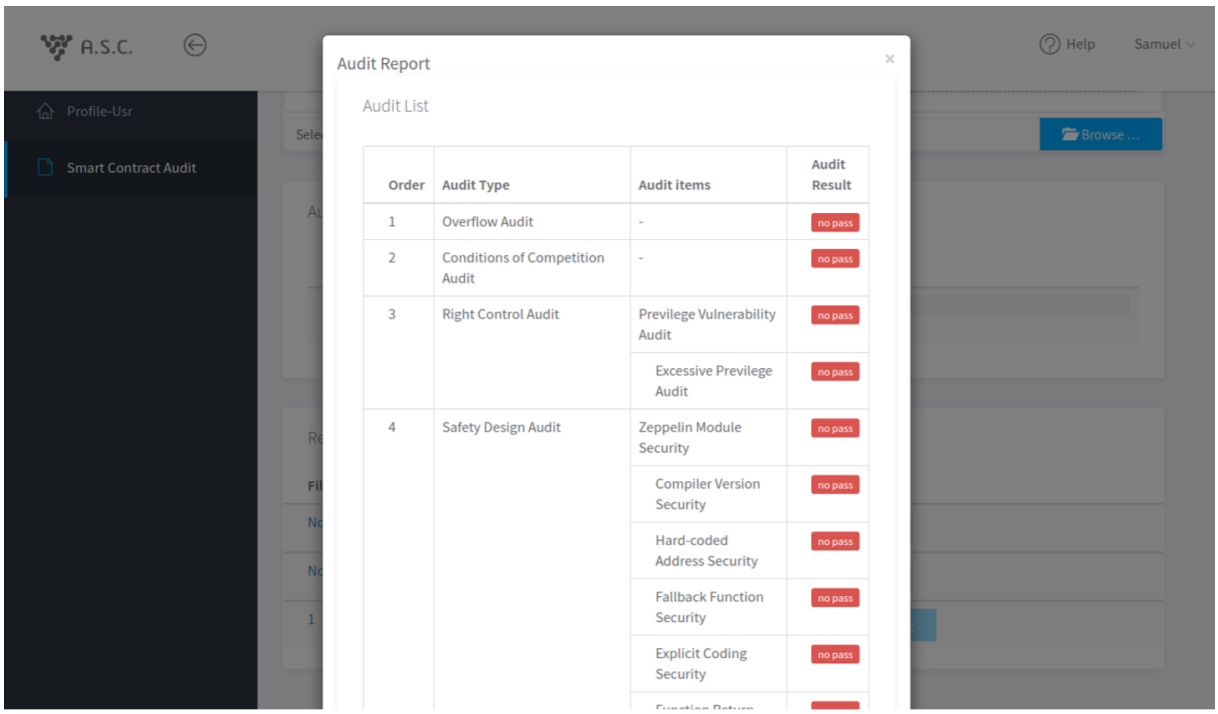
  （1） 智能合约提交



（2） 查看审计进展

