

|  |  |
| --- | --- |
| **题目** | 基于区块链的药品供应链 |
| **专业** | 软件工程区块链方向 |
| **班级** | B190601 |
| **姓名** | 吴远标 |
| **学号** | 2019000101185 |

|  |
| --- |
| 本人承诺本作业是本人独立撰写完成的。凡涉及他人的观点和材料，均做了注释，如有抄袭或其他学术不端行为，本人承担由学术不端行为所导致的相应责任和处分。 |

# 目录

[1 目录 1](#_Toc58969654)

[1. 概述 2](#_Toc58969655)

[1.1 背景 2](#_Toc58969656)

[1.2 风险、问题 2](#_Toc58969657)

[2 系统分析与设计 2](#_Toc58969658)

[2.1 系统总结结构 2](#_Toc58969659)

[2.2 开发环境的配置 2](#_Toc58969660)

[2.3 运行环境的配置 2](#_Toc58969661)

[2.4 系统功能描述 3](#_Toc58969662)

[2.4.1 功能汇总 3](#_Toc58969663)

[2.4.2 子系统A 3](#_Toc58969664)

[2.4.3 子系统B 3](#_Toc58969665)

[2.5 系统功能流程 4](#_Toc58969666)

[3 系统实现与测试 4](#_Toc58969667)

[3.1 系统采用的关键技术 4](#_Toc58969668)

[3.2 关键程序流程 5](#_Toc58969669)

[3.3 关键代码分析 5](#_Toc58969670)

[3.4 接口测试用例 5](#_Toc58969671)

[3.5 用户界面测试的检查表 6](#_Toc58969672)

[4 系统界面及操作 6](#_Toc58969673)

[4.1 系统访问与登录 6](#_Toc58969674)

[4.2 用户注册 7](#_Toc58969675)

[4.3 药品查询 8](#_Toc58969676)

[4.4 制药厂信息查询 8](#_Toc58969677)

# 概述

【简单描述】医药溯源是指对药品的生产加工、物流运输、零售 使用等过程进行记录追踪，一种基于区块链的医药防伪溯源系统。利用区块链技术记录各类药品、疫苗在生产、流通、使用整个流程的所有环节可追溯，实现全程可视化监管，让各部门可通过区块链获得产品在每一个流通环节的信息。当检测到药品或药品的特定方的负面或意外的副作用时，召回程序提供了销毁药品的运输。非常重要的是，要防止回收药品的转售，必须对其进行跟踪，并在最短的时间内向生产商/进口商提供退货流程

## 背景

目前正在使用的系统目前常用于药品防伪的有以下三种技术-1. 无线射频智能（RFID)技术，它从上世纪40年代由雷达技术转化而来，目前无法大范 围应用，主要原因是生产成本过高，其标签的价钱是一维条形码标签的10倍以上。2. 二维码防伪技术，标签表现为图形，消费者从表面无法以肉眼了解任何信息。其识读 设备的成本相比一维条形码识读设备高出了 3倍左右，其应用受到一定的限制。3. 电码与一维条形码相结合的防伪技术，为每个产品赋予一个独一无二的数字串， 一般 在16位到24位，消费者可拨打电话、发送短信和上网进行査询。

