

基于区块链的供应链金融平台实验报告

姓名：谢俊杰 学号：18340181

一、背景

在传统供应链金融中，金融机构（银行）会根据对企业的信用评级，来评估企业的风险承担能力，从而决定能否批准企业的融资申请。对于供应链的上游企业，一般都能容易做出判断，然而对于处于下游的中小企业而言，金融机构则需要详细的信用分析以评估其还款能力，这个过程将增加很多经济成本，而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。

在区块链+供应链金融中，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，减少中小企业的融资难度。

二、方案设计

在该实验中，需要实现的功能如下：

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。该功能可以理解为上游企业和下游企业间签订的一个应收账款单据，即交易凭证。我我们也就需要在链上**增加**该单据凭证的各种信息记录。

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。该转让功能可理解为先将车企签订给轮胎公司的单据的当前剩余金额**更新**为减去转让部分的数值，再在链上**增加**车企签订给轮毂公司的转让部分的账款单据。

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。该融资功能需要银行等机构**查询**企业在链上的应收账款单据，再由银行视情况是否同意批准融资，后者与功能一的**增加**功能类似，也是在链上**增加**融资单据的记录。

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。该支付结算功能可理解为上游企业向下游企业支付相应的账款后，需要将单据凭证从链上**删除**。

FISCO BCOS提供合约CRUD接口开发模式，可以通过合约创建表，并对创建的表进行增删改查操作，将合约中涉及的数据存储在FISCO BCOS平台的AMDB的表结构中。而针对上述这些功能，我们需要设计一个注册企业的表 `t_company` 和一个记录账款单据的表 `t_receipt`。

而在后端调用链端的合约时，使用python-sdk的API `sendRawTransactionGetReceipt()`，收集相应的参数，根据合约地址和合约abi及接口名，发送交易并获取交易执行结果。

核心代码

企业登录：登录后可进行签发、转让、融资、支付这些功能的操作

后端代码

```
def signin_clicked(self):
    global cur_user
    cinfo_list = []
    with open('comp_info.csv', 'r', encoding = 'utf-8') as f:
        csv_file = csv.reader(f)
```


链端代码

```
//为企业在链上注册
function insert_company(string company_name) public returns(int256) {
    int256 ret = select_company(company_name);
    if (ret == 0) {
        Table t = openTable("t_company");
        Entry e = t.newEntry();
        e.set("status", "valid");
        e.set("name", company_name);
        int256 count = t.insert(company_name, e);
        emit RegisterResult(count, company_name);
        return count;
    }
    //企业已存在
    else {
        emit RegisterResult(0, "failed");
        return 0;
    }
}
```

签发应收账款：在链上增加该单据凭证的各种信息记录

后端代码

```
def sign_clicked(self):
    global cur_user
    to = self.lineEdit_to.text()
    amount = self.lineEdit_amount.text()
    deadline = self.dateEdit.date().toString('yyyy-MM-dd')
    args = [cur_user, to, int(amount), deadline]
    receipt = client.sendRawTransactionGetReceipt(address, contract_abi, 'sign',
args)
    if hex_to_signed(receipt['output']) == 0:
        QMessageBox.information(self, 'Error', 'Fail!', QMessageBox.Ok)
    else:
        QMessageBox.information(self, 'Hint', 'Sign successfully!',
QMessageBox.Ok)
    self.close()
```

链端代码

```
//签发应收账款单据
function sign(string from_str, string to_str, int256 total_amount_int, string
ddl_str) public returns(int256) {
    //双方企业是否存在
    int256 ret = select_company(from_str);
    if (ret == 0) {
        emit InsertResult(0, from_str, to_str, total_amount_int,
total_amount_int, ddl_str);
        return 0;
    }
    ret = select_company(to_str);
```

```

        if (ret == 0) {
            emit InsertResult(0, from_str, to_str, total_amount_int,
total_amount_int, ddl_str);
            return 0;
        }
        int256 count = insert(from_str, to_str, total_amount_int, total_amount_int,
ddl_str);
        return count;
    }
}