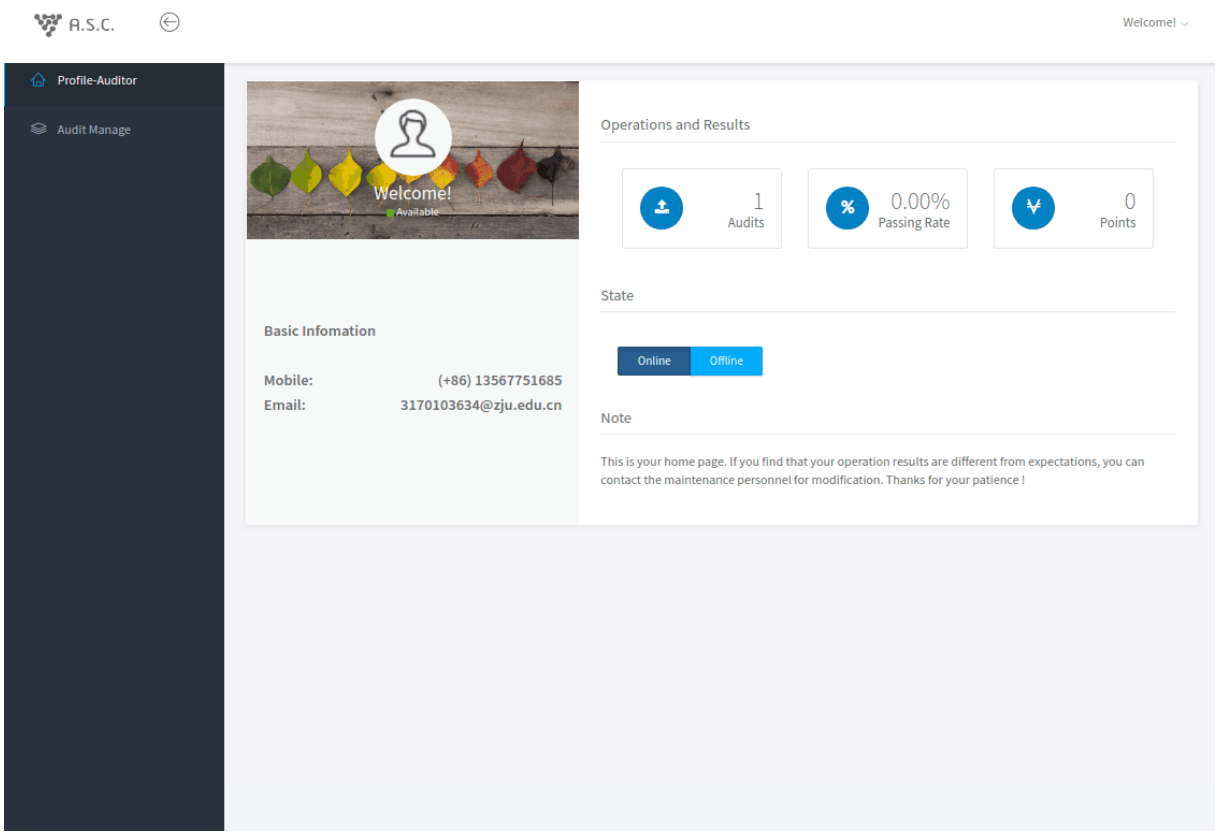


（3） 查看审计结果



