2020年秋季学期区块链大作业：基于区块链的供应链金融平台

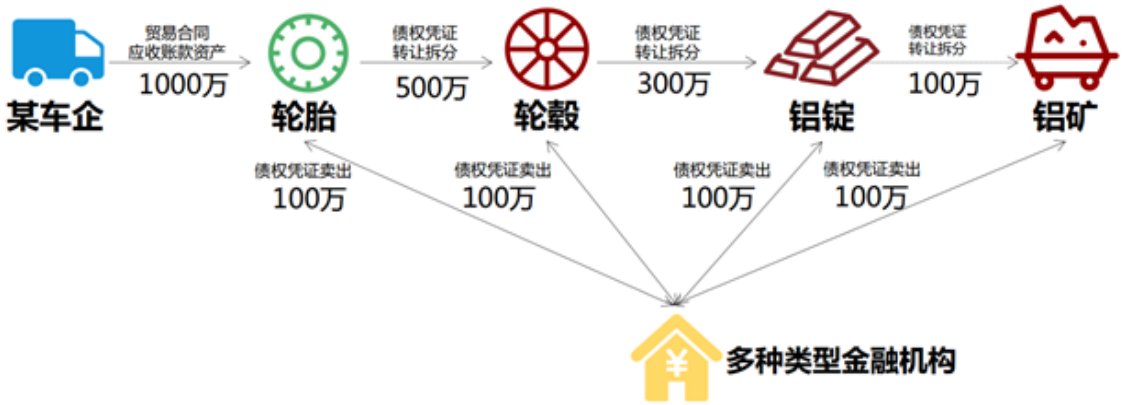
实验报告

组员：18340174 吴晓淳 18340166 王若琪
时间：2021年1月26日

一、题目介绍

基于已有的开源区块链系统FISCO-BCOS（https://github.com/FISCO-BCOS/FISCO-BCOS），以联盟链为主，开发基于区块链或区块链智能合约的供应链金融平台，实现**供应链应收账款资产的溯源、流转。**

场景介绍：



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。

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。

实现功能：

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。

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。

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。

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。

加分项（30分）：

a. 功能：除了必要功能之外，实现更多的功能，限定：供应链金融领域相关

b. 底层：改进区块链平台底层或自行开发区块链（共识算法等）

c. 合约：实现链上数据隐私保护（同态加密、属性加密等）

d. 前端：友好高效的用户界面

e. 其他有挑战性的创新

二、实验过程

2.1 存储设计

针对本应用设计一个企业管理的表 t_company 和应收账款单据管理的表`t_receipt`，其中，表`t_company`字段如下所示：

- `company`：主键，企业名称(string类型)
- `item`：产品名称(string类型)
- `balance`：企业资产(int256类型)

其中， company 是主键，即操作 t_company 表时需要传入的字段，区块链根据该主键字段查询表中匹配的记录。

表 t_receipt 字段如下所示：

- `key`：主键(string类型)
- `from`：付款企业名称(string类型)
- `to`：收款企业名称(string类型)
- `amount`：账款金额(int256类型)
- `due_date`：还款截止日期(string类型)
- `status`：应收账款单据状态(string类型)

其中，`key`是主键.

2.2 主要功能设计

① 转账

当企业在资金充足的情况下采购商品时，将支付的过程视为企业之间的转账；当企业利用应收账款向银行申请融资时，银行参与融资，即发放贷款的过程可以视为银行向企业转账； 当企业支付结算应收账款时，将支付结算的过程视为企业之间的转账。

② 采购商品，签发应收账款

当企业A向企业B采购商品时，若采购金额大于企业A现有资产的一半，则假定企业A资金短缺并选择暂不向企业B转账，而是向企业B签订了对应金额的应收账款单据，承诺于某一天归还企业B，即向表 t_receipt 中插入字段 from 为企业A， to 为企业B， amount 为对应金额， due_date 为还款日期的记录，并设置字段 status 为“identified”表示银行为这笔交易作见证，从而确定这笔交易的真实性，如若采购金额小于等于企业A现有资产的一半，则企业A选择立即向企业B转账，从而完成这笔交易。

