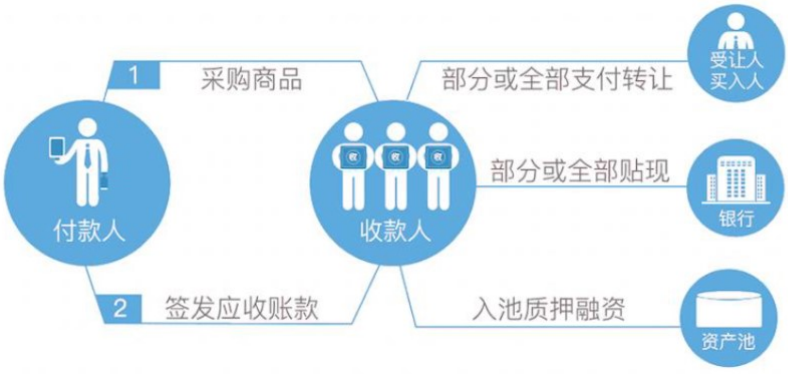
基于区块链的供应链金融平台报告

姓名	学号	专业
王广烁	18340165	18级网络空间安全
张洪宾	18340208	18级计科（超算）
孙新梦	18340149	18级计科（超算）

实验要求

> 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。

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。



传统供应链金融：

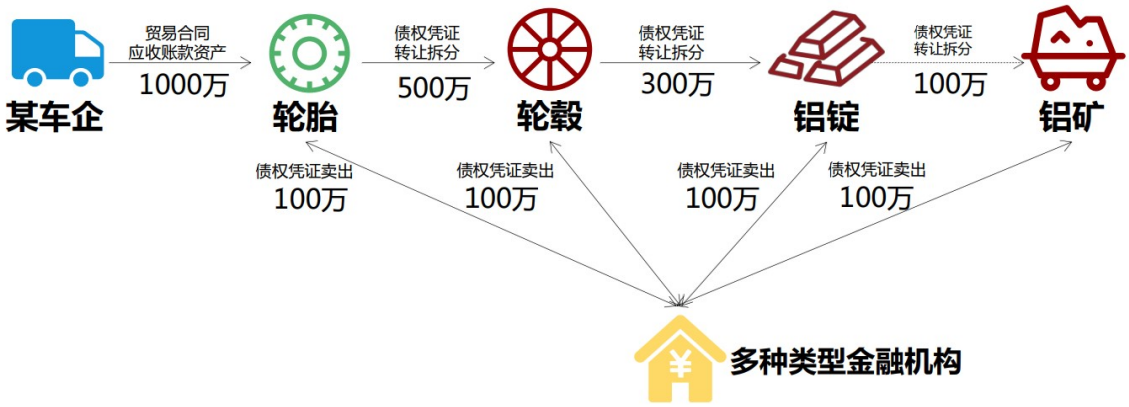
某车企（宝马）因为其造车技术特别牛，消费者口碑好，所以其在同行业中占据绝对优势地位。因此，在金融机构（银行）对该车企的信用评级将很高，认为他有很大的风险承担的能力。在某次交易中，该车企从轮胎公司购买了一批轮胎，但由于资金暂时短缺向轮胎公司签订了1000万的应收账款单据，承诺1年后归还轮胎公司1000万。这个过程可以拉上金融机构例如银行来对这笔交易作见证，确认这笔交易的真实性。在接下里的几个月里，轮胎公司因为资金短缺需要融资，这个时候它可以凭借跟某车企签订的应收账款单据向金融结构借款，金融机构认可该车企（核心企业）的还款能力，因此愿意借款给轮胎公司。但是，这样的信任关系并不会往下游传递。在某个交易中，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。



实验经过

设计思路

合约设计

我们使用内置的Table.sol来设计我们存储使用的表格

- 注册公司的表格
- 应收账款的表格

H5 注册公司的表格 t_company

注册公司的表格用来存放我们注册的公司信息，辅助实现登录的功能。存储的数据有名称，产品名称，企业资产

表项	类型	备注
company	string	企业名字（主键）
item	string	企业产品
balance	int256	企业资产

H5 应收账款的表格 t_receipt

这里的设计像是一个公有的账本，记录了所有的在本平台注册过的公司的账单，记录了谁欠谁多少钱，应该什么时候还清和账单状态

表项	类型	备注
key	string	对应区块的key(主键)
from	string	付款企业名称
to	string	收款企业名称
amount	int256	账款金额
due_date	string	还款截止日期
status	string	账单状态

