项目设计说明

课程名称	区块链原理与技术	任课老师	黄华威
年级	2018级	专业	软件工程
组长	张欢	学号	18342126
组员	杨玲	学号	18342115
组员	李小曼	学号	18339008

一、项目背景

传统供应链金融：

某车企（宝马）因为其造车技术特别牛，消费者口碑好，所以其在同行业中占据绝对优势地位。因此，在金融机构（银行）对该车企的信用评级将很高，认为他有很大的风险承担的能力。在某次交易中，该车企从轮胎公司购买了一批轮胎，但由于资金暂时短缺向轮胎公司签订了1000万的应收账款单据，承诺1年后归还轮胎公司1000万。这个过程可以拉上金融机构例如银行来对这笔交易作见证，确认这笔交易的真实性。在接下里的几个月里，轮胎公司因为资金短缺需要融资，这个时候它可以凭借跟某车企签订的应收账款单据向金融结构借款，金融机构认可该车企（核心企业）的还款能力，因此愿意借款给轮胎公司。但是，这样的信任关系并不会往下游传递。在某个交易中，轮胎公司从轮毂公司购买了一批轮毂，但由于租金暂时短缺向轮胎公司签订了500万的应收账款单据，承诺1年后归还轮胎公司500万。当轮毂公司想利用这个应收账款单据向金融机构借款融资的时候，金融机构因为不认可轮胎公司的还款能力，需要对轮胎公司进行详细的信用分析以评估其还款能力同时验证应收账款单据的真实性，才能决定是否借款给轮毂公司。这个过程将增加很多经济成本，而这个问题主要是由于该车企的信用无法在整个供应链中传递以及交易信息不透明化所导致的。

二、解决方法

区块链+供应链金融：

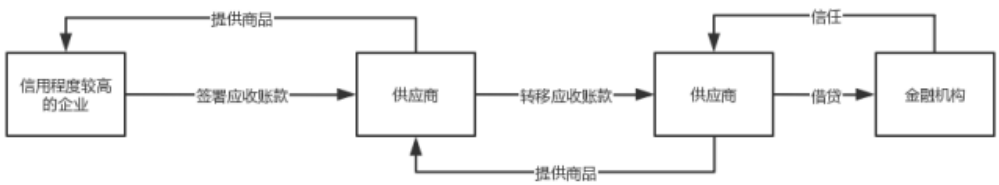
将供应链上的每一笔交易和应收账款单据上链，同时引入第三方可信机构来确认这些信息的交易，例如银行，物流公司等，确保交易和单据的真实性。同时，支持应收账款的转让，融资，清算等，让核心企业的信用可以传递到供应链的下游企业，减小中小企业的融资难度。

具体来说：智能合约是要由核心企业来部署的，只有核心企业才可以开具应收账款、而一级供应商可以根据自己的情况去拆分核心企业所开具的应收账款，也可以拿去银行融资，由于核心企业开具应收账款这一个操作是由智能合约调用的，因此记录在了链上，二级供应商或银行都可以在链上查看这份应收账款是否真实有效，确保了可信性。二级供应商到三级供应商或二级供应商去融资也是同样的道理。核心企业到期还款的时候，之前拆分或融资都会记录在链

上，所以核心企业只需要查看链上的所有应收账款的收款人地址和金额即可完成还款，而无需再花时间去验证这些子应收账款是否可信。

三、具体方案设计

（一）方案架构图



整个流程都得上链，在区块链上记录。

（二）存储设计

Receipt

在智能合约中，最主要的是Receipt这个结构体，这个结构体保存了应收账款的欠款方、收款方，金额、还款日期等重要信息，在拆分账款、融资、还是结算，都得用到。

```
struct Receipt{
        uint id;                //收据编号——主键
        address from;           //欠款人
        address to;             //收款人
        uint amount;            //应收款金额
        uint startDate;         //开票日期
        uint endDate;           //截止日期
        bool isusedLoan;        //是否被用作于向金融机构贷款的信用凭证
        bool isRepay;           //是否已经还钱了
        string description;     //其它备注、描述
}
//应收款单据编号 1,2,3,...
uint receiptId;
```

Company

生产企业或者金融机构，金融机构包括银行可以认证交易

```
struct Company{
        address companyAddr;     //公司地址——主键
        string name;                //公司名字
        uint creditRating;          //公司信用等级
}
```

```
mapping(address => Company) public companies;   //参与供应链的公司
Company[] public banks;                         //参与供应链的银行
```

（三）核心功能

功能一：实现采购商品—签发应收账款交易上链

例如车企从轮胎公司购买一批轮胎并签订应收账款单据。

首先是设计两个映射方便后续使用。

pending是单号到收据的映射，存放等待签署到应收款单据。 收款方发起一个应收款单据，合约会生成一个单号，并将应收款单据放入pending，付款方根据单号签署应收款单据，此时单据正式生效，然后放入receipts中。

receipts单据收款方地址到Receipt[]的映射。给出一个收款方企业地址，可以方便的找到该企业所有的应收帐款。

```
//等待签署的应收款单据
mapping(uint => Receipt) public pending;
//公司到应收帐款的映射
mapping(address => Receipt[]) public receipts;
```

