Supplychain报告

组号	姓名	学号	专业
12	李金敏	18342050	软件工程
12	赖培文	18342041	软件工程
12	李赞辉	18342053	软件工程

目录

实验要求

项目背景

传统供应链存在种种痛点：

- 传统的供应链金融交易信息不透明，银行准入条件比较高，商业汇票存在信用度低问题，导致中小型企业融资难 。
- 供应链上存在信息孤岛，银行需要对贷款公司进行详细的信用分析，以及公司财务能力分析，导致评估成本和风险较高。

总体介绍

- 基于区块链、智能合约等，实现基于区块链的供应链金融平台。

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。

- 实现功能
    - 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。
    - 实现应收账款的转让上链，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。
    - 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。
    - 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。

环境要求

- Linux Ubuntu 18.04.1
- FISCO-BCOS 2.7.0

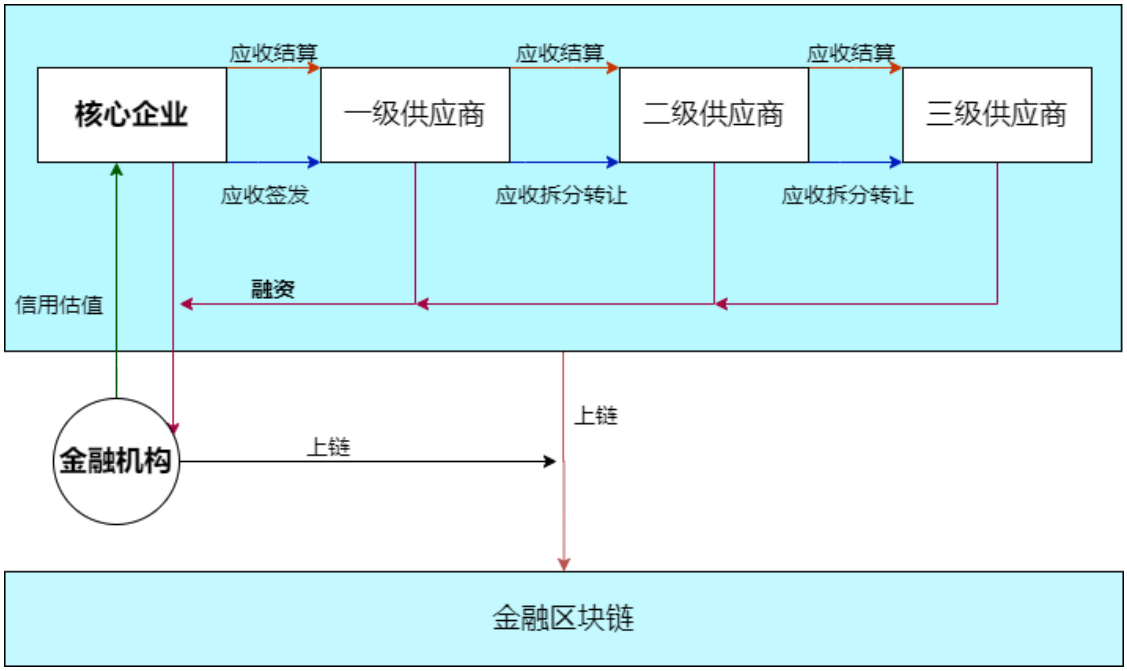
项目设计

问题分析

- 中小企业融资难
    - 传统的供应链金融交易信息不透明，银行准入条件比较高，商业汇票存在信用度低问题，导致核心企业信用传递难，下游中小型企业融资难 。
    - 缺乏可信的贸易场景，在供应链场景下，核心企业为可信的贸易背景，银行通常之服务核心企业及其一级供应商的融资需求，下游中小企业融资难
- 银行评估成本和风险高
    - 供应链上存在信息孤岛，银行需要对贷款公司进行详细的信用分析，以及公司财务能力分析，导致评估成本和风险较高。
    - 缺乏可信的贸易场景，银行需验证应收款单据的真实性，导致评估成本和风险较高。
- 履约风险无法有效控制
    - 供应商和核心企业、融资方、融资机构之间的支付和约定结算受限于各参与主体的契约精神和履约意愿，不确定因素较多，导致履约风险无法有效控制