转账

转账功能是本应用较为重要的基础功能。

- 当企业A资金充足的时候（产品总价小于等于资产一半）与企业B交易，购置B的产品的时候，支付过程为转账（与签下应收账款对比）
- 当A用应收账款向银行方申请融资时，银行发放贷款给A的过程也看作银行向A转账
- 当A支付欠B的应收账款的时候，也看作A向B转账

采购产品，新增应收账款

相对于A资金充足的时候（产品总价小于等于资产一半）直接转账购买产品，当A花费的金额超过自己资金balance的一半，此时A资金短缺，为了避免资金周转不灵，选择和B签订应收帐单，承诺未来某一天还给B。

具体操作为在`t_receipt`中新增一条记录如下

key	**from**	**to**	**amount**	**due_date**	**status**
...	A	B	$$$	还款日期	identified

状态“identified”表示银行证明这份交易。

转让应收账款

当企业B欠着A的钱，但是A想要从企业C购买其他产品的时候，也就是 t_receipt 中存放了from B to A和from A to C的记录的时候，并且A向C购买的总价小于B向A购买的总价，A企业可以选择把B对A的欠款抵作现金，转让这部分B将来要支付给A的钱给C，减少中间过程。

具体操作应该为删除from A to C的应收账款记录，增加下面的记录

key	from	to	amount	due_date	status
...	B	C	A欠C的金额	还款日期	identified

结算应收账款

当from企业资金充裕之后，在还款日期之前，表格 t_receipt 的from企业结算钱给to企业，调用transferbalance函数实现企业之间的转账，之后删去这条记录。

用应收账款向银行申请贷款

银行同样也可以作为from和to的对象，也就是我们需要在共有账本表格 t_receipt 上记录一条企业应该什么时候归还银行贷款的日期。

key	from	to	amount	due_date	status
...	A	BANK	$$$	还款日期	identified

银行认证

在我们设计的系统中，一条应收账款需要银行进行认证之后才算真的生效，把status字段从**identified**变成**authorized**才可以参与其他功能比如支付应收账款和转让应收账款的功能。

功能设计与实现

开发环境

- OS:MacOS
- 前端:python+Qt5
- IDE:QtCreator

注册用户

企业使用本应用的时候需要先进行注册,填上企业的名称,产品的名称以及目前的资金。

链端代码

在合约中我们设计注册这一函数,输入的参数是从用户界面输入绑定的企业名称,产品名称,企业资金,输出注册的结果代号

后端代码

后端代码是接受前端给出的绑定参数,调用合约的函数,完成主要逻辑的代码部分

针对我们的注册功能,设计函数press_register,输入前端绑定的企业名称,密码,读入的三个text文段,之后判断名字是否不合法,若是合法就判断输入的账号密码是否正确,若是正确则调用链端函数注册企业,这个过程中可以输出日志方便调试

前端代码

设计点击注册按钮到注册页面，在输入框输入用户名和密码点击确认注册，把读入的用户名和密码绑定传给后端

查询应收账款

企业可以按照自己的名称和密码来寻找自己的账单, 应收账款的记录,按照格式显示给用户

链端代码

链端保存的表格 t_receipt 可以用from字段来查询和某企业相关的应收账款

输入string类型的企业名称,根据mode返回对应记录

后端代码

后端设计函数调用链端函数,主要是接受前端绑定的企业名称,之后判断名称是否合法,若是合法则调用链端函数,传入名称,查询相关结果

前端代码

点击查询债务的tab，显示总债务数目，总债权数额，下面用表格形式显示债务和债权情况

下单商品，新增应收账款

当企业采购另一个企业的产品,并且总金额大于自己的一般资产的时候,就需要新增应收账款,算作欠条,承诺from企业在某期限之前归还to企业多少钱

链端代码

链端函数输入from,to,amount_of_money,截止日期四个信息,首先判断from和to的企业在注册表里是否找得到,然后判断总价格是否大于from资产的一半,如果没有超过一半,就算作转账,调用转账函数,否则应收账款表格新增一条表的记录

后端代码

从前端得到用户信息,提取信息组成参数,调用智能合约的函数,做出相应的显示.