```

转让应收账款：将某笔拥有的应收账款单据部分或全部转让给其他公司

后端代码

```

def transfer_clicked(self):
    from_ = self.lineEdit_from.text()
    to = self.lineEdit_to.text()
    tot_a = self.lineEdit_total_a.text()
    cur_a = self.lineEdit_cur_a.text()
    trans_a = self.lineEdit_trans_a.text()
    deadline = self.dateEdit.date().toString('yyyy-MM-dd')
    args = [from_, cur_user, to, int(tot_a), int(cur_a), int(trans_a), deadline]
    receipt = client.sendRawTransactionGetReceipt(address, contract_abi,
'transfer', args)
    if hex_to_signed(receipt['output']) == -1:
        QMessageBox.information(self, 'Error', 'Fail!Transfer_amount is more than
cur_amount.', QMessageBox.Ok)
    elif hex_to_signed(receipt['output']) == 0:
        QMessageBox.information(self, 'Error', 'Fail!', QMessageBox.Ok)
    else:
        QMessageBox.information(self, 'Hint', 'Transfer successfully!',
QMessageBox.Ok)
    self.close()

```

链端代码

```

//转让
function transfer(string from_str, string to_str, string to_to_str, int256
total_amount_int, int256 cur_amount_int, int256 transfer_amount, string ddl_str)
public
returns(int256) {
    //双方企业是否存在
    int256 ret = select_company(from_str);
    if (ret == 0) {
        emit InsertResult(0, from_str, to_str, total_amount_int, cur_amount_int,
ddl_str);
        return 0;
    }
    ret = select_company(to_str);
    if (ret == 0) {
        emit InsertResult(0, from_str, to_str, total_amount_int, cur_amount_int,
ddl_str);
        return 0;
    }
}

```

```

int256 ret_code = 0;
int256 cur = cur_amount_int - transfer_amount;
//转让金额大于剩余金额
if (cur < 0) {
    ret_code = -1;
    return ret_code;
}
//转让后剩余金额为0则移除
else if (cur == 0) {
    remove(from_str, to_str, total_amount_int, cur_amount_int, ddl_str);
}
else {
    update(from_str, to_str, total_amount_int, cur, ddl_str);
}
ret_code = insert(from_str, to_to_str, transfer_amount, transfer_amount,
ddl_str);
return ret_code;
}

```

融资：向银行申请融资

后端代码

```

def apply_clicked(self):
    finance_amount = self.lineEdit.text()
    deadline = self.dateEdit.date().toString('yyyy-MM-dd')
    args = [cur_user, int(finance_amount), deadline]
    receipt = client.sendRawTransactionGetReceipt(address, contract_abi,
'finance', args)
    if hex_to_signed(receipt['output']) == 0:
        QMessageBox.information(self, 'Error', 'Fail!', QMessageBox.Ok)
    else:
        QMessageBox.information(self, 'Hint', 'Apply successfully!',
QMessageBox.Ok)
    self.close()

```

链端代码

```

//融资
function finance(string from_str, int256 total_amount_int, string ddl_str)
public returns(int256) {
    int256 ret = select_company(to_str);
    if (ret == 0) {
        emit InsertResult(0, from_str, "bank", total_amount_int,
total_amount_int, ddl_str);
        return 0;
    }
    int256 count = insert(from_str, "bank", total_amount_int, total_amount_int,
ddl_str);
    return count;
}

```

支付结算应收账款：上游企业向下游企业支付相应的账款后，需要将单据凭证从链上删除

后端代码

```
def pay_clicked(self):
    if self.tablewidget.selectionModel().hasSelection():
        row = self.tablewidget.currentRow()
        from_ = self.tablewidget.item(row, 0).text()
        to = self.tablewidget.item(row, 1).text()
        tot_amount = self.tablewidget.item(row, 2).text()
        cur_amount = self.tablewidget.item(row, 3).text()
        deadline = self.tablewidget.item(row, 4).text()
        args = [from_, to, int(tot_amount), int(cur_amount), deadline]
        receipt = client.sendRawTransactionGetReceipt(address, contract_abi,
"pay", args)
        QMessageBox.information(self, 'Hint', 'Pay successfully!', QMessageBox.Ok)
    else:
        QMessageBox.information(self, 'Hint', 'Please select a receipt.',
QMessageBox.Ok)
    self.refresh()
```