合约设计

- 方案架构



- 存储设计
  - 金融机构，例如银行

```
struct Bank{
    address ad;//唯一标识
    string name;//银行名字
}
```

    - 企业

```
struct Company{
    address ad;//唯一标识
    string name;//公司名字
    uint money;//公司账户余额
    uint creditValues;//信用额度
}
```

    - 应收账款

```
struct Receipt{
    uint id;//唯一标识
    address from;//欠款方
    address to;//收款方
    uint money;//金额
    uint start_date;//开始时间
    uint end_date;//还款时间
    bool isLoan;//是否用来贷款
    string info;//其他信息
}
```

    - 存储到链上的数据
      - event 签发应收账款：包括发起和确认收到
      - event 应收账款转让
      - event 融资
      - event 应收账款到期结算

- 核心功能
  - 采购商品—签发应收账款，交易上链
    - 欠款方发起应收账款
      1. 检查消息发起者地址是否和Receipt中from一致，若不一致停止
      2. 加入Receipt
      3. 触发event 签发应收账款：已发起
    - 收款方签署应收账款
      1. 检查消息发起者地址是否和Receipt中to一致，若不一致停止
      2. 检查起始时间是否合法
      3. 触发event 签发应收账款：已签署
  - 应收账款的转让上链
    1. 检查链上的所有应收账款是否存在该转让应收账款，若无则停止
    2. 检查应收账款金额是否大于需要转让的金额，若不满足则停止
    3. 转让应收账款
    4. 触发event 应收账款转让
  - 利用应收账款向银行融资上链
    1. 检查是否是银行，若不是则停止
    2. 找到应收账款单据并检查是否有效，无效则停止
    3. 检查应收账款金额是否大于贷款的金额，若不满足则停止
    4. 触发event 融资
  - 应收账款支付结算上链
    1. 检查应收账款是否存在，若无则停止
    2. 检查欠款方和发起者是否一致，若不满足则停止
    3. 检查应收账款的金额是否大于等于还款金额，若不满足则停止
    4. 从应收账款的金额中减去还款金额，若不为0，则返回
    5. 删除该应收账款
    6. 触发event 应收账款到期结算

面向问题解释

- 解决中小企业融资难问题
  - 核心企业可以签发应收账款，然该应收账款可以部分转让给下一级企业，下游中小企业利用应收账款可直接向银行等金融机构融资，使得核心企业的信用逐级下传，解决了中小企业融资难问题。
- 解决银行评估成本和风险高问题
  - 银行向中下游企业融资时，无需再对该企业评估，只需要判断该企业所持有应收账款是否是来自核心企业的，根据核心企业的信用额度和还款能力即可评估风险，解决了银行评估成本和风险高问题。
- 解决履约风险无法有效控制问题
  - 目前还没写出到期自动还款部分，有待解决。

功能测试

智能合约部署

注意需要提前启动节点

通过控制台进行部署

```
deploy Supplychain
```

```
[group:1]> deploy Supplychain
transaction hash: 0x80cbe8ccce11c6d28001eb49212d121f1f664606ffcc687d7c15e858f47a
e93c
contract address: 0x16d8876c3a83b832f4279b4da896dd661c94fcc6
```

智能合约调用

SetBank

设置银行名字、地址和金额；

```
[group:1]> call Supplychain 0x16d8876c3a83b832f4279b4da896dd661c94fcc6 SetBank b
ank 0x444331593ed0b4125ef81086142f6761c759a42e 1000
transaction hash: 0xfb823d60eb18017775799eec55d942a28c5bda2b8ed0fd246a671179e8c1
1ea2
-------------------------------------------------------------------------------
-------------
transaction status: 0x0
description: transaction executed successfully
-------------------------------------------------------------------------------
-------------
Output
Receipt message: Success
Return message: Success
Return value: [true]
-------------------------------------------------------------------------------
-------------
Event logs
Event: {}
```