前端代码

设计输入框输入卖家名称，金额，付款的期限，收集交易的信息,如从谁那里买多少东西，绑定传给后端

清算应收账款

当企业资金充足时,在期许的期限之前归还从前的欠款

链端代码

清算应收账款,支付欠款

输入可以搜索到应还账款表的信息,from,to,账款金额,还款日期,返回结算结果(0为成功,1为失败)

根据信息找到对应表的记录并删除

后端代码

得到前端用户的选择(相应的应还账款记录),调用智能合约函数,做出结果的展示.若是没有认证或者没有选择记录需要进行提醒

前端代码

界面让用户能够点击选择自己应还的账款记录，然后设计按钮点击确认还款,调用后端函数消除此条记录

转让债务

当一个企业购买产品时,可以用另一个企业欠他的钱来抵换成购买新产品的现金,这样可以减少表中的记录数目,让欠他钱的企业直接还款到新购买产品的企业上去

链端代码

用来转让应收账款

输入买方企业,卖方企业,第三方企业,收款金额,返回转让的结果.

返回值	意义
0	成功
-1	买方企业不存在
-2	卖方企业不存在
-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

函数在判断输入合法之后,会插入一条新的记录从第三方企业到卖方的应还记录

后端代码

后端得到前端用户的输入和选择,（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）中的函数，并作出相应的显示

前端代码

前端接受并绑定用户的输入和选择。从债务（别人欠该企业）和债权（该企业欠别人）各选择一个记录，输入数额和到期日，点击提交就可以转让债务。绑定传给后端。

申请贷款

供应链上所有企业都可以用营收账单向银行申请贷款解决资金周转问题.