重新定义问题

综上所述，电码防伪查询的及时性最高， 一维条形码标签和识读成本最实用。目前电 码与一维条形码结合的技术除广泛用于物流外，还被防伪公司普遍采用，其条形码数字将 电码防伪融入其中，实现数字物流与产品防伪双重功效。国家药品电子监管也采用了该技术。但是， 一维条码加电码防伪技术仍存在如下缺点：1.查询收费抑制了査询需求。无论是手机上网査询的上网费，还是短信查询的双向收费， 抑或拨打信息台电话、市话、长途电话的电话费，其査询收费均抑制了査询需求和应用。2. 査询不方便。销售终端绝大部分没有提供电话和可上网的电脑供査询；手机短信查询 等候回复时间长：即使部分消费者采用手机上网査询，也会因网络速度慢，消费者嫌麻烦从 而不能或较难实现在购药的同时进行防伪实时认证。3. 查询输入防伪条码信息极易产生错误引起系统误判。消费者无论是拨打电话、短信， 抑或通过上网査询，都必须在固网电话、移动电话或电脑键盘上，键入防伪査询中心的电话 号码和一维条码中一长串的阿拉伯数字，很难保证输入不出现错误，并且还要耗费时间等候 防伪中心（或声讯台）话务员，将20位码输入电脑（此环节也有可能听记和输入错误）才能 得到查询结果。操作层面致命的弱点使查询输入极易产生错误而无法达到预期的防伪效果。4. 未查询的条形码标签容易被利用制假。在所有采用一维条形码防伪技术的产品中，査 询率最高的是烟酒类商品，査询率也仅为1%，即100件商品中，99件即便是假货也将根本 不会被发现，同时99%没被査询过的标签中，若有部分是在防伪数据库中已激活入网的条形 码被制假者贴在假货上，则防伪系统根本无法识别它是假货。显然， 一维条形码加电码防伪技术无法突破在消费者防伪査询时不方便、易出错和必须 付费（800、 400电话，企业需支付通信成本）所导致的防伪查询率极低，以及未查询的条码 标签被利用制假后，其技术无法辨别真伪的致命瓶颈，所有防伪技术都无法单凭防伪标签和 目前的査询方式做到真正防伪。

推荐一个方案并说明理由

那么，是否有一种技术可以帮我们降低药品防伪的成本呢？当然有，区块链技术。

区块链具有去中心化、分布式存储、数据的时间序列性且不可篡改和否认、智能合约的自动执行、安全和隐私保护等特点。

利用区块链去中心化进行成本管理，有效降低药品防伪的成本。一是区块链让药品管理更加简洁。通过区块链，可以让技术取代许多复杂的工作，节省大量的时间和费用等。去中心化减少了人员，简化了步骤，可以降低药品的查询成本和防伪成本。

信息上链

基于区块链技术的药品防伪主要有药品制造商和医生或患者两方：

结合区块链技术，药品制造商将药品信息上链可以省掉中介职能，这些职能包括数据核对、提供收据和采购订单等。通过区块链的IPFS储存机制将药品信息储存在链上，便于医生或患者查验，使医生或患者完成交易和其他管理任务所花费的时间大大减少。缩短交易时间，让医生或患者花更多时间在其他重要业务职能上。

## 风险、问题

【项目将面临的风险和问题，在务实性和项目潜力乐观之间达成平衡】

信息安全

在传统的制药、买药和用药环节中患者将自己的个人病历传递至医院数据库，而不法分子通过收集病历、筛选病历、出售病历获取盈利，这也是将个人信息泄露的源头。医院对患者的信息保护不到位，造成患者信息泄露，并且被用于交易，轻则会使得患者遭受垃圾信息和垃圾电话的骚扰，对其生活产生不必要的麻烦；重则发生互联网诈骗，使得患者财产损失或者威胁到其生命安全。

结合区块链技术将使患者的信息安全问题得到缓解，患者将病历信息储存在IPFS中，IPFS的存储机制是类似于共享模式的应用，它的去中心化的特点让看病时的不法分子没有了可控之机，让患者病历信息的泄露问题从源头得到解决，从而达到保护患者病历信息安全的效果。同时IPFS机制是分布式的存储，患者的隐私安全也得以保证。