链端代码

```
//还款
function pay(string from_str, string to_str, int256 total_amount_int, int256
cur_amount_int, string ddl_str) public {
    remove(from_str, to_str, total_amount_int, cur_amount_int, ddl_str);
}
```

银行拒绝融资申请

后端代码

```
def refuse_clicked(self):
    if self.tablewidget.selectionModel().hasSelection():
        row = self.tablewidget.currentRow()
        from_ = self.tablewidget.item(row, 0).text()
        to = self.tablewidget.item(row, 1).text()
        tot_amount = self.tablewidget.item(row, 2).text()
        cur_amount = self.tablewidget.item(row, 3).text()
        deadline = self.tablewidget.item(row, 4).text()
        args = [from_, to, int(tot_amount), int(cur_amount), deadline]
        receipt = client.sendRawTransactionGetReceipt(address, contract_abi,
"remove", args)
        QMessageBox.information(self, 'Hint', 'Refuse successfully!',
QMessageBox.Ok)
    else:
        QMessageBox.information(self, 'Hint', 'Please select a receipt.',
QMessageBox.Ok)
    self.refresh()
```

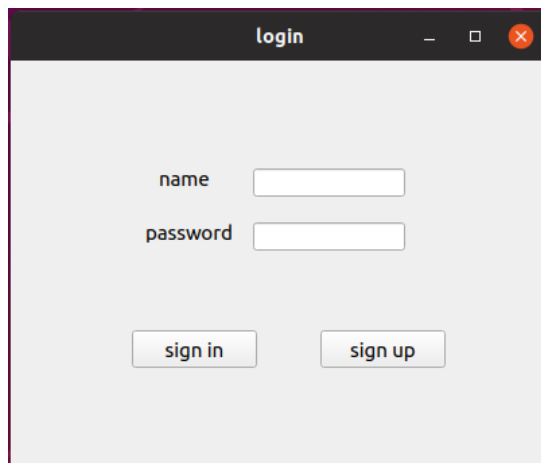
链端代码

```
//移除账款单据
function remove(string from_str, string to_str, int256 total_amount_int, int256
cur_amount_int, string ddl_str) public
```

```
returns(int256) {
    Table t = openTable("t_receipt");
    Condition c = t.newCondition();
    c.EQ("from", from_str);
    c.EQ("to", to_str);
    c.EQ("total_amount", total_amount_int);
    c.EQ("cur_amount", cur_amount_int);
    c.EQ("deadline", ddl_str);
    int256 count = t.remove("valid", c);
    emit RemoveResult(count, from_str, to_str, total_amount_int, cur_amount_int,
    ddl_str);
    return count;
}
```