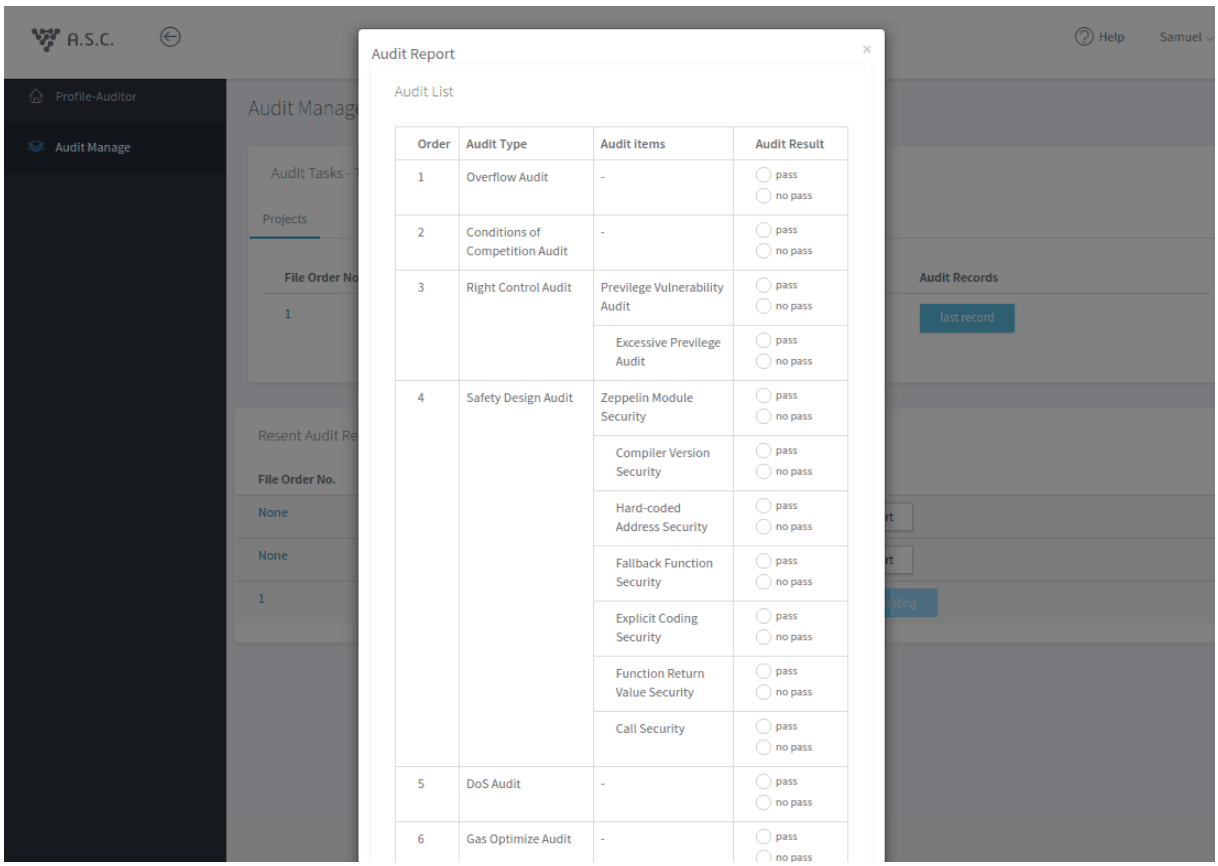
- 审计人员页面
  - 设计人员主页面(profileauditor.ejs)