结果

```
[group:1]> call Supplychain 0x16d8876c3a83b832f4279b4da896dd661c94fcc6 bank
-------------------------------------------------------------------------------
-------------
Return code: 0
description: transaction executed successfully
Return message: Success
-------------------------------------------------------------------------------
-------------
Return values:
[
    "0x444331593ed0b4125ef81086142f6761c759a42e",
    "bank"
]
-------------------------------------------------------------------------------
-------------
```

SetCore

设置核心企业名字、地址和银行余额

```
[group:1]> call Supplychain 0x16d8876c3a83b832f4279b4da896dd661c94fcc6 SetCore c
ore 0x9b862df1657c55452639b8d742063539c4735318 1000
transaction hash: 0x3ac97439309c65d6191771e30ca4abc5ee2668d50a5185fb55bf5636c01b
b0be
---------------------------------------------------------------------------------
------------
transaction status: 0x0
description: transaction executed successfully
---------------------------------------------------------------------------------
------------
Output
Receipt message: Success
Return message: Success
Return value: [true]
---------------------------------------------------------------------------------
------------
Event logs
Event: {}
```

AddSME

添加中小企业SME1、SME2

```
[group:1]> call Supplychain 0x16d8876c3a83b832f4279b4da896dd661c94fcc6 AddSME SM
E1 0xabecaf5104016fed1766179bf67b7097ad71a398 100
transaction hash: 0x0d2e4fa80e3e8c01d5438a3716a816545be86584e783330e4e669b1d52a9
48e8
---------------------------------------------------------------------------------
------------
transaction status: 0x0
description: transaction executed successfully
---------------------------------------------------------------------------------
------------
Output
Receipt message: Success
Return message: Success
Return value: [true]
---------------------------------------------------------------------------------
------------
Event logs
Event: {}
```

```
[group:1]> call Supplychain 0x16d8876c3a83b832f4279b4da896dd661c94fcc6 AddSME SM
E2 0x895ceae4b130196f69d5b7a644edd5986d9b1f19 50
transaction hash: 0xed97b83bd72100eeeba7834dfc3be7c406e663c1797443c6ccd020ec6c1b
6c61
---------------------------------------------------------------------------------
------------
transaction status: 0x0
description: transaction executed successfully
---------------------------------------------------------------------------------
------------
Output
Receipt message: Success
Return message: Success
Return value: [true]
---------------------------------------------------------------------------------
------------
Event logs
Event: {}
```

结果

```
[group:1]> call Supplychain 0x16d8876c3a83b832f4279b4da896dd661c94fcc6 companys
0xabecaf5104016fed1766179bf67b7097ad71a398
-------------------------------------------------------------------------------
------------
Return code: 0
description: transaction executed successfully
Return message: Success
-------------------------------------------------------------------------------
------------
Return values:
[
    "0xabecaf5104016fed1766179bf67b7097ad71a398",
    "SME1",
    false
]
```

```
[group:1]> call Supplychain 0x16d8876c3a83b832f4279b4da896dd661c94fcc6 companys
0x895ceae4b130196f69d5b7a644edd5986d9b1f19
-------------------------------------------------------------------------------
------------
Return code: 0
description: transaction executed successfully
Return message: Success
-------------------------------------------------------------------------------
------------
Return values:
[
    "0x895ceae4b130196f69d5b7a644edd5986d9b1f19",
    "SME2",
    false
]
-------------------------------------------------------------------------------
------------
```

IssueReceipt

中小企业SME1发起应收账款单据，金额为200

```
[group:1]> call Supplychain 0x16d8876c3a83b832f4279b4da896dd661c94fcc6 IssueRece
ipt 0xabecaf5104016fed1766179bf67b7097ad71a398 0x9b862df1657c55452639b8d74206353
9c4735318 200 36000000 testReceipt
transaction hash: 0xc0cd33d542ce93fc4b1d845eac6f4af7d28ce69475913d40344a447e0a20
d1fd
-------------------------------------------------------------------------------
------------
transaction status: 0x0
description: transaction executed successfully
-------------------------------------------------------------------------------
------------
Output
Receipt message: Success
Return message: Success
Return value: [1]
-------------------------------------------------------------------------------
------------
Event logs
Event: {"ReceiptIssued":[["0xabecaf5104016fed1766179bf67b7097ad71a398","receipt
Issued"]]}
```