③ 转让应收账款

当应收账款单据表 t_receipt 中已经有企业A向企业B签订的应收账款单据以及企业B向企业C签订的应收账款单据的记录时，若后者的应收账款金额比前者少，则企业B可以选择将企业A向它签订的应收账款金额的一部分转让给企业C，从而使得企业B向企业C签订的应收账款金额减少，并向表 t_receipt中加入由于转让产生的企业A向企业C签订的应收账款单据记录，字段 from 为企业A， to 为企业C， amount 为转让账款金额，due_data 为原来企业A和企业B约定好的还款日期，并设置字段 status 为 “identified”，等待银行进行见证后变为 “authorized”。

④ 支付结算应收账款

在表 t_receipt 的 from 属性中的企业应于还款日期前向 to 属性中的企业支付相应欠款，具体表现为调用 transferBalance 函数实现企业之间的转账，并从表 t_receipt 中删去对应的应收账款单据记录表示成功支付结算应收账款。

⑤ 利用应收账款向银行申请融资

在表 t_receipt 中插入字段 from 为银行(“bank”)， to 为企业， amount 为融资金额的应收账款单据记录，并调用 transferBalance 函数实现企业向银行的贷款。

⑥ 银行进行认证

银行选择之前交易记录并进行认证，使得字段 status 之前的 “identified”变为 “authorized”。银行进行认证后，记录才能生效，未认证的记录不能参与其他功能的完成。

2.3 核心代码介绍

2.3.1 注册

企业注册时需要填写企业名称，产品信息以及初始企业资金，并检查该企业是否已经注册，返回值中 0 表示注册成功，-1 表示企业已经存在，-2 表示其他错误。

链端代码：

企业注册时需要填写企业名称，产品信息以及初始企业资金，并检查该企业是否已经注册，返回值中 0 表示注册成功，-1 表示企业已经存在，-2 表示其他错误。

```
    function register(string memory company, string memory address_, string memory type_, int balance) public returns(int256){
        //int256 ret_code = 0;
        int256 ret= 0;
        string memory temp_address = "";
        string memory temp_type = "";
        int temp_balance = 0;

        // 查询企业是否存在
        (ret, temp_address, temp_type, temp_balance) = selectFromCompany(company);
        if(ret != 0) {
            //打开表
            TableFactory tf = TableFactory(0x1001);
            Table table = tf.openTable("haha_t");

            Entry entry = table.newEntry();
            entry.set("company", company);
            entry.set("address", address_);
            entry.set("type", type_);
            entry.set("balance", balance);

            // 插入
```

```
            int count = table.insert(company, entry);
            if (count == 1) {
                // 成功
                return 0;
            } else {
                // 失败? 无权限或者其他错误
                return -2;
            }
        } else {
            // 企业已存在
            return -1;
        }

    }
```