链端代码

输入参数为申请企业名称,申请金额,还款日期,因为所有的另一方都是银行,所以不需要记录另一方的信息

用的是转账的函数实现,实现从bank到该企业的转账，最后输出贷款结果代号

后端代码

从前端得到用户的输入选择,调用智能合约函数,根据返回结果做出相应的显示和提示.如贷款成功或者时未认证等错误

前端代码

设计选择框和输入框和按钮来让用户选择贷款并确认,调用后端函数并接受结果,进行展示

功能测试

1.编译运行

启动节点 `bash nodes/127.0.0.1/start_all.sh`

将 **source** 文件夹内容放入配置好的python SDK 文件夹中，直接运行 **main.py**。部署方法参考Python SDK

运行 **main.py** 之后需要先进行账户的注册, 之后再进行转账等交易工作。具体可以参考演示视频中的内容

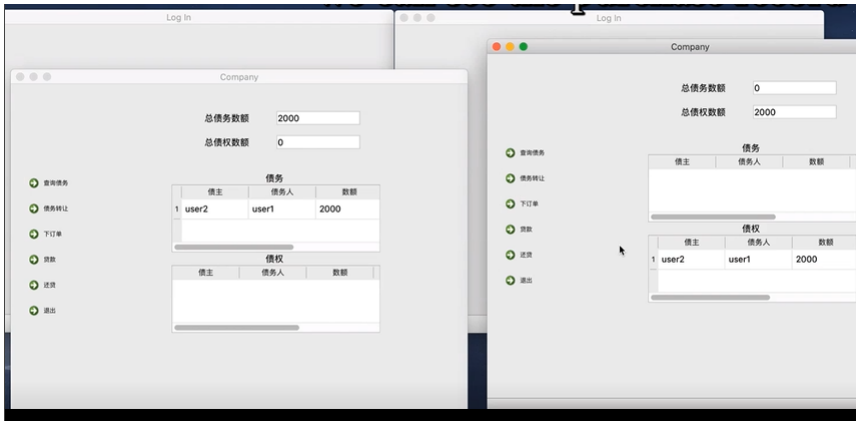
2.注册功能：

运行应用后出现登录/注册界面，当新用户头一次使用的时候，需要注册，输入设定的用户名和密码，店家确定注册，之后再用这一组用户名和密码进行登录。

登陆后可以看到初始界面，在这个界面左边可以选择不同功能——

3.查询债务

点击左侧查询债务，可以看到债务（谁欠我钱）和债权（我欠谁钱）

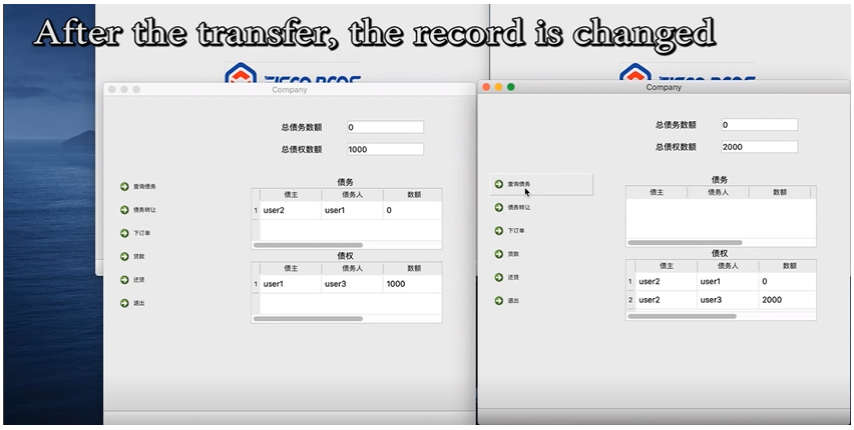


4.债务转让

从债务（别人欠该企业）和债权（该企业欠别人）各选择一个记录，输入数额和到期日，点击提交就可以转让债务。

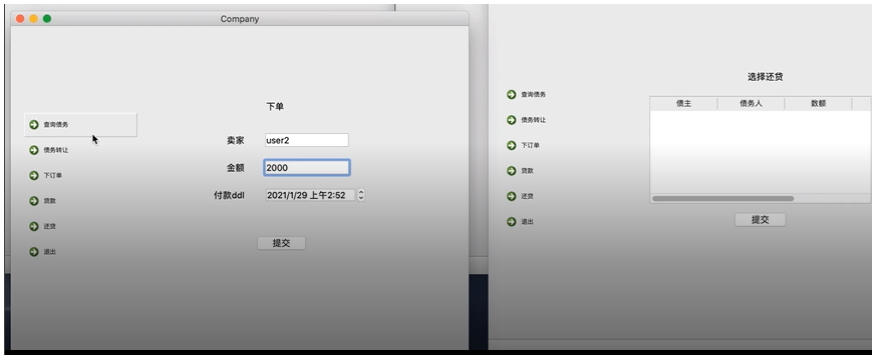
从user1要购买user3的产品，用user2欠user1的债务抵押2000元，最后只有user2欠user32000元。