（1） 个人信息简介

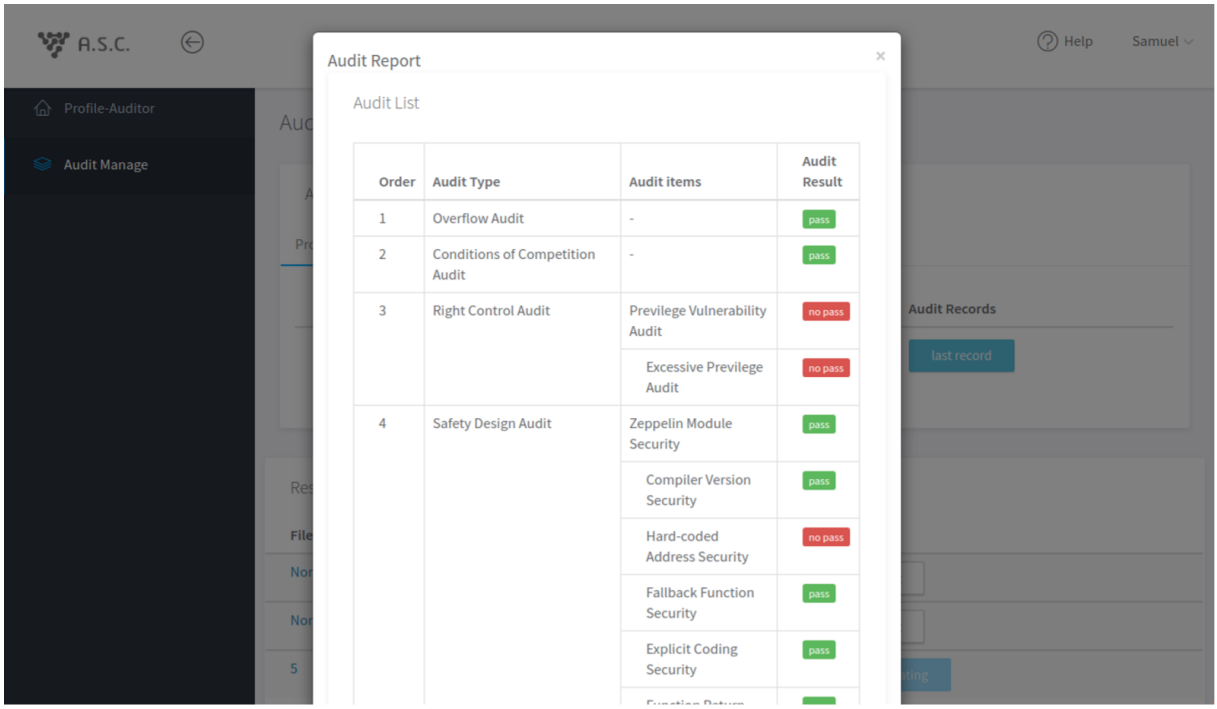


- 审计人员功能页面(auditmanage.ejs)

（1） 智能合约审计



（2） 审计结果查询



2.2 后端配置

后端使用js、Jquery、ajax、mysql和ipfs文件系统进行配置

- bootstrap-fileinput: 在终端输入 sudo npm install -g bootstrap-fileinput安装
- cookie-parser: 在终端输入 sudo npm install -g cookie-parser安装
- ethereumjs-account: 在终端输入 sudo npm install -g ethereumjs-account安装
- ethereumjs-tx: 在终端输入 sudo npm install -g ethereumjs-tx安装
- http-errors: 在终端输入 sudo npm install -g http-errors安装
- ipfs-http-client: 在终端输入 sudo npm install -g ipfs-http-client安装, 在安装之前，一定要把nodejs升级到稳定版本的最新版本
- morgan: 在终端输入 sudo npm install -g morgan安装
- mysql2: 在终端输入 sudo npm install -g mysql2安装