后端代码：

当在用户操作点击注册，输入信息并确认后，会读入用户输入的名称name、密码password、启动资金balance，并判断名称的合法性，如果超过256就判断不合法，并显示错误提示。接下来完成地址、公钥、私钥的设置，最后调用智能合约（链端代码）中的register函数，并根据返回值作出显示判断，如果正常执行完成，就输出注册成功的提示框。

```
    def press_register(self):
        name, password, balance = self.line_name.text(), self.line_pwd.text(), self.line_balance.text()
        # balabce代表启动资金
        balance = int(balance)
        if len(name) > 256:
            QMessageBox.warning(self, 'Error', '名称过长。')
            sys.exit(1)
        print("starting : {} {} ".format(name, password))
        ac = Account.create(password)
        print("new address :\t", ac.address)
        print("new privkey :\t", encode_hex(ac.key))
        print("new pubkey :\t", ac.publickey)

        kf = Account.encrypt(ac.privateKey, password)
        keyfile = "{}/{}.keystore".format(client_config.account_keyfile_path, name)
        print("save to file : [{}]".format(keyfile))
        with open(keyfile, "w") as dump_f:
            json.dump(kf, dump_f)
            dump_f.close()
        print(
            "INFO >> Read [{}] again after new account,address & keys in file:".format(keyfile))
        with open(keyfile, "r") as dump_f:
            keytext = json.load(dump_f)
            privkey = Account.decrypt(keytext, password)
            ac2 = Account.from_key(privkey)
            print("address:\t", ac2.address)
            print("privkey:\t", encode_hex(ac2.key))
            print("pubkey :\t", ac2.publickey)
            print("\naccount store in file: [{}]".format(keyfile))
            dump_f.close()

        global client, contract_abi, to_address
```

```
        args = [name, ac.address, 'Company', balance]
        # 调用智能合约中的register函数
        receipt =
client.sendRawTransactionGetReceipt(to_address,contract_abi,"register",args
)
        print("receipt:",receipt['output'])

        QMessageBox.information(self,'Prompt','注册成功。', QMessageBox.Ok)
```

2.3.2 查询应收账款单据记录

根据模式mode在表 t_receipt 中查询记录，当mode为0时，返回字段 from 符合条件的记录还可以指定字段；当 mode为1时，返回字段 to 符合条件的记录；还可以指定字段 from 和 to 的内容在表中查询 t_receipt 符合条件的记录。

描述 : 根据企业名称查询产品信息和企业资产

参数： company_name : 企业名称

返回值：

参数一：成功返回0，企业不存在返回-1
参数二：第一个参数为0时有效，产品信息
参数三：第二个参数为0时有效，企业资产

```
    function selectFromCompany(string company) public constant
returns(int256, string memory, string memory, int) {
        // 打开表
        TableFactory tf = TableFactory(0x1001);
        Table table = tf.openTable("haha_t");
        // 查询
        Entries entries = table.select(company, table.newCondition());
        return getResult2(entries);
    }
```

```
    function select(string memory from, string memory to) public constant
returns(string[] memory,string[] memory,int[] memory,string[]
memory,string[] memory) {
        TableFactory tf = TableFactory(0x1001);
        Table table = tf.openTable("wow_t");
        Condition condition = table.newCondition();
        condition.EQ("from",from);
        condition.EQ("to",to);
        Entries entries = table.select("active", condition);
        return getResult1(entries);
    }
```

2.3.3 采购商品，签发应收账款

当企业A向企业B采购商品时，若采购金额大于企业A现有资产的一半，则假定企业A资金短缺并选择暂不向企业B转账，而是向企业B签订了对应金额的应收账款单据，承诺于某一天归还企业B，如若采购金额小于等于企业A现有资产的一半，则企业A选择立即向企业B转账，从而完成这笔交易；返回值中0 表示采购成功，-1 表示付款企业不存在，-2 收款企业不存在，-3 表示其他错误。

链端代码：

```
```

```
    function purchase(string memory from, string memory to, int amount, string memory due_date) public returns(int256) {
        int256 ret = 0;
        string memory temp_address = "";
        string memory temp_type = "";
        int balance = 0;

        (ret, temp_address, temp_type, balance) = selectFromCompany(to);
        if(ret!=0){
            return -2;
        }

        (ret, temp_address, temp_type, balance) = selectFromCompany(from);
        if(ret!=0){
            return -1;
        }

    if(balance>>1 >= amount){
        ret = transferBalance(from, to, amount);
        if(ret==0){
            return 0;
        }
        else{
            return -3;
        }
    }
    else{
            int256 count = insert(from,to,amount,due_date,"identified");
            if(count==1){
                return 0;
            }
            else{
                return -3;
            }
        }
    }
```