结果

单据被成功添加

```
[group:1]> call Supplychain 0x16d8876c3a83b832f4279b4da896dd661c94fcc6 pending 1

-------------------------------------------------------------------------------
------------
Return code: 0
description: transaction executed successfully
Return message: Success
-------------------------------------------------------------------------------
------------
Return values:
[
    1,
    "0xabecaf5104016fed1766179bf67b7097ad71a398",
    "0x9b862df1657c55452639b8d742063539c4735318",
    200,
    1607866731228,
    1607902731228,
    false,
    "testReceipt"
]
```

SignReceipt

核心企业签署应收账款单据

```
[group:1]> call Supplychain 0x16d8876c3a83b832f4279b4da896dd661c94fcc6 SignRecei
pt 0x9b862df1657c55452639b8d742063539c4735318 1
transaction hash: 0x56634848b496214f391fb6b81fa85b4d2a768f07be1e00232a9996b84b6a
5f69
-------------------------------------------------------------------------------
------------
transaction status: 0x0
description: transaction executed successfully
-------------------------------------------------------------------------------
------------
Output
Receipt message: Success
Return message: Success
Return value: [true]
-------------------------------------------------------------------------------
------------
Event logs
Event: {"ReceiptIssued":[["0x9b862df1657c55452639b8d742063539c4735318","receipt
signed"]]}
```

TransferTo

中小企业SME1转让应收账款10给SME2

```
[group:1]> call Supplychain 0x16d8876c3a83b832f4279b4da896dd661c94fcc6 TransferT
o 0xabecaf5104016fed1766179bf67b7097ad71a398 1 0x895ceae4b130196f69d5b7a644edd59
86d9b1f19 10
transaction hash: 0xdb4b261eb91d72db0d61bdc945c0a7f6a7ddaf163b2f09c8cba82513dcd6
219e
-------------------------------------------------------------------------------
------------
transaction status: 0x0
description: transaction executed successfully
-------------------------------------------------------------------------------
------------
Output
Receipt message: Success
Return message: Success
Return value: [2]
-------------------------------------------------------------------------------
------------
Event logs
Event: {"Transfered":[["0xabecaf5104016fed1766179bf67b7097ad71a398","0x895ceae4b
130196f69d5b7a644edd5986d9b1f19",10,"transfer successfully"]]}
```

结果

单据发生改变

```
[group:1]> call Supplychain 0x16d8876c3a83b832f4279b4da896dd661c94fcc6 receipts
0x895ceae4b130196f69d5b7a644edd5986d9b1f19 0
---------------------------------------------------------------------------
------------
Return code: 0
description: transaction executed successfully
Return message: Success
---------------------------------------------------------------------------
------------
Return values:
[
    2,
    "0x895ceae4b130196f69d5b7a644edd5986d9b1f19",
    "0x9b862df1657c55452639b8d742063539c4735318",
    10,
    1607867198962,
    1607902731228,
    false,
    "testReceipt"
]
```

MakeLoan

银行根据单据发放贷款给SME1

```
[group:1]> call Supplychain 0x16d8876c3a83b832f4279b4da896dd661c94fcc6 MakeLoan
0x444331593ed0b4125ef81086142f6761c759a42e 0xabecaf5104016fed1766179bf67b7097ad7
1a398 1 10
transaction hash: 0x5f7f82ab8a685f53dd03a1a16b32b16b6eae9254b578347619da60aa6fb2
1d3d
---------------------------------------------------------------------------
------------
transaction status: 0x0
description: transaction executed successfully
---------------------------------------------------------------------------
------------
Output
Receipt message: Success
Return message: Success
Return value: [true]
---------------------------------------------------------------------------
------------
Event logs
Event: {"Loaned":[["0x444331593ed0b4125ef81086142f6761c759a42e","0xabecaf5104016
fed1766179bf67b7097ad71a398",10,"loaned successfully."]]}
```