2.3 智能合约

智能合约撰写

智能合约上链元素：

用户提交智能合约时，将文件的哈希值上链

我们为智能合约的审查提供审查模板，在负责此智能合约的5个审查人员完成审查后，将审查结果的哈希值上传至链

将审查人员的地址和获得的积分的哈希值上传至链

在用户确认合约结果后，将用户为智能合约的审查而付出的以太币交易上传至链

```
pragma solidity >=0.4.17 <=0.6.0;
contract FormalVerification {

    address auditor;
    address user;
```

```
    address payable Beneficiary;
    uint timeOutLimit;
    mapping(address => uint) auditors;            // 审计人员的积分
    mapping(uint => string) reportHash;      // 审计人员的报告hash
    mapping(uint => uint) reportScore;            // 智能合约报告的分数(bytes可以是hash的字符串形式)

    constructor() public {
        auditor = msg.sender;        // 审计人员
        user = msg.sender;           // 用户
        timeOutLimit = 259200;       // 审计人员审核智能合约的时间为三天
        Beneficiary = 0x1EBD9225519E4549ad418c814951da2d84Ac6438;       //收款方的地址
    }


    function setReportScore(uint flag,uint reportFlag ,uint reportscore) public {
// 上链智能合约当前的分数
        require(flag == 1 && msg.sender == auditor,"Warning: You are not permitted!");
        reportScore[reportFlag] = reportscore;
    }

    function getReportScore(uint flag,uint reportFlag) public view returns(uint ret){
        require(flag == 1 && msg.sender == auditor,"Warning: You are not permitted!");
        ret = reportScore[reportFlag];
    }

    function setAuditScore(uint flag,uint currentScore) public {                 // 上链当前审计人员的新分数和当前审计报告的个数
        require(flag == 1 && msg.sender == auditor,"Warning: You are not permitted!");
        auditors[msg.sender] = currentScore;
    }

    function getAuditScore(uint flag) public view returns(uint ret){
        require(flag == 1 && msg.sender == auditor,"Warning: You are not permitted!");
        ret = auditors[msg.sender];
    }

    function setAuditorScore() public{
        auditors[0xf25bb2d9f47da0B0acA59f73d677E651Dd8aAf20] = 0;
        auditors[0x8f3cc6254AAE17815de13BbA0081C72476Db5abE] = 0;
        auditors[0x53C30f23c397eBE5811D5f0CE482938E0B7220ac] = 0;
        auditors[0x1242C0939a04A9d73C54907184C0D8f6A50D7c51] = 0;
        auditors[0x36344Fc242c2ac775B845262adA4B9F313f3329b] = 0;
    }

    function TimeOut(uint flag,uint period) public view returns(uint ret){            // 判断审计人员是否超时
        require(flag == 1 && msg.sender == auditor,"Warning: You are not permitted!");
// 审计人员
        if(period <= timeOutLimit){
            ret = 1;
        }
        else{
```

```
            ret = 2;
        }
    }

    function AfterAudit(uint flag,uint timeOut,uint count,string memory Hash) public {
     // 审计人员提交后获得新的积分
        require(flag == 1 && msg.sender == auditor,"Warning: You are not permitted!");
     // 审计人员
        reportHash[count] = Hash;                                                    // 存储新审计的
报告hash值
        if(timeOut == 2){                                                            // 判断是否超
时，并计算新的积分
            auditors[msg.sender] += 50;
        }
        else{
            auditors[msg.sender] += 100;
        }
    }

    function getAuditorScore(uint flag) public view returns(uint ret){
        require(flag == 1 && msg.sender == auditor,"Warning: You are not permitted!");
// 审计人员
        ret = auditors[msg.sender];
    }

    function Transfer(uint flag) public payable {           // 用户在提交智能合约审计时转账
        require(flag == 0 && msg.sender == user,"Warning: You are not permitted!");
        require(user.balance >= 10000000000000000 wei,"Warning: Your account balance is
insufficient.");
        Beneficiary.transfer(msg.value);
    }

    function getBalance() public view returns(uint ret){
        ret = msg.sender.balance;
    }
}
```

智能合约部署

首先启动ganache,运行在8545端口

配置`truffle-config.js`文件，使其在8545端口监听

将写好的sol文件使用`truffle compile`命令进行合约编译

然后使用`truffle migrate`命令进行合约部署

再使用truffle console终端简单测试后，合约部署成功

使用`solcjs命令`获取abi后，通过web3进行调用。

Chapter 3 : 使用说明

3.1 用例

3.1.1 用户用例

表3.1.1.1 用户登录

用例	用户登录
主要参与者	基本用户
目标	登录客户端
前提条件	计算机成功连接网络、安装好以太坊钱包插件
触发器	用户点击登录按钮
场景	1.用户得到钱包登录页面 2.用户注册或者使用助记词登录成功 3.得到用户地址 4.判断此用户是普通用户还是审计员 5.登录成功
异常	1.身份认证失败（用户使用审计人登录窗口）

表3.2.1.2 用户提交智能合约

用例	用户提交智能合约
主要参与者	基本用户
目标	提交智能合约
前提条件	用户已登录
触发器	用户点击文件上传或拖动文件进入窗口并上传
场景	1.用户已登录 2.用户进入"Smart Contract Audit"面板 3.用户点击"Browser"或拖动文件进入窗口 4.用户点击"Upload"（上传）或"Remove"（删除）并重新选择文件 5.上传成功
异常	1.文件过大 2.超出一个文件 3.正在审计的报告数量超出限制，无法上传