后端代码：

在得到用户输入信息（向谁购买、购买金额）后，提取信息并组成参数，调用智能合约（链端代码）中的purchase函数，并作出相应的显示，如果正常执行，就输出购买成功的提示框。

```
    # 提交purchase设置
    def submit_purchase(self):
        global client, contract_abi, to_address
        args = [self.line_pur_from.text(), self.company_name , int(self.line_pur_amt.text()), self.purchase_date.dateTime().toString("yyyy/MM/dd hh:mm:ss")]
        client.sendRawTransactionGetReceipt(to_address, contract_abi, "purchase", args)
        QMessageBox.information(self,'Prompt','购买成功。', QMessageBox.Ok)
```

2.3.4 转让应收账款

当应收账款单据表 t_receipt 中已经有企业A向企业B签订的应收账款单据以及企业B向企业C签订的应收账款单据的记录时，企业B可以选择将企业A向它签订的应收账款金额的一部分转让给企业C，从而使得企业B向企业C签订的应收账款金额减少，并向表 t_receipt 中加入由于转让产生的企业A向企业C签订的应收账款单据记录；返回值中 0 表示转让成功，-1 表示付款企业不存在，-2 表示收款企业不存在，-3 表示第三方企业不存在，-4表示转让金额超过付款企业和收款企业之间的应收账款金额，-5 表示转让金额超过收款企业和第三方企业之间的应收账款金额。

链端代码：

描述：转让应收账款

参数：

from ：付款企业名称

to ：收款企业名称

third ：第三方企业名称

amount：账款金额

返回值：

0 采购成功

-1 付款企业不存在

-2 收款企业不存在

-3 第三方企业不存在

-4 转让金额超过付款企业和收款企业之间的应收账款金额

-5 转让金额超过收款企业和第三方企业之间的应收账款金额

```
    function transfer(string memory from, string memory to, string memory third, int amount) public returns(int256){
        int256 ret = 0;
        string memory temp_address = "";
        string memory temp_type = "";
        int balance = 0;
        string[] memory from_list;
        string[] memory to_list;
        string[] memory due_date_list;
        int[] memory amount_list;
        string[] memory status_list;

        (ret, temp_address, temp_type, balance) = selectFromCompany(from);
        if(ret!=0){
            return -1;
        }
        (ret, temp_address, temp_type, balance) = selectFromCompany(to);
        if(ret!=0){
            return -2;
        }
        (ret, temp_address, temp_type, balance) = selectFromCompany(third);
        if(ret!=0){
            return -3;
        }
     (from_list,to_list,amount_list,due_date_list,status_list) = select(from, to);
```

```
        if(amount_list[uint256(0)] < amount){
            return -4;
        }
        update(from, to, amount_list[uint256(0)]-amount,
status_list[uint256(0)], due_date_list[uint256(0)]);
        insert(from, third, amount, due_date_list[uint256(0)],
"identified");

        (from_list, to_list, amount_list, due_date_list,status_list) =
select(to, third);
        if(amount_list[uint256(0)] < amount){
            return -5;
        }
        update(to, third, amount_list[uint256(0)]-amount,
status_list[uint256(0)], due_date_list[uint256(0)]);
        return 0;
    }
```

后端代码：

得到用户的输入和选择（选择两条记录，从而根据选择判断出付款企业、收款企业和第三方企业，并确定金额）后，调用调用智能合约（链端代码）中的transfer函数，并作出相应的显示。

```
    def submit_transfer(self):
        global client, contract_abi, to_address
        if self.table_trans_lent.selectionModel().hasSelection() and
self.table_trans_bor.selectionModel().hasSelection():
            row_lent = self.table_trans_lent.currentRow()
            row_bor = self.table_trans_bor.currentRow()
            args = [self.table_trans_lent.item(row_lent, 1).text(),
self.company_name, self.table_trans_bor.item(row_bor, 0).text(),
int(self.line_trans_amt.text())]
            print(args) # 在终端输出，便于调试
            if self.table_trans_bor.item(row_bor, 3).text() == "authorized"
and self.table_trans_lent.item(row_lent, 3).text() == "authorized":
                result = client.sendRawTransactionGetReceipt(to_address,
contract_abi, "transfer", args)
                print("receipt:",result['output'])
                # 由于难以找到bug，所以这一功能暂时只能在链端用命令实现
                QMessageBox.information(self,'Prompt','请在链端使用该功能。',
QMessageBox.Ok)

            else:
                QMessageBox.warning(self,'Error','请先认证涉及到的交易。',
QMessageBox.Ok)
        else:
            QMessageBox.warning(self,'Prompt','请先选择记录。',
QMessageBox.Ok)
```