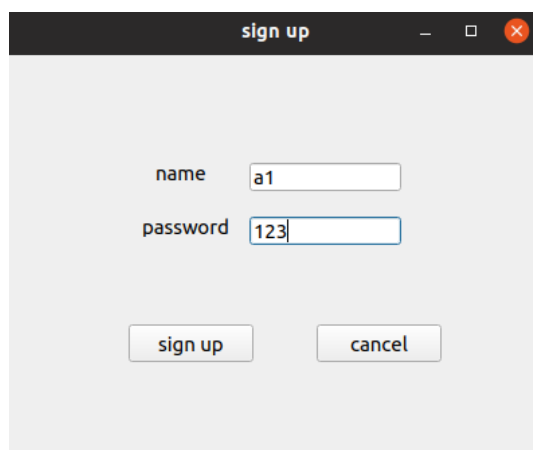
三、实验结果

运行程序，初始为登录界面：

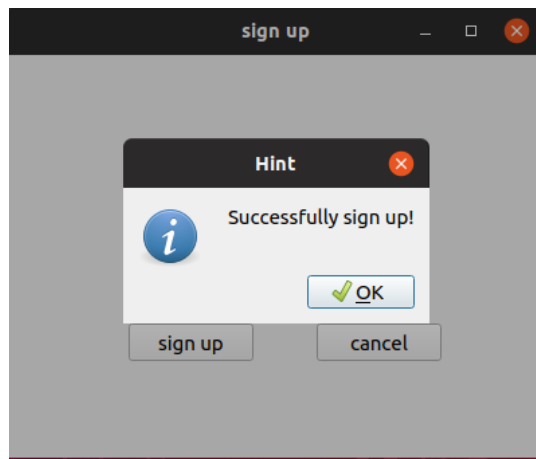


The screenshot shows a window titled "login" with a light gray background. It contains two text input fields: "name" and "password". Below the "password" field, there are two buttons: "sign in" and "sign up". The window has standard OS window controls (minimize, maximize, close) in the top right corner.

首先点击sign up，注册三个企业及银行金融机构，注册成功弹出提示：



The screenshot shows a window titled "sign up" with a light gray background. It contains two text input fields: "name" and "password". The "name" field contains the text "a1" and the "password" field contains the text "123". Below the "password" field, there are two buttons: "sign up" and "cancel". The window has standard OS window controls (minimize, maximize, close) in the top right corner.



登录a1，登录后界面：

A screenshot of a 'company' window titled 'receipt'. It features a table with five columns: 'from', 'to', 'total_amount', 'cur_amount', and 'deadline'. The table is currently empty. Below the table is a 'refresh' button. At the bottom of the window are four buttons arranged in a 2x2 grid: 'sign', 'transfer', 'finance', and 'pay'.

签发应收账款100给a2：

A screenshot of a 'company_sign' window titled 'receipt'. It features a table with five columns: 'from', 'to', 'total_amount', 'cur_amount', and 'deadline'. The table is empty. Below the table is a 'refresh' button. Below the 'refresh' button are three form fields: 'to' with the value 'a2', 'amount' with the value '100', and 'deadline' with the value '31/01/2021'. At the bottom are two buttons: 'sign' and 'cancel'.

回到主界面进行刷新即可看到具有相关信息的单据：

The screenshot shows a window titled 'company' with a 'receipt' section. Inside the receipt section, there is a table with the following data:

	from	to	total_amount	cur_amount	deadline
1	a1	a2	100	100	2021-01-31

Below the table is a 'refresh' button. At the bottom of the window, there are four buttons arranged in a 2x2 grid: 'sign', 'transfer', 'finance', and 'pay'.

退出a1的登录后，继续登录a2，同样可以看到与a2相关的单据：

This screenshot is identical to the one above, showing the 'company' window with the 'receipt' section. The table contains the same data row, and the buttons ('sign', 'transfer', 'finance', 'pay', 'refresh') are in the same positions.

a2转让账款40给a3：

company_transfer

receipt

	from	to	total_amount	cur_amount	deadline
1	a1	a2	100	100	2021-01-31

refresh

froma1transfer amount40

toa3deadline31/01/2021

total amount100

cur amount100transfercancel

回到主界面进行刷新即可看到剩余账款60的单据：

company

receipt

	from	to	total_amount	cur_amount	deadline
1	a1	a2	100	60	2021-01-31

refresh

signtransfer

financepay

退出a2的登录后，继续登录a3，同样可以看到与a3相关的单据：

company

receipt

	from	to	total_amount	cur_amount	deadline
1	a1	a3	40	40	2021-01-31

refresh

signtransferfinancepay

a3对银行进行申请融资30:

