2023年全国职业院校技能大赛

高职组

“区块链技术应用”

赛项赛卷（4卷）

任

务

书

参赛队编号：

**背景描述**

随着消费需求的不断变化，消费者对于食品安全的关注度越来越高，不仅要求生鲜肉品的肥瘦比例、脂肪含量等，更包括产品生产和物流管理过程。传统供应链（如农场、加工厂、配送商、零售商等）条件下这些数据的保存基本依靠纸张记录，具有一定的随意性，消费者无法识别这些数据的真伪。因此，供应链管理正面临着效率和安全透明的双重挑战，迫切需要进行有效变革，促进食品供应链管理更加高效透明和安全。

**手机屏幕截图

中度可信度描述已自动生成**

为了解决以上问题，使用新的高效率、高透明和真实性的新型供应链来打破传统食品供应链的研究迫在眉睫。使用区块链技术不仅能够开创食品溯源的新时代、为改善食品安全状况打下基础，还能够帮助企业提高食品追溯体系的运转效率，从而减少浪费，降低成本，促进可持续发展。如通过使用加密技术，写入区块链的数据在不被监测的情况下将无法被更改，提高了数据可信度，有效遏制食品造假问题。区块链与供应链相结合的一个最为重要的特点是使得商品信息在区块链上可追溯。同时，所有参与方的数据均上网可查，增加供应链各方的交互，提高了办事效率。用户在购买商品时，可通过扫描生成的二维码获取食品从源头到培育到运输以及最后摆上货架的所有相关信息。这种透明供应链系统一方面可以保障用户自身利益，也可以提高商店利润率，实现双赢。

## 模块一：区块链产品方案设计及系统运维（35分）

选手完成本模块的任务后，将任务中设计结果、运行代码、运行结果等截图粘贴至客户端桌面【区块链技术应用赛\重命名为工位号\模块一提交结果.docx】中对应的任务序号下。

### 任务1-1：区块链产品需求分析与方案设计

本环节需要依据项目背景完成需求分析与方案设计，具体要求如下:

1. 依据项目背景中食品供应链溯源系统的业务架构图，进行业务分析，使用工具编制所有的用户角色、用户用例以及相应的业务系统功能图；
2. 设计区块链系统的架构，画出系统架构图，指明需要的节点个数、名称等信息；结合功能划分，列出表格说明各个主要模块功能；
3. 根据需求分析、承诺集设计，完成区块链应用系统业务流程图；
4. 整合所有内容，设计接口编写并形成概要设计说明书“01概要设计说明书.doc”，保存至【区块链技术应用赛\重命名为工位号\】目录下。

### 任务1-2：区块链系统部署与运维

通过给定区块链项目需求，进行区块链系统部署，包括系统部署、控制台部署等。通过监控工具完成对网络、节点服务的监控。最终利用业务需求规范，完成系统日志、网络参数、节点服务等系统结构的维护。

1. 部署区块链服务器，配置管理平台参数及访问端口；
2. 部署项目节点，获取管理平台与节点服务的加密通信；
3. 部署应用程序接口服务，应用工具完成接口部署；
4. 部署区块链系统，对接应用服务器和数据库。

**子任务1-2-1：搭建区块链系统**

在本机搭建一条分布式存储的4节点区块链系统，其网络端口要求如下：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **机构名称** | **节点数** | **P2P端口** | **channel端口** | **rpc端口** |
| 机构A | 1 | 30100 | 20100 | 8010 |
| 机构B | 1 | 30200 | 20200 | 8020 |
| 机构C | 2 | 30300 | 20300 | 8030 |