2.3.5 清算应收账款，支付欠款

应收账款到期时核心企业向下游企业支付相应的欠款；返回值中 0 表示清算成功，-1 表示其他错误。

描述 : 清算应收账款，支付欠款

参数：

from ：付款企业名称

to ：收款企业名称

amount : 账款金额

due_date: 还款日期

返回值：

0 清算成功

-1 其他错误

链端代码：

```
    function repay(string memory from, string memory to, int amount, string memory due_date) public returns(int256){
        int256 ret_code = 0;
        int256 ret = transferBalance(from, to, amount);
        int256 count = remove(from, to, amount, due_date);
        if(count == 1 && ret == 0){
            ret_code = 0;
        }
        else{
            ret_code = -1;
        }
        return ret_code;
    }
```

后端代码：

得到用户的输入和选择（选择相应的欠款条目）后，调用调用智能合约（链端代码）中的repay函数，并进行是否认证过之类的判断，并作出相应的显示，如果未认证，就提示未认证，如果未选择记录就提示用户选择，还款成功或失败也都输出响应提示。

```
    def repay(self):
        global client, contract_abi, to_address
        if self.table_repay.selectionModel().hasSelection():
            row = self.table_repay.currentRow()
            args = [self.table_repay.item(row, 0).text(), self.table_repay.item(row, 1).text(), \
                    int(self.table_repay.item(row, 2).text()),self.table_repay.item(row, 4).text()]
            print(args)
            if self.table_repay.item(row, 3).text() == "authorized":
                # 调用智能合约中的repay函数，并返回结果
                result = client.sendRawTransactionGetReceipt(to_address, contract_abi, "repay", args)
                print("receipt:",result)
                res = hex_to_signed(result['output'])
                if res == 0:
                        QMessageBox.information(self,'Prompt','还款成功。', QMessageBox.Ok)
                else:
                        QMessageBox.warning(self,'Prompt','还款失败。', QMessageBox.Ok)
                self.table_repay.setRowCount(0)
                self.set_table_repay_content(self.company_name)
            else:
```

```
                QMessageBox.warning(self,'Error','请先认证相关交易。',
QMessageBox.Ok)
        else:
            QMessageBox.warning(self,'Prompt','请先选择相关记录。',
QMessageBox.Ok)
```

2.3.6 利用应收账款向银行申请融资

供应链上所有企业可以利用应收账款单据向银行申请融资；返回值中 0 表示融资成功，-1 表示其他错误。

描述 : 向银行申请融资

参数 :

to ：申请融资企业名称

amount : 账款金额

due_date: 还款日期

返回值:

0 融资成功

-1 其他错误

链端代码：

```
    function finance(string memory to, int256 amount, string memory
due_date) public returns(int256){
        int256 count = insert(to,"bank",amount,due_date,"identified");
        int ret = transferBalance("bank",to,amount);
        if(count==1 && ret==0){
            return 0;
        }
        else{
            return -1;
        }
    }
```

后端代码：

得到用户的输入后，调用调用智能合约（链端代码）中的finance函数，并进行是否认证过之类的判断，并作出相应的显示，如果符合融资的要求，就提示融资成功，如果不符合要求，则提示无法进行融资。

```
    def submit_finance(self):
        _amt = int(self.line_fin_amt.text())
        _due = self.finance_date.dateTime().toString("yyyy/MM/dd hh:mm:ss")
        if _amt > (self.total_lent - self.total_borrowed):
            QMessageBox.warning(self,'Error',"无法进行融资，当前容量为{}.".format(str(self.total_lent - self.total_borrowed)), QMessageBox.Ok)
        else:
            global client, contract_abi, to_address
            args = [self.company_name, _amt,_due]
            client.sendRawTransactionGetReceipt(to_address, contract_abi, "finance", args)
            QMessageBox.information(self,'Prompt','融资成功。', QMessageBox.Ok)
```