PayForReceipt

付款前

```
[group:1]> call Supplychain 0x16d8876c3a83b832f4279b4da896dd661c94fcc6 balances
0x9b862df1657c55452639b8d742063539c4735318
---------------------------------------------------------------------------
------------
Return code: 0
description: transaction executed successfully
Return message: Success
---------------------------------------------------------------------------
------------
Return values:
[
    1000
]
---------------------------------------------------------------------------
------------
```

```
[group:1]> call Supplychain 0x16d8876c3a83b832f4279b4da896dd661c94fcc6 balances
0xabecaf5104016fed1766179bf67b7097ad71a398
-------------------------------------------------------------------------------
-------------
Return code: 0
description: transaction executed successfully
Return message: Success
-------------------------------------------------------------------------------
-------------
Return values:
[
    110
]
```

核心企业根据单据支付SME1金额10

```
[group:1]> call Supplychain 0x16d8876c3a83b832f4279b4da896dd661c94fcc6 PayForRec
eipt 0x9b862df1657c55452639b8d742063539c4735318 0xabecaf5104016fed1766179bf67b70
97ad71a398 10 1
transaction hash: 0xd97e9744dba340781167cf6d981e48bb393b44259ecff3f068396f105248
408b
-------------------------------------------------------------------------------
-------------
transaction status: 0x0
description: transaction executed successfully
-------------------------------------------------------------------------------
-------------
Output
Receipt message: Success
Return message: Success
Return value: [true]
```

结果

付款后，对比上图可见：

核心企业的银行余额减少10

```
[group:1]> call Supplychain 0x16d8876c3a83b832f4279b4da896dd661c94fcc6 balances
0x9b862df1657c55452639b8d742063539c4735318
-------------------------------------------------------------------------------
-------------
Return code: 0
description: transaction executed successfully
Return message: Success
-------------------------------------------------------------------------------
-------------
Return values:
[
    990
]
```

中小企业的银行余额增加10

```
[group:1]> call Supplychain 0x16d8876c3a83b832f4279b4da896dd661c94fcc6 balances
0xabecaf5104016fed1766179bf67b7097ad71a398
-------------------------------------------------------------------------------
-------------
Return code: 0
description: transaction executed successfully
Return message: Success
-------------------------------------------------------------------------------
-------------
Return values:
[
    120
]
```

功能实现解释

单据实现

```
struct Receipt{
    uint id;//唯一标识
    address from;//欠款方
    address to;//收款方
    uint money;//金额
    uint start_date;//开始时间
    uint end_date;//还款时间
    bool isLoan;//是否用来贷款
    string info;//其他信息
}
```

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。

发起单据通过`IssueReceipt`进行：根据传入参数创建单据，单据被放入等待签署的队列中；

签署单据通过`SignReceipt`进行：从等待签署的队列中查找所属企业的单据，进行签署，存入相应中小企业的已签署单据队列中；

```
function IssueReceipt(address SME, address core, uint amount, uint timeInterval,
string info) public returns(uint id_) {
    pending[rid] = Receipt(rid, SME, core, amount, now, now+timeInterval, false,
info);
    id_ = rid;
    rid++;
    emit ReceiptIssued(SME, "receipt Issued");
}

function SignReceipt(address core, uint receiptID) public returns(bool) {
    Receipt storage r = pending[receiptID];
    //签署人必须是核心企业
    require(companys[core].creditValues);
    //单据未到期
    require(now < r.end_date);
    receipts[r.SME].push(r);
    emit ReceiptIssued(core, "receipt signed");
    return true;
}
```

功能二

实现应收账款的转让上链，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。

中小企业通过`TransferTo`转让应收账款：先from企业的已签署单据队列中寻找到相应单据，判断是否满足条件，如不满足则报错，否则则修改单据，减去相应金额，然后创建新的单据，将相应金额填入；

```
function TransferTo(address from, uint receiptID, address to, uint amount)
public returns(uint id_) {
    Receipt storage fromReceipt;
    uint len = receipts[from].length;
```

```
    for (uint i = 0; i < len; i++) {
        if (receipts[from][i].id == receiptID) {
            fromReceipt = receipts[from][i];
            break;
        }
        require(i != len-1, "no such receipt id.");
    }

    require(fromReceipt.amount >= amount && amount > 0);
    //转移账款
    fromReceipt.amount -= amount;
    receipts[to].push(Receipt(rid, to, fromReceipt.core, amount, now,
fromReceipt.end_date, fromReceipt.isLoan, fromReceipt.info));
    id_ = rid;
    rid++;
    emit Transfered(from, 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。

通过`MakeLoan`向指定银行申请指定单据 、指定金额的融资：从已签发队列中找到相应的单据，判断是否已用于贷款，如未贷款，则执行；