搭建完区块链系统后，需检查每个节点的连接数和运行状态。

**子任务1-2-2：控制台配置与使用**

为区块链系统配置控制台，使用控制台与区块链系统进行简单交互（配置控制台相关工具和软件在 “/root/Desktop/src” 目录下），交互步骤如下：

（1）查询节点版本信息；

（2）部署HelloWorld合约；

（3）查看交回执，解释每个字段含义；

（4）查询部署合约的区块信息。

**子任务1-2-3：区块链系统权限分配**

为该区块链系统分配权限，大赛平台提供了机构 A、机构 B 和机构 C 管理员账号的私钥文件，现有任务如下：

（1）需要将这三个账号分别设置为委员账号；

（2）设置机构 A 账号的投票权重为2；

（3）新增运维账号D，并部署HelloWorld合约。

### 任务1-3：区块链系统测试

设计对区块链系统的测试流程；结合实际业务需求，调用部署的智能合约中进行系统测试、性能测试等；根据业务需求，分析并且修复给定智能合约中的安全漏洞。利用模拟业务和测试工具来完成对区块链系统服务数据的测试。

1. 部署节点前置服务，实现基于浏览器测试启动情况验证;
2. 将大赛平台中新建的material、logistics、user账户以p12的形式导出，并导入大赛平台，验证账户地址一致，并做接口测试;
3. 分析最近一分钟的节点日志打印,收集日志关键错误打印信息,分析节点的共识消息处理、出块、交易数量等信息,判断节点的健康度。

## 模块二：智能合约开发与测试（30分）

选手完成本模块的任务后，将任务中设计结果、运行代码、运行结果等截图粘贴至客户端桌面【区块链技术应用赛\重命名为工位号\模块二提交结果.docx】中对应的任务序号下。

### 任务2-1：智能合约设计

根据食品供应链溯源的功能需求，完成以下任务：

1.设计合约接口，画出各智能合约的UML图；

2.理解程序调用逻辑，画出各合约关系的时序图。

### 任务2-2：智能合约开发

使用Solidity编程语言开发智能合约，包括存储合约设计开发，接口合约设计开发，最终实现区块链溯源存证功能。针对Solidity智能合约进行代码检查和测试，验证函数执行结果是否与预期结果相符。

**子任务2-2-1：原材料合约接口编码**

1.编写原材料接口newMaterial，初始化原材料信息，返回合约地址，并实现原材料信息上链功能，合约部分代码如下：

|  |
| --- |
| contract Material {  struct Material{  address owner;  string name;  string id;  string memo;  uint createdAt;  bool exist;  }  mapping(string => Material) public materials;  // 可自行添加形参和返回值  function newMaterial (string memory \_name, string memory \_id,...) public {  // TODO  \*\*\*  }  } |

2.编写获取存原材料接口getMaterial，根据合约地址获取原材料信息，合约部分代码如下：

|  |
| --- |
| // 可自行添加形参和返回值  function getMaterial(string memory id) public view returns (...) {  // TODO  \*\*\*  } |

**子任务2-2-2：食品物流接口编码**

1.编写食品物流上链接口addLogistic，实现食品物流信息上链功能，合约部分代码如下：

|  |
| --- |
| pragma experimental ABIEncoderV2;  contract Logistics {  struct LogisticsData {  address cargo;  address orgin;  address destination;  string memo;  uint createdAt;  uint queryCount;  }  LogisticsData[] private \_logisticsData;  uint public recordCount;  uint public queryCount;  // 可自行添加形参和返回值  function addLogistics(address cargo, address orgin，...) public {  // TODO  \*\*\*  } |

2.编写获取食品物流信息的接口getLogistics，根据食品产品编号获取物流信息，合约部分代码如下：

|  |
| --- |
| // 可自行添加形参和返回值  function getLogistics(uint256 \_id) public view returns (string memory, \*\*\*) {  // TODO  \*\*\*  } |

### 任务2-3：智能合约测试

消除合约代码错误和警告，成功编译智能合约并部署，通过调用合约接口，模拟实际业务流程检查合约状态和数据，发现可能存在的缺陷和漏洞，编写高质量、可复用的测试用例，从而提升开发效率。