# 系统分析与设计

## 系统总结结构

【*将系统分解为若干子系统，绘制物理图和逻辑图，说明各子系统的主要功能。*】

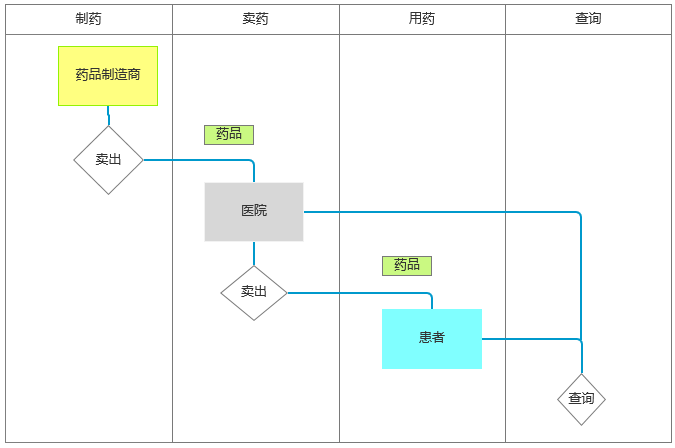
用户注册 药品制造商、医院和患者注册

药品生产和加工 药品制造商制造出药品

药品转出 制造商把药卖给医院，医院买了药，再卖给患者，

药品使用（使用后药品不再卖出）

药品查询 查询药品的各种信息





## 开发环境的配置

|  |  |  |
| --- | --- | --- |
| 类别 | 标准配置 | 最低配置 |
| 计算机硬件 | CPU Inter(R) Core(TM) i5-4460 CPU@ 3.20GHz 硬盘 13GB以上（空闲）内存 8GB 操作系统  Windows 7 64bit及以上 显卡 2048MB显存或以上 | CPU Inter(R) Core(TM) i3-4160 CPU@ 3.60GHz 硬盘 13GB以上（空闲）内存4GB 操作系统Windows 7 64bit及以上显卡 1024MB显存 |
| 软件 | RemixIDE,Geth,Go,EthereumWallet，谷歌浏览器Google Chrome，Metamask | VisualStudioCode，Geth,Go,EthereumWallet，MicrosoftEdge，Metamask |
| 其它 | 无 | npm,truffle框架 |

## 运行环境的配置

|  |  |  |
| --- | --- | --- |
| 类别 | 标准配置 | 最低配置 |
| 计算机硬件 | CPU Inter(R) Core(TM) i5-4460 CPU@ 3.20GHz 硬盘 13GB以上（空闲）内存 8GB 操作系统  Windows 7 64bit及以上 显卡 2048MB显存或以上 | CPU Inter(R) Core(TM) i3-4160 CPU@ 3.60GHz 硬盘 13GB以上（空闲）内存4GB 操作系统Windows 7 64bit及以上显卡 1024MB显存 |
| 软件 | Geth,Go,EthereumWallet，谷歌浏览器Google Chrome，Metamask | VisualStudioCode，Geth,Go,EthereumWallet，MicrosoftEdge，Metamask |
| 其它 | 无 | npm,truffle框架 |

## 系统功能描述

### 功能汇总

|  |  |
| --- | --- |
| **子系统A 药品管理子系统** | |
| 模块名称 | 功能简述 |
| *药品生产* | *药品制造商用制造药品* |
| *药品出售* | *药品制造商给药品制定价格和改变药品的状态为待售* |
| *…药品购买* | *确认药品的状态为待售，确认当前账户所付金额是否足够，医院买药品制造商制造的药，患者买医院开的药* |
| **子系统B 用户管理子系统** | |
| 模块名称 | 功能简述 |
| *用户注册* | *药品制造商注册，运输商注册，医院注册，患者注册* |
| *用户注销* | *药品制造商注销，运输商注销，医院注销，患者注销* |
| *…* |  |