company_finance

receipt

	from	to	total_amount	cur_amount	deadline
1	a1	a3	40	40	2021-01-31

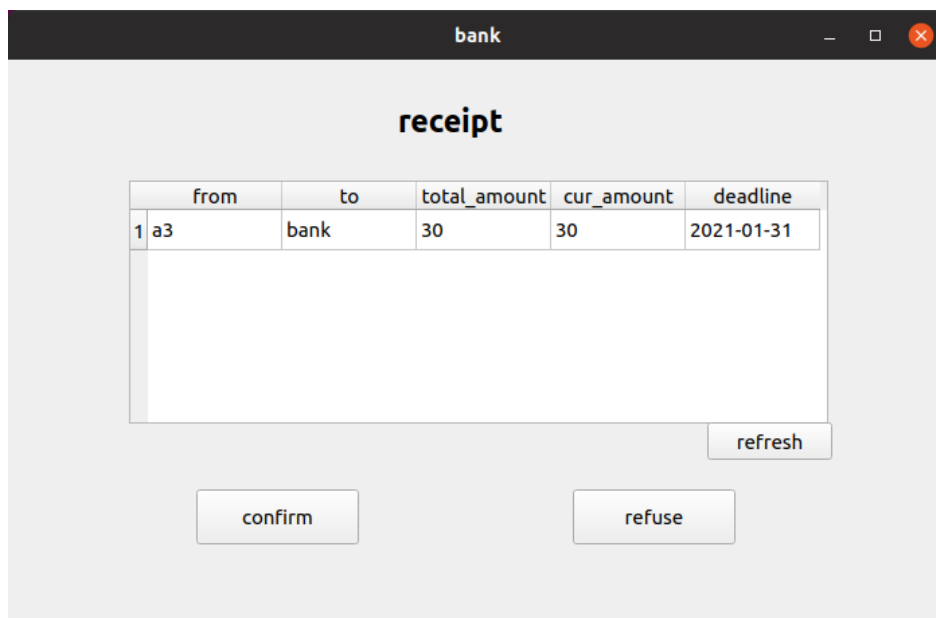
refresh

finance amount 30

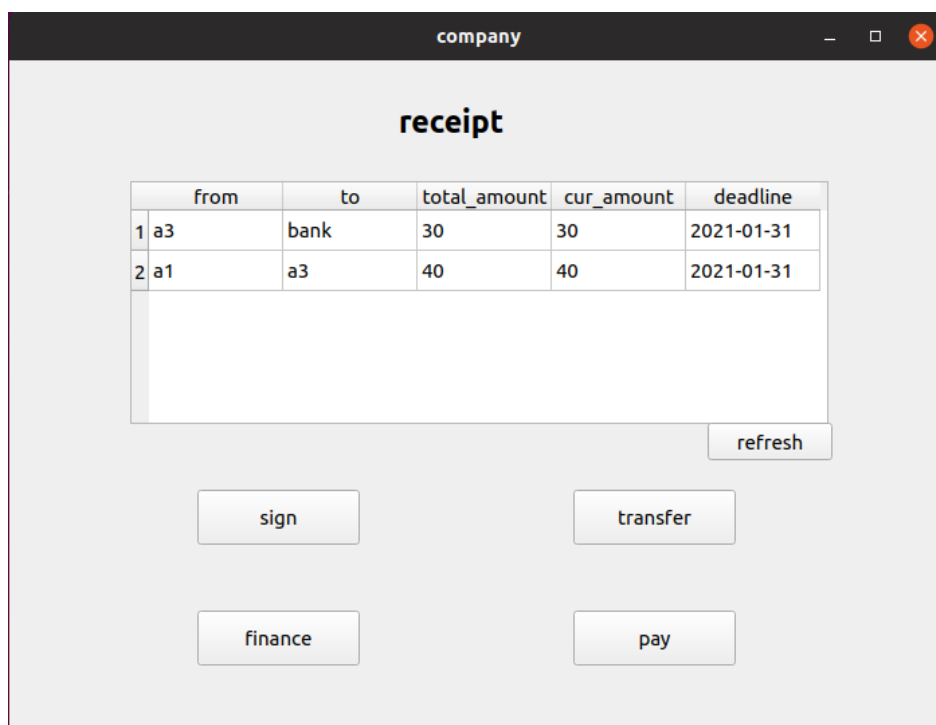
deadline 31/01/2021

applycancel

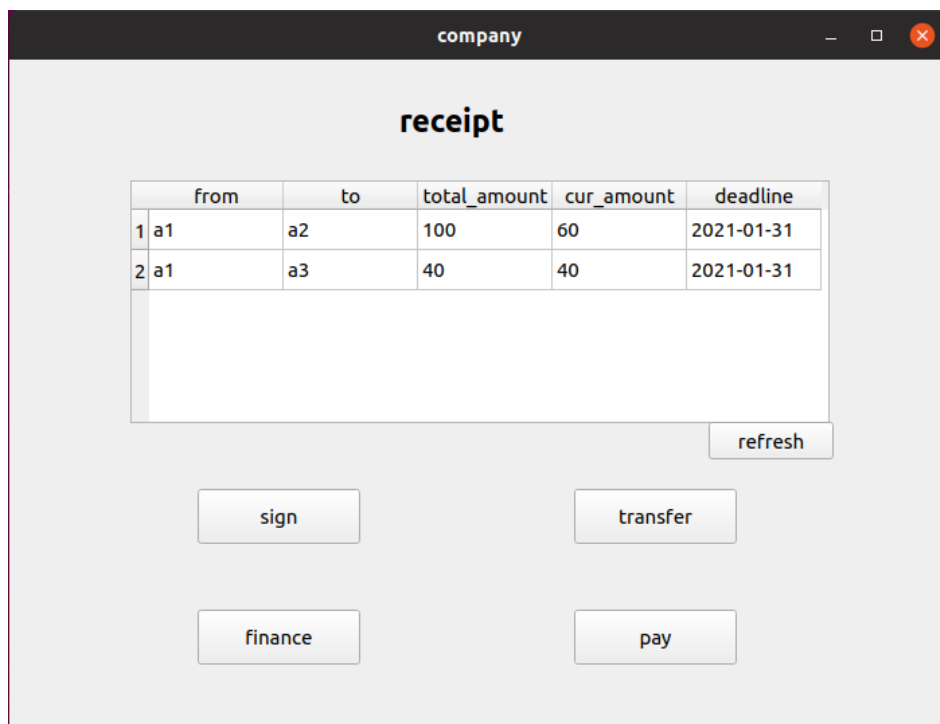
退出a3的登录后，继续登录银行账户bank，同样可以看到与bank相关的融资：