**子任务2-3-1：业务测试**

1.解决代码错误和警告，正确编译并部署合约，成功获取部署的合约地址和ABI。

2.调用食品和物流合约接口newMaterial、getMaterial、addLogistic和getLogistics，模拟完整验证业务流程。

将上述任务中的编写代码及运行截图保存至指定位置。

**子任务2-3-2：用例测试**

为食品原材料信息上链（newMateria）和获取食品信息（getMateria）编写测试用例，部分代码如下：

|  |
| --- |
| @Test  public void Materia() throws Exception {  // deploy contract  Materia materia = Materia.deploy();  Assert.assertNotNull(logistic);  // TODO  \*\*\*  } |

## 模块三：区块链应用系统开发（30分）

选手完成本模块的任务后，将任务中设计结果、运行代码、运行结果等截图粘贴至客户端桌面【区块链技术应用赛\重命名为工位号\模块三提交结果.docx】中对应的任务序号下。

### 任务3-1：区块链应用前端开发

**子任务3-1-1：商品溯源前端开发**

结合给定区块链业务需求和前端页面示例，完成前端首页页面，具有如下功能：

1.根据输入的食品编号或食品合约地址，查询食品的原材料信息和物流信息；

2.物流信息格式为【时间 日期 中转信息 备注信息】

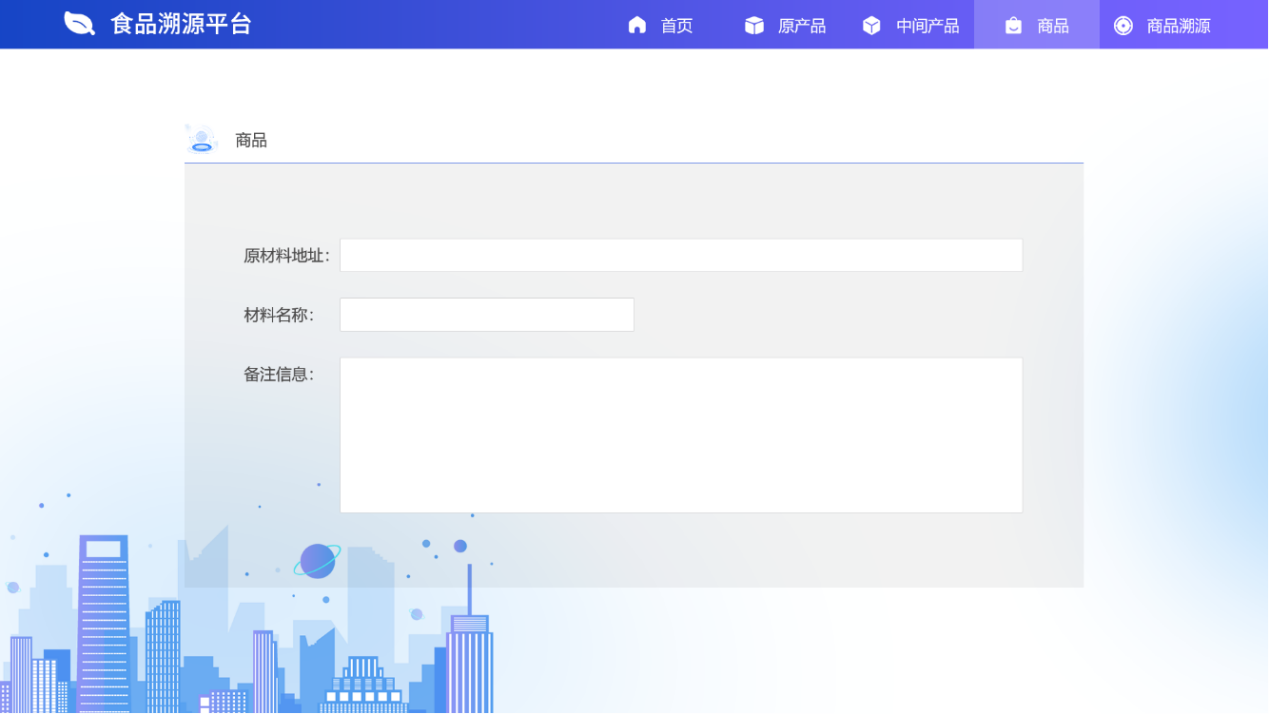


部分示例代码如下：

|  |
| --- |
| <template>  <div id="app">  <div class="navs">  <div class="navs\_left">\*\*\*溯源系统</div>  <div class="navs\_right">  <el-menu background-color="#67992a" text-color="#fff" :default-active="activeIndex" class="el-menu-demo" mode="horizontal" @select="handleSelect">  <el-menu-item index="1">原材料</el-menu-item>  <el-menu-item index="2">物流</el-menu-item>  <el-menu-item index="3">用户</el-menu-item>    </el-menu>  </div>  </div>  < />  </div>  </template>  //TODO  \*\*\* |

**子任务3-1-2：商品页面前端开发**

实现商品界面，具体展示内容有原材料地址、材料名称、备注信息



部分示例代码如下：

|  |
| --- |
| <template>  <div id="app">  <div class="navs">  <div class="navs\_left">\*\*\*溯源系统</div>  <div class="navs\_right">  <el-menu background-color="#67992a" text-color="#fff" :default-active="activeIndex" class="el-menu-demo" mode="horizontal" @select="handleSelect">  <el-menu-item index="1">原材料</el-menu-item>  <el-menu-item index="2">物流</el-menu-item>  <el-menu-item index="3">用户</el-menu-item>    </el-menu>  </div>  </div>  < />  </div>  </template>  //TODO  \*\*\* |