```
function MakeLoan(address _bank, address loanTo, uint receiptID, uint
loanAmount) public returns(bool) {
    //必须是银行发放贷款
    require(_bank == bank.ad, "Only the bank can make loan");

    //找到应收款单据
    Receipt storage r;
    uint i;
    uint len = receipts[loanTo].length;
    for (i = 0; i < len; i++) {
        if (receipts[loanTo][i].id == receiptID) {
            r = receipts[loanTo][i];
            break;
        }
        require(i != len-1, "no such receipt id.");
    }

    //true说明该单据已经被用来贷款了
    require(r.isLoan == false, "the receipt has been used for loan");
    require(r.amount >= loanAmount);
    require(companys[r.core].creditValues);

    r.isLoan = true;
    balances[bank.ad] -= loanAmount;
    balances[loanTo] += loanAmount;
    emit Loaned(_bank, loanTo, loanAmount, "loaned 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。

通过`PayForReceipt`实现核心企业向下游企业支付欠款：查找到相应的单据，然后判断是否符合还款条件，然后执行还款，修改两家企业的银行余额；

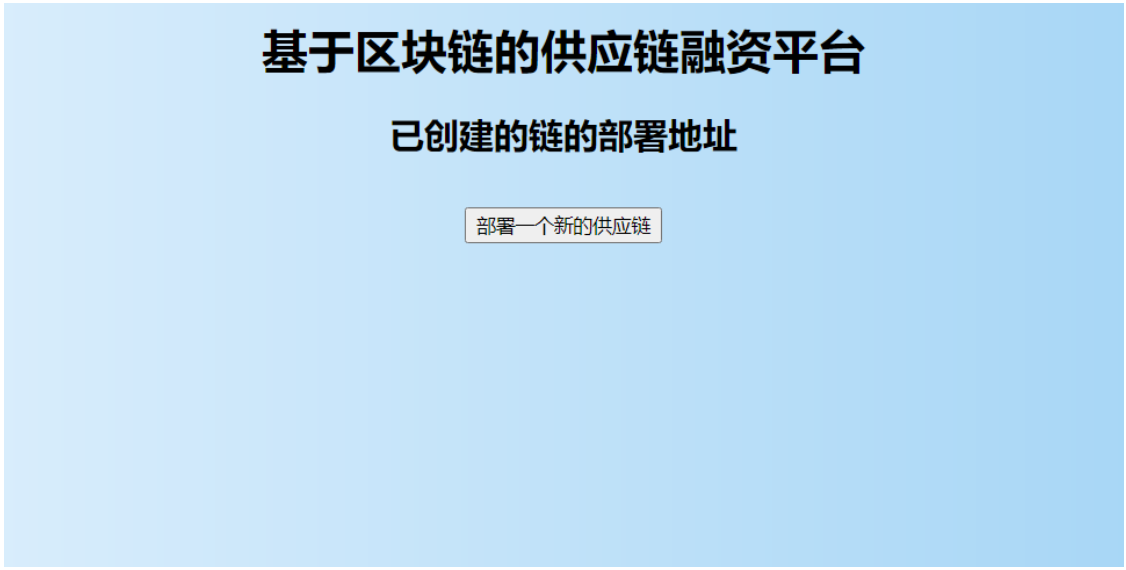
```
function PayForReceipt(address core, address SME, uint amount, uint receiptID)
public returns(bool) {
    Receipt storage r;
    uint i;
    uint len = receipts[SME].length;
    for (i = 0; i < len; i++) {
        if (receipts[SME][i].id == receiptID) {
            r = receipts[SME][i];
            break;
        }
        require(i != len - 1,"no such receipt id.");
    }

    require(r.core == core,"sender doesn't match receipt's client");
    require(r.amount >= amount,"payment exceeds receipt'amount");
    r.amount -= amount;
    balances[core] -= amount;
    balances[SME] += amount;
    if(r.amount > 0)
        return true;
    for(; i < len-1; i++) {
        receipts[SME][i] = receipts[SME][i+1];
    }
    delete receipts[SME][i];
    //receipts[owner].length--;
    emit pay(core, SME, amount, "pay for receipt successfully.");
    return true;
}
```