三、功能测试及界面效果

下面展示一些特定功能测试时的界面，具体测试操作请参考测试视频。

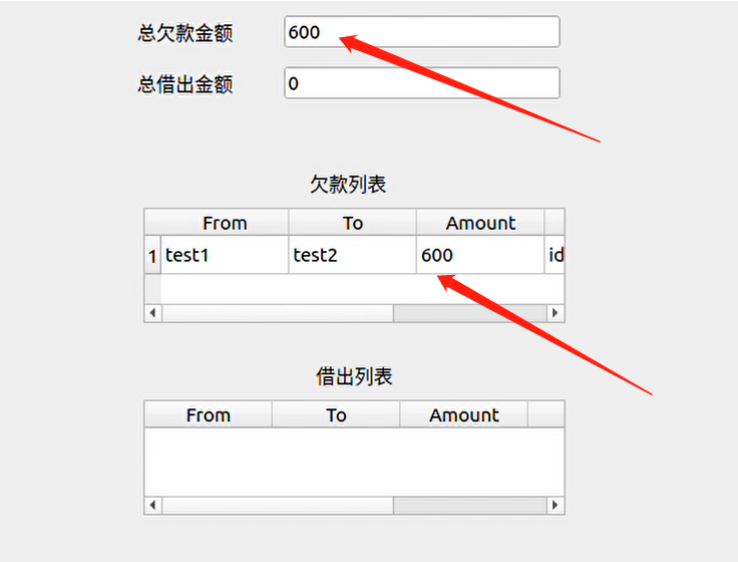
登录界面：

注册界面：

购买界面:

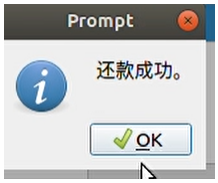
购买成功:

test1 向 test2 购买 600 元的物品后，增加一条记录，并且欠款金额改变:



认证：

还款：



转移：

（由于转移界面有未找到的bug，所以目前只能在链端进行测试）

在链端可以正常使用该功能：

调用 transfer 接口完成企业Bob向企业Cindy转让企业Alice和Bob之间的部分应收账款4 000的操作，由于4 000小于等于Bob和Cindy之间的应收账款金额7 000，并且4 000小于等于Alice和Bob之间的应收账款10 000，因此，转让成功，Bob和Cindy之间的应收账款单据的金额由7 000减少为3 000，并且向表 t_receipt 中插入字段 from 为企业Alice， to 为企业Cindy， amount 为4 000， due_date 为原来Alice和Bob约好的还款日期”20201207“的记录，这笔交易后通过调用selectFromReceipt 接口查询检查上述插入新的账款单据记录的操作，发现操作生效，即转让应收账款成功。