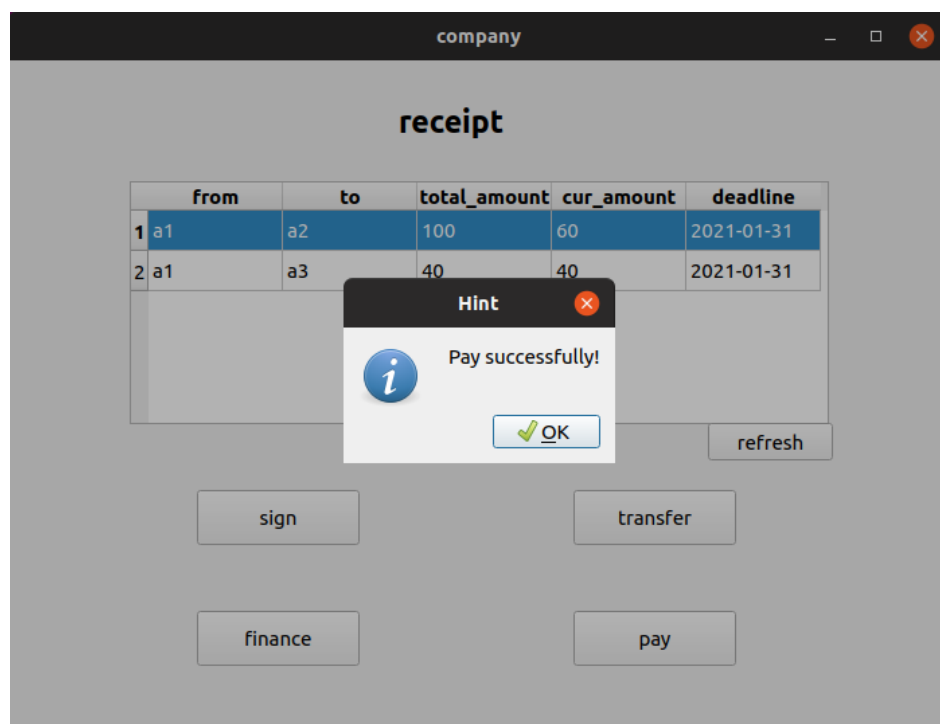
选中该单据后确认即可批准融资，再次登录a3，即可看到融资成功：

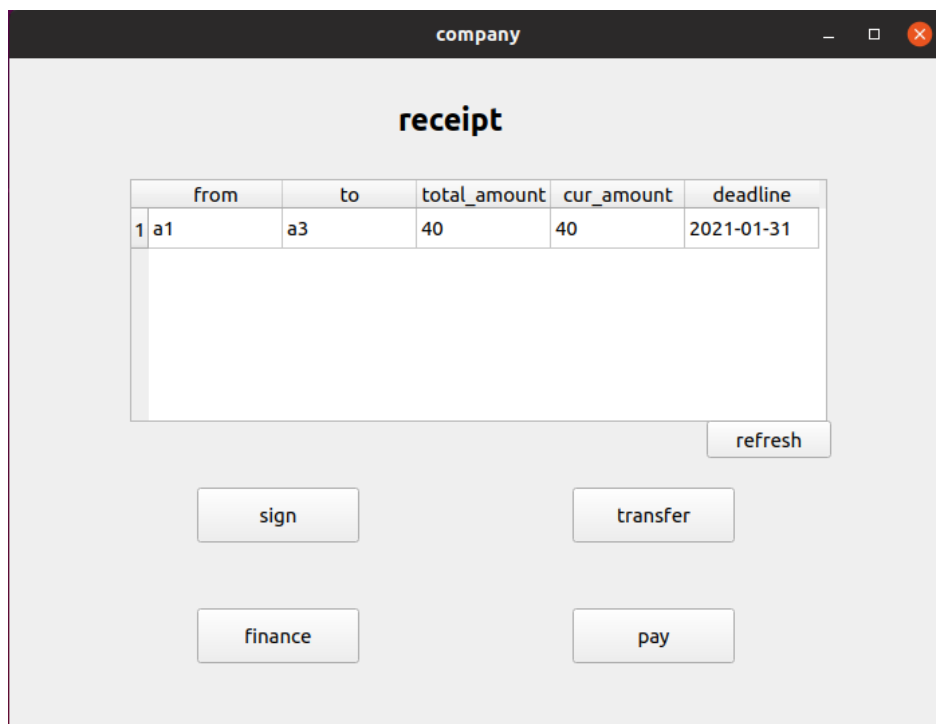


登录a1：



选择第1条单据，进行支付结算：





四、加分项

前端实现了友好高效的界面，并且在用户进行不当操作时能进行弹窗提示用户正确的操作方式，在操作成功时也会及时提示用户已操作成功，与用户有一定程度的友好交互，提高了效率。

五、实验心得

在这个基于区块链的供应链金融平台实验中，我学习到了使用已有的开源区块链系统FISCO BCOS完成私有链的搭建，在私有链上的操作使我回顾了课上学习到的区块链的共识机制、区块的解释等等，同时能够根据具体的供应链场景设计相应的智能合约并进行调用，也能够使用solidity对智能合约进行编写。在最终软件制品的任务中，一开始我是对怎样将后端和链端连接起来是比较困惑的，网上也难以找到相应的解释，感到无从下手，但是在仔细翻阅FISCO BCOS的技术文档后，发现在应用开发SDK中有类似的教程，最终我选择使用python SDK完成该任务，很多实现的方式都是在参照文档代码来实现的，所以在这次的实验中我也意识到在面对不太熟悉的技术时，文档是最好的老师。同时这次的大作业让我认识到类似FISCO BCOS的区块链在生活中应用的方式和场景，如供应链金融平台、交易的溯源、重要文件的管理等等，也意识到区块链蕴含的巨大的发展潜力。