前后端展示

前端

- 注册登录界面

- 部署新的供应链界面



- 详细信息界面

后端

- 后端与链端交互

使用 FISCO BCOS 官方提供的 Node.js SDK，该SDK内嵌CLI工具，供用户在命令行中与区块链进行交互。通过在处理函数中创建子进程来执行shell命令，复用现有CLI工具来实现数据上链和获取数据。

以“合约供应链部署”为例子展示如何复用CLI工具与区块链进行交互

```
//导入child_process的exec
var exec = require('child_process').exec;
//设置CLI工具路径
```

```
var path = '../nodejs-sdk/packages/cli/cli.js'

function deploy() {
    //使用CLI工具部署供应链合约
    cmd = path + ' ' + 'deploy Supplychain';
    exec(cmd,
    //回调函数处理标准输出
    function (error, stdout, stderr) {
        //将命令行输出解析成JSON格式
        result = JSON.parse(stdout);
        console.log(result);
        //设置合约地址为部署成功
        contractAddress = result.contractAddress;

        if (error !== null) {
            console.log('exec error: ' + error);
        }
    });
}
```

- 前端与后端交互

使用 Node.js 的 http 包创建 Server 对指定端口进行监听，通过路由设置相应处理函数对前端请求进行处理（尚未完成）

```
http.createServer(function (request, response) {

    //路由设置

}).listen(8080);
```

实验总结

- 项目设计

  参考一些区块链合约的资料之后，我们初步拟定了整个合约所需要的数据结构和要实现的函数功能。

- 项目实现

  根据设计，定义基本数据结构 `Bnak` 、 `Company` 和 `Receipt` 及其相应字段，定义所需要的映射和队列，定义设置函数和功能函数；接着，按照功能要求，对相应函数进行实现。

- 项目测试

  编译部署 `Supplychain` 合约到链上，先设置好银行、核心企业和中小企业，然后逐步调用相应函数进行功能测试，查看返回结果和最终状态，均符合预期。

- 前端

  用HTML写了前端界面：注册登录界面、部署新的供应链界面、详细信息界面。

- 后端

  使用官方提供的Node.js SDK中CLI工具与链端进行交互，编写路由设置相应的处理函数对前端请求进行处理。

- 感想

通过实验，我们对该供应链的整体结构与功能都有了深入了解，初步运用 solidity 语言编写代码。在代码编写和测试的过程中，也遇到很多语法问题和不知所以的bug，需要借助各类参考资料来解决，最后总算是能够实现基本功能。在整个实验过程中，通过亲自动手实践，我们更加理解了课程上所学的一些区块链的一些知识，对区块链的底层以及应用部分有了深刻和清晰的认识。

- 不足与改进

  对于前端，我们的前端制作还是简陋一些，需要添加更加丰富的内容。

  对于后端，在与链端的交互中，可以更进一步调用Node.js API进行二次开发，降低流程复杂度，提高执行速度，但由于并没有完全读懂CLI工具的实现，无法很好地整合使用Node.js API，最后直接复用Node.js SDK的CLI工具；在与前端的交互中，可以更加细致地将这部分与链端交互部分分离，后端直接向前端提供API接口，可以更好地分离前端和链端，但是出于快速开发，方便调整，选择直接放在一起。

  对于项目，目前我们后端已经可以和链端交互，还缺乏路由设置和请求处理函数的前后端交互，后续会添加。

参考材料

- FISCO BCOS - Node.js SDK
- FISCO BCOS - 案例精编
- 蚂蚁集团 - 双链通供应链金融服务平台
- 区块链+应收账款融资
- 区块链与供应链金融 白皮书（1.0 版）