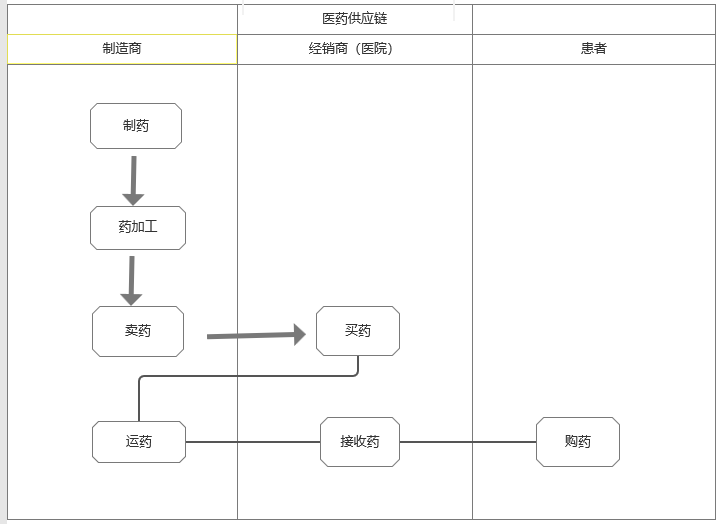
### 药品管理子系统

|  |  |
| --- | --- |
| 模块名称 | *药品生产* |
| 功能描述 | *制造药品，并给药品结构里的字段赋值，包括通用商品代码，拥有者ID，制造商ID,制造商工厂名称，制造商工厂信息，制造商工厂地理位置，药品ID，药品信息* |
| 接口与属性 | 1. ***提示：****用专业的设计（开发）工具来设计本模块的接口与属性，说明函数功能、输入参数、输出参数、返回值等。此处粘贴即可。function makefood(uint \_upc, address \_originManufacturerID, string \_originFactoryName, string \_originFactoryInformation, string  \_originFactoryLatitude, string  \_originFactoryLongitude, string  \_foodNotes) public* 3. *onlyManufacturer* 4. *{* 5. *// Add the new food as part of foods* 6. *food memory temp\_food = food({* 7. *sku:sku + 1,* 8. *upc:\_upc,* 9. *ownerID:\_originManufacturerID,* 10. *originManufacturerID:\_originManufacturerID,* 11. *originFactoryName:\_originFactoryName,* 12. *originFactoryInformation:\_originFactoryInformation,* 13. *originFactoryLatitude:\_originFactoryLatitude,* 14. *originFactoryLongitude:\_originFactoryLongitude,* 15. *foodID:sku+\_upc,* 16. *foodNotes:\_foodNotes,* 17. *foodState:State.Made,* 18. *foodPrice:0,* 19. *distributorID:0,* 20. *carrierID:0,* 21. *customerID:0* 22. *});* 23. *foods[\_upc] = temp\_food;* 24. *foods[\_upc].foodState = State.Made;* |
| 补充说明 | *输入参数有：通用商品代码，拥有者ID，制造商ID,制造商工厂名称，制造商工厂信息，制造商工厂地理经度，纬度，药品ID，药品信息*  *没有输出参数，*  *没有返回值，* |

### 用户管理子系统

|  |  |
| --- | --- |
| 模块名称 | *用户注册* |
| 功能描述 | *这里以药品制造商注册为例，添加一个药品制造商* |
| 接口与属性 | 1. ***提示：****用专业的设计（开发）工具来设计本模块的接口与属性，说明函数功能、输入参数、输出参数、返回值等。此处粘贴即可。* 2. ***// Define a function 'addManufacturer' that adds this role定义添加制造商的规则的功能*** 3. ***function addManufacturer(address account) public onlyManufacturer {*** 4. ***\_addManufacturer(account);*** 5. ***}***   **// Define a modifier that checks to see if msg.sender has the appropriate role定义一个修饰符来检查，请求发送者是否符合条件isManufacturer(msg.sender)=true**  **modifier onlyManufacturer() {**  **require(isManufacturer(msg.sender));**  **\_;**  **}**  **// Define a function 'isManufacturer' to check this role这个function有返回值，是个布尔值，上面的修饰符调用了这个function**  **function isManufacturer(address account) public view returns (bool) {**  **return Roles.has(manufacturers, account);**  **}** |
| 补充说明 | *输入参数是 metamask 上的账号地址，*  *有onlyManufacturer修饰符修饰，对该function的call调用作了规定，*  *没有输出参数，*  *没有返回值，* |

## 系统功能流程



# 系统实现与测试

## 系统采用的关键技术

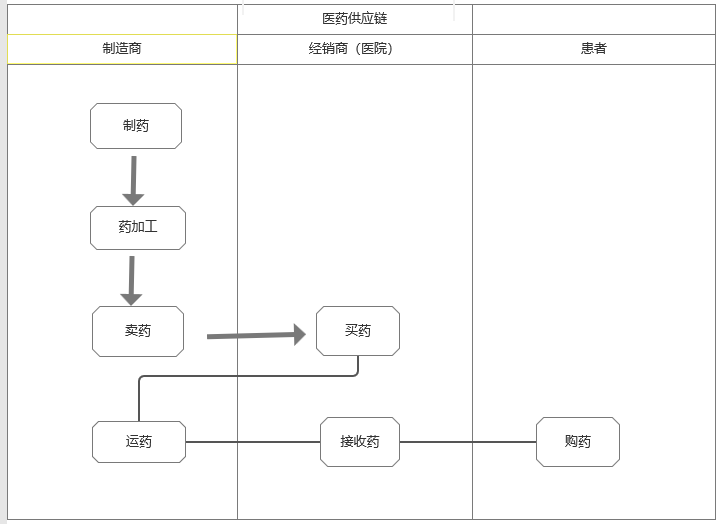
[以太坊，智能合约，web3js]