### 任务3-2：区块链应用后端开发

使用Java-SDK与区块链进行交互，将Solidity智能合约转译为可供Java调用的文件，实现区块链编程。

**子任务3-2-1：获取交易总量**

使用Java语言编写后端代码进行交互，获取区块链的交易总量，要求如下：

1.函数返回十进制的整数；

2.函数有调用异常处理，若合约交互失败需返回失败原因。

部分代码如下：

|  |
| --- |
| @GetMapping("/txSum")  public String getTotalTransactionCount() {  ApplicationContext context = new ClassPathXmlApplicationContext("\*\*");  // TODO  \*\*\*  } |

**子任务3-2-2：获取区块信息**

编写后端代码进行交互，获取区块链的最近10个区块的信息，包括时间戳、区块哈希、区块高度，要求如下：

1.函数返回最近十个区块的信息；

2.函数有调用异常处理，若合约交互失败需返回失败原因。

部分代码如下：

|  |
| --- |
| @GetMapping("/blockByNumber")  public ObjectNode getBlockByNumber() {  ObjectNode result = new ObjectMapper().createObjectNode();  try {  ApplicationContext context = new ClassPathXmlApplicationContext("config.xml");  //TODO  \*\*\*  }  } |

**子任务3-2-3：区块链sdk保持连接**

定义service用来保持与区块链的连接，减少与区块链的重复连接，加快访问速度，连接部分代码如下：

|  |
| --- |
| @Service  public class MyService {  private final Client client;  public MyService() {  }  //TODO  \*\*\*  } |

**子任务3-2-4：原材料智能合约交互**

开发食品溯源系统中原材料的接口，根据系统中原有的代码补充produceMaterial接口，实现食品原材料的生产与查询，代码示例如下：

|  |
| --- |
| @PostMapping("/produceMaterial")  public ObjectNode produceMaterial(@RequestBody ObjectNode params) {  ObjectNode result = new ObjectMapper().createObjectNode();  //检查参数  if (params == null || params.get("name") == null  || params.get("id") == null || params.get("memo") == null) {  result.put("status", "error");  result.put("message", "请求参数错误");  return result;  }  //TODO  } |

**子任务3-2-5：供应链溯源合约交互**

开发供应链合约中获取食品所有物流信息的后端接口，根据系统中原有的代码补充getLogistics接口，实现根据食品的id查询食品物流信息，示例代码如下：

|  |
| --- |
| @GetMapping("/getLogistics")  public ObjectNode getLogistics(String cargo) {  ObjectNode result = new ObjectMapper().createObjectNode();  try {  ApplicationContext context = new ClassPathXmlApplicationContext("config.xml");  // TODO  \*\*\*  } |