具体的通过IssueReceipt和SignReceipt两个函数实现，具体代码如下：

```
//公司发起应收款单据,owner为收款人，client为欠款人
function IssueReceipt(address owner, address client, uint amount, uint sta
rtDate, uint endDate) public returns(uint _id){
        require(
            msg.sender == owner,
            "You don't have permission to issue this receipt!"
        );
        pending[receiptId] = Receipt(receiptId, client, msg.sender, amount
, startDate, endDate, false, false, "");
        _id = receiptId;
        receiptId++;
        emit ReceiptIssued(owner, "Receipt Issued");
    }

//客户签署应收款单据
function SignReceipt(uint _id)public returns(bool){
        Receipt r = pending[_id];
        //签署人必须是该单据的欠款人
        require(
            r.from == msg.sender,
            "You don't have permission to sign this receipt! "
```

```
        );
        //单据未到期
        require(
            now < r.endDate,
            "You have passed the repayment date."
        );
        receipts[r.to].push(Receipt(r.id, r.from, r.to, r.amount, now, r.endDate, false, false, ""));
        emit ReceiptSigned(msg.sender, "Receipt Signed");
        return true;
    }
```

功能二：实现应收账款的转让上链

例如，轮胎公司从轮毂公司购买一笔轮毂，便将于车企的应收账款单据部分转让给轮毂公司。轮毂公司可以利用这个新的单据去融资或者要求车企到期时归还钱款。

```
function TransferTo(uint _receiptid, address to, uint amount) public returns(uint _id){
        Receipt storage senderReceipt;
        for (uint i = 0; i < receipts[msg.sender].length; i++){
            if (receipts[msg.sender][i].id == _receiptid){
                senderReceipt = receipts[msg.sender][i];
                break;
            }
            require(
                i != receipts[msg.sender].length - 1,
                "no such receipt id."
            );
        }
        require(senderReceipt.amount >= amount && amount > 0);
        //转移账款
        senderReceipt.amount -= amount;
        receipts[to].push(Receipt(receiptId, senderReceipt.from, to, amount, now, senderReceipt.endDate, false, false, ""));
        _id = receiptId;
        receiptId++;
        Transfered(msg.sender, to, amount, "transfer successfully");
    }
```

功能三：利用应收账款向银行融资上链，供应链上所有可以利用应收账款单据向银行申请融资。

下游企业向银行融资的前提是，核心公司与该企业之间存在应收账款交易，并且金额大于等于该企业需要融资的金额。如果可以进行融资，则与应收账款转让的结果类似，记录核心企业与银行之间的交易。

```
    function MakeLoan(address loanTo, uint loanAmount, unit receiptid) public returns(bool) {
        unit i;
        //只能由银行调用
        unit count = 0;
        for(i = 0; i < banks.length; i++){
            if(banks[i].ad == msg.sender){
                count++;
            }
        }
        require(count == 1);
        //遍历所有的应收账款，找到应收款单据
        Receipt storage rec;
        for(i = 0; i < receipts[loanTo].length; i++){
            if(receipts[loanTo][i].id == receiptid){
                rec = receipts[loanTo][i];
                break;
            }
            require(i != receipts[msg.sender].length - 1, "No such receipt id.");
        }
        //该单据已经被用于贷款
        require(rec.isusedLoan == false, "The receipt has already been used for loan.");

        //融资金额不能大于核心公司与该企业之间存在应收账款交易
        require(rec.amount >= loanAmount);
        //融资成功
        rec.isusedLoan = true;
        Loan(loanTo, loanAmount, "Loan Successfully.");
        return true;
    }
```

功能四：应收账款支付结算上链，应收账款单据到期时核心企业向下游企业支付相应的欠款。

结算函数只能由核心公司调用，采用的思路是遍历存放所有应收账款的数组，把到期还款的而还没结算的应收账款给结算掉，相应的结果是核心公司金额减少，账本相应收款人金额增加。

```
  function PayForReceipt(address owner, uint amount,uint receiptid) public returns(bool){
        Receipt storage r;
        uint i;
        for (i = 0; i < receipts[owner].length;i++)
        {
          //遍历所有应收账单
            if (receipts[owner][i].id==receiptid)
            {
                r = receipts[owner][i];
                break;
            }
            require(i != receipts[msg.sender].length - 1,"no such receipt id.");
        }

        require(r.to== msg.sender,"sender doesn't match receipt's client");
        //结算的金额不能够大于账单金额
        require(r.amount >= amount,"payment exceeds receipt'amount");
        r.amount -= amount;
        if(r.amount > 0)
            return true;
        for(;i<receipts[owner].length - 1;i++)
        {
            receipts[owner][i] = receipts[owner][i+1];
        }
         //结算成功
        delete receipts[owner][i];
        receipts[owner].length--;
        pay(msg.sender, owner, amount,"pay for receipt successfully.");
        return true;
    }
}
```