以太坊是去中心化应用程序的全球开源平台。在以太坊上，您可以编写控制数字价值的代码，完全按照编程运行，并可在世界任何地方访问。以太坊对所有人开放。

你只需要一个钱包就可以参加。

以太坊对每个人都开放获取数字货币和数据友好型服务——无论您的背景或位置如何。它是加密货币以太坊(ETH)背后的一种社区构建技术，今天您可以使用数千个应用程序。以太坊是一种可以让你以少量费用向任何人发送加密货币的技术。它还为每个人都可以使用、没有人可以关闭的应用程序提供动力。

## 关键程序流程



## 关键代码分析

|  |  |
| --- | --- |
| **程序名称** | **// Provides basic authorization control 提供基本的授权控制**  **contract Ownable {**  **address private origOwner;**  **// Define an Event 定义一个事件**  **event TransferOwnership(address indexed oldOwner, address indexed newOwner);**  **/// Assign the contract to an owner 把合约分配给拥有者**  **constructor () internal {**  **origOwner = msg.sender;**  **emit TransferOwnership(address(0), origOwner);**  **}**  **/// Look up the address of the owner 查阅拥有者的地址**  **function owner() public view returns (address) {**  **return origOwner;**  **}**  **/// Define a function modifier 'onlyOwner' 定义一个修饰符**  **modifier onlyOwner() {**  **require(isOwner());**  **\_;**  **}**  **/// Check if the calling address is the owner of the contract**  **function isOwner() public view returns (bool) {**  **return msg.sender == origOwner;**  **}**  **/// Define a function to renounce ownerhip 定义一个放弃拥有权的功能**  **function renounceOwnership() public onlyOwner {**  **emit TransferOwnership(origOwner, address(0));**  **origOwner = address(0);**  **}**  **/// Define a public function to transfer ownership 定义一个用于转让拥有权的公开的功能**  **function transferOwnership(address newOwner) public onlyOwner {**  **\_transferOwnership(newOwner);**  **}**  **/// Define an internal function to transfer ownership 定义一个用于转让拥有权的内部的功能**  **function \_transferOwnership(address newOwner) internal {**  **require(newOwner != address(0));**  **emit TransferOwnership(origOwner, newOwner);**  **origOwner = newOwner;**  **}**  **}** |
| **功能描述** | **提供基本的授权控制,拥有权的授权，查阅，放弃和转移** |
| *记录编程技术、问题与对策、心得体会等等*  在最近的一个版本(2018年3月8日v0.4.21)中，引入了' emit '关键字来emit事件。这将有助于区分功能和事件，这是导致以太坊硬分叉和以太坊经典诞生的DAO(Decentralized Autonomous Organization)黑客的原因之一。  根据Solidity发布说明:一般:支持并推荐使用emit EventName();显式地调用事件。  为了使事件在常规函数调用中脱颖而出，现在应该使用emit EventName()而不是EventName()来“调用”事件。确保您的编译器版本是0.4.21或更高版本，对于较低版本，它将抛出编译错误。 | |

## 接口测试用例

|  |  |  |  |
| --- | --- | --- | --- |
| 接口A的函数原型 | // Define a function 'sellMedicine' that allows a manufacturer to mark an medicine 'ForSale'定义销售药品的function来允许制造商将一药品标记为待售    function sellMedicine(uint \_upc, uint \_price) public    // Call modifier to check if upc has passed previous supply chain stage    packed(\_upc)    // Call modifier to verify caller of this function    onlyManufacturer    {      // Update the appropriate fields      medicines[\_upc].medicineState = State.ForSale;      medicines[\_upc].medicinePrice = \_price;      // Emit the appropriate event      emit ForSale(\_upc);    } | | |
| 输入/动作 | | 期望的输出/相应 | 实际情况 |
| 典型值… (H10940007,100) | | 成功将该药品的状态改为待售 | 成功将该药品的状态改为待售 |
| 边界值… (H10940007,1) | | 成功将该药品的状态改为待售 | 成功将该药品的状态改为待售 |
| 异常值… (H10940007,0) | | 操作失败，提示：出现错误 | 操作失败，提示：出现错误 |
| 接口B的函数原型 | // Define a function 'addManufacturer' that adds this role    function addManufacturer(address account) public onlyManufacturer {      \_addManufacturer(account);    } | | |
| 输入/动作 | | 期望的输出/相应 | 实际情况 |
| 典型值…0xEA03e508Ae066bA6893a9c8bF8B048EC85860d73 | | 提示：添加制造商成功 | 提示：添加制造商成功 |
| 边界值…无 | | 无 | 无 |
| 异常值… "123jfksladjflasdjfl123123" | | 提示：地址输入错误 | 提示：地址输入错误 |
| … | |  |  |