表3.2.1.3 用户查询审计结果

用例	用户查询审计结果
主要参与者	基本用户
目标	查询审计结果
前提条件	用户已登录、审计结果已产生
触发器	用户点击查询按钮
场景	1.用户已登录 2.用户进入"Smart Contract Audit"面板 3.用户点击查询选项"Check Report" 4.系统显示智能合约审计报告结果
异常	1.审计结果未产生

3.2.2 审计人员用例

表3.2.2.1 审计人员登录

用例	审计人员登录
主要参与者	审计人员
目标	登录客户端
前提条件	计算机成功连接网络、安装好以太坊钱包插件
触发器	审计人员点击登录按钮
场景	1.审计人员得到钱包登录页面 2.审计人员注册或者使用助记词登录成功 3.得到审计人员地址 4.判断此审计人员是普通用户还是审计员 5.登录成功
异常	2.身份认证失败（审计人使用用户登录窗口）

表3.2.2.2 审计合约

用例	审计合约
主要参与者	审计人员
目标	审计合约
前提条件	审计人员已登录、已有审计任务派发
触发器	审计人员点击“Audit”按钮
场景	1.审计人员已登录 2.进入"Audit Manage"面板 3.审计人员填入审计结果 4.提交审计结果
异常	1.审计结果未填写完整 2.超出审计时间

表3.2.2.3 查看已提交的审计报告

用例	查看已提交的审计报告
主要参与者	审计人员
目标	显示审计人员自己已经完成的审计报告
前提条件	审计人员收到用户提交的合约并已审计提交
触发器	审计人员点击查看按钮
场景	1.审计人员已登录 2.进入"Audit Manage"面板 3.审计人员点击查看 4.在当前页面上显示已提交的审计报告 5.查看完毕后关闭
异常	1.点击查看无反应 2.在当前页面可以显示弹框，但是没有出现审计结果 3.在当前页面可以显示出审计结果，但是存在错误

表3.2.2.4 设置在线状态

用例	设置在线状态
主要参与者	审计人员
目标	改变审计人员的状态：在线或者离线
前提条件	审计人员登录成功
触发器	审计人员点击状态改变按钮
场景	1.审计人员已登录 2.进入"Audit Manage"面板 3.审计人员点击状态改变按钮 4.online和offline两个按钮的样式发生改变
异常	1.点击查看无反应 2.点击按钮没有效果（按钮样式不发生改变） 3.点击按钮无法改变状态（按钮样式改变但是状态不改变）

表3.2.2.5 查看历史审计记录

用例	查看历史审计记录
主要参与者	审计人员
目标	显示当前审计人员的历史审计报告
前提条件	审计人员登录成功
触发器	审计人员点击Check Report按钮
场景	1.审计人员已登录 2.进入"Audit Manage"面板 3.审计人员点击Check Report按钮 4.当前页面显示出某一历史审计报告
异常	1.点击按钮无反应 2.在当前页面可以显示弹框，但是没有出现审计结果 3.在当前页面可以显示出审计结果，但是存在错误

3.2 求助查询

表3.2.1 求助查询

用例	求助查询
主要参与者	用户/审计人员
目标	求助查询
前提条件	用户/审计人员已登录
触发器	点击右上角"Help"按钮
场景	1.点击右上角"Help"按钮 2.获得帮助文档
异常	1.未登录

思考

此项目的idea来自导师，将只能合约的审查结果进行综合然后上链，其实本项目在一定程度上可以减少审计人员的作弊行为，但是如果有线下勾结的话，还是无法避免，需要公司内部采取其他措施保证审计效率。

本项目为SRTP的一部分内容，原本是想把此平台打造成一个智能合约的审计平台，融人工审核和自动化审计为一体，现在人工审计再加一些智能合约验证模板差不多完成了，自动化审计是我们SRTP的主要内容，用图神经网络和形式化验证进行自动化审计，但是现在还没有完成，所以，没有办法展示一个功能强大的平台。等到我们SRTP结束之后，会将此平台进行进一步的完善。