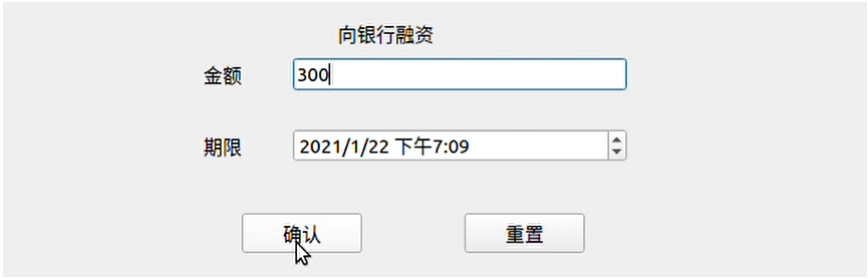
```
[group:1]> call supplychain 0x3e46381f73b08fdf6fc1c207a62aeaeeaa4f422d transfer Alice Bob Cindy 4000

transaction hash: 0x825a35b1835c79fdbabcc9527d620fcd63ca34862e8b62f3c75f9bd5397641f9
-------------------------------------------------------------------------------------------
transaction status: 0x0
description: transaction executed successfully
-------------------------------------------------------------------------------------------
Output
Receipt message: Success
Return message: Success
Return value: 0
-------------------------------------------------------------------------------------------
Event logs
Event: {"InsertResult":[[1]],"UpdateResult":[[1],[1]]}
```

```
[group:1]> call supplychain 0x3e46381f73b08fdf6fc1c207a62aeaeeaa4f422d selectFromReceipt Alice 0
---------------------------------------------------------------------------------------------
Return code: 0
description: transaction executed successfully
Return message: Success
---------------------------------------------------------------------------------------------
Return values:
[
    [
        "Alice",
        "Alice"
    ],
    [
        "Bob",
        "Cindy"
    ],
    [
        6000,
        4000
    ],
    [
        "20201207",
        "20201207"
    ],
    [
        "identified",
        "identified"
    ]
]
---------------------------------------------------------------------------------------------
```

```
[group:1]> call supplychain 0x3e46381f73b08fdf6fc1c207a62aeaeeaa4f422d selectFromReceipt Bob 0
---------------------------------------------------------------------------------------------
Return code: 0
description: transaction executed successfully
Return message: Success
---------------------------------------------------------------------------------------------
Return values:
[
    [
        "Bob"
    ],
    [
        "Cindy"
    ],
    [
        3000
    ],
    [
        "20201207"
    ],
    [
        "identified"
    ]
]
---------------------------------------------------------------------------------------------
```

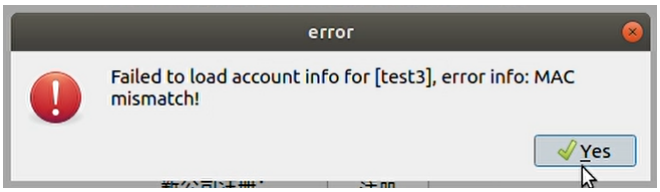
融资：



各种报错提示：

出现使用未认证的交易：

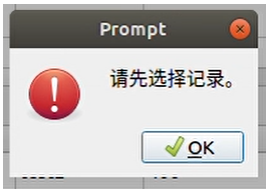
登录密码错误：



公司名称不存在：

不符合融资要求：

未选择记录：



四、心得体会

本次实验要求我们学习FISCO BCOS 和区块链相关知识，并完成一个从链端、后端到前端都很完整的项目，对我们来说算是一个不小的挑战。

因为专业是计科，所以之前并没有过任何后端、前端开发的学习与经历，刚着手的时候毫无头绪，所以后来这些部分的编程结构有参考网上的资料，特此说明。

链端的设计非常考验我们的细致全面的思维，因为这个有关数据库的内容，多个功能之间的联系可谓是牵一发而动全身，所以在编写和调试的过程中，通过解决各种各样的问题，我们也收获了很多。

最后，我们在分工合作方面也颇有感悟。因为本次区块链项目需要分工合作，这需要小组成员之间保持良好的交流，写代码时需要附上必要的注释，讲明函数的参数的含义、函数的功能以及函数返回值的含义，在小组成员遇到问题时，其他小组成员需要检查是否自己实现的部分出现的小问题导致该问题的发生。比如，在本次试验中，吴晓淳同学主要负责第二阶段链端的实现，但update函数的参数中的status_list和due_date_list顺序颠倒了，从而导致王若琪同学在实现后端时，会遇到status中的值和due_date中的值在调用update函数后顺序颠倒的问题，但在二人的检查下，终于找出问题并进行了纠正。

五、加分项

- 友好的用户操作界面。
- 鲁棒性，设置了各种错误提示机制，在操作错误的时候能够正确继续程序。