## 用户界面测试的检查表

|  |  |
| --- | --- |
| 检查项 | 测试人员的类别及其评价 |
| 窗口切换、移动、改变大小时正常吗？ | 不正常，反应迟钝 |
| 各种界面元素的文字正确吗？（如标题、提示等） | 正确 |
| 各种界面元素的状态正确吗？（如有效、无效、选中等状态） | 有效，可以选中 |
| 各种界面元素支持键盘操作吗？ | 支持键盘操作 |
| 各种界面元素支持鼠标操作吗？ | 支持鼠标操作 |
| 对话框中的缺省焦点正确吗？ | 焦点正确 |
| 数据项能正确回显吗？ | 不能正确回显 |
| 对于常用的功能，用户能否不必阅读手册就能使用？ | 可以，比如注册，登陆功能 |
| 执行有风险的操作时，有“确认”、“放弃”等提示吗？ | 没有 |
| 操作顺序合理吗？ | 当然合理 |
| 有联机帮助吗？ | 没有 |
| 各种界面元素的布局合理吗？美观吗？ | 布局合理，美观 |
| 各种界面元素的颜色协调吗？ | 颜色协调，白色和淡蓝色 |
| 各种界面元素的形状美观吗？ | 形状美观，是长宽成比例的矩形 |
| 字体美观吗？ | 字体美观 |
| 图标直观吗？ | 没有图标 |
| 变量的数据类型有错误吗？ | 没有 |
| …存在误差累积吗？ | 不存在 |

# 系统界面及操作

## 系统访问与登录

* **访问方式一：通过网址直接访问**

在地址栏键入医药溯源平台（下简称医药溯源系统）访问地址http://127.0.0.1并回车，浏览器中会出现系统登录页面。



## 用户注册

系统用户登录前，首先根据自己要注册的用户类型在用户注册界面的类型选择的下拉框选择对应的用户类型注册，并填写相应的个人信息。

用户注册步骤如下：

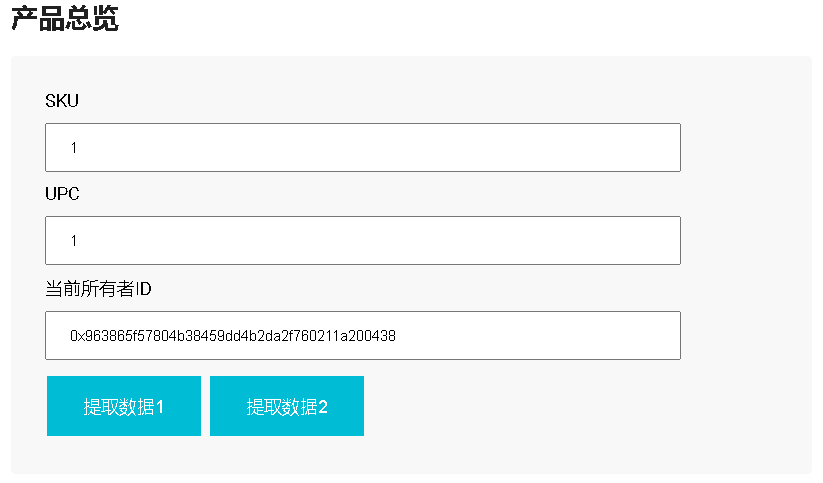
* 打开用户注册对话框

打开系统登录页面，点击“注册”按钮，弹出“用户注册”对话框。

* 提交注册流程

输入用户信息（包括metamask账号），并选择相应用户类型（药品制造商，药品经销商，药品运输商，患者）后，点击“提交”后，提示用户注册流程启动。

## 药品查询



## 制药厂信息查询