可以看到在转让之后，记录的变化：

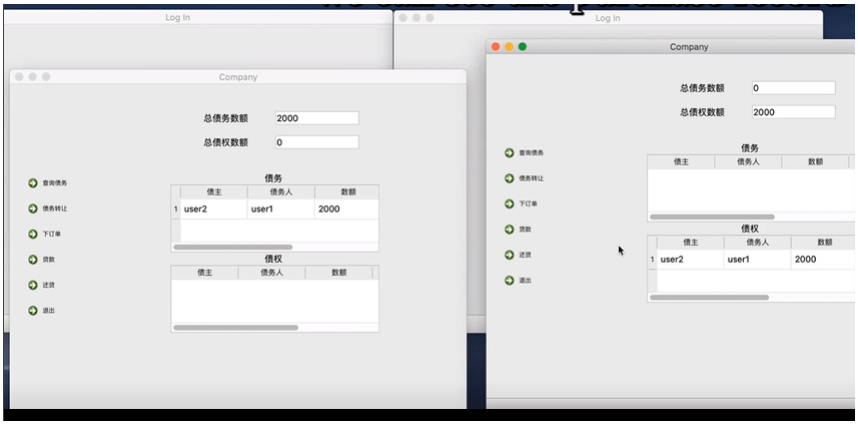


5.下订单

企业购买产品下订单，购买user2的2000元产品

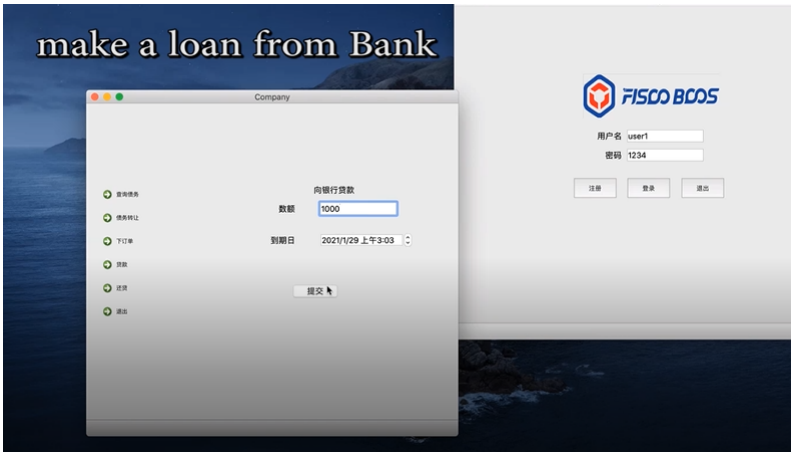


之后查看债务表，发现出现此债务



6.贷款

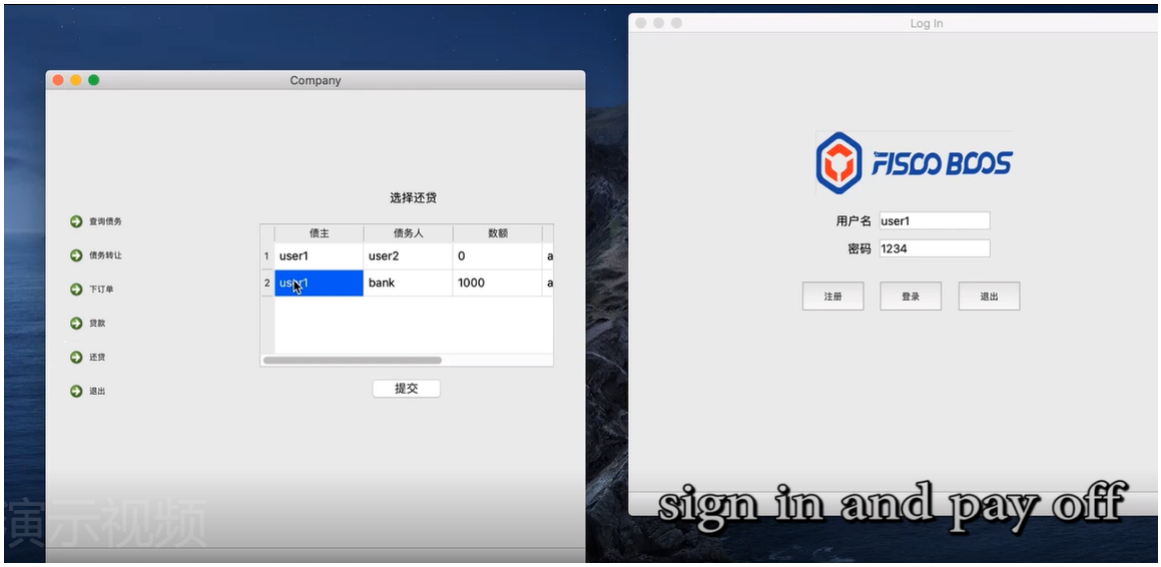
贷款相当于从银行那里借债务，点击贷款，输入贷款金额点击提交



之后查看自己的债权表格发现出现此条记录：

7.还款

还款时点击还款tab，之后点击需要还得贷款，点击提交则可还清债务



实验总结

1.实验分工

姓名	分工
王广烁	前端链端设计，剪辑视频
张洪宾	前端后端设计，录制视频
孙新梦	链端项目设计，实验报告

2.技术亮点

- 基于MacOS解决了部分不能运行的问题
- 友好的操作界面，设计良好，高效引导用户使用
- 鲁棒的程序设计，考虑到诸多错误（如未注册，无记录),设置错误提示机制

3.实验感想

本次实验实现了一个有友好用户界面,较为鲁棒的基于区块链的供应链系统,从前端的设计到后端的逻辑,再到链上函数的逻辑无一不是需要我们细细斟酌慢慢修改的.

我们遇到了许多开发上的问题,一是寒假在家沟通的不便利,我们使用GitHub协同开发,增加沟通频率和深度,让我们三个人都收获颇多.其二是关于前端的实现,由于之前不大接触过这样的前端开发,从零开始入门会较为吃力,最后使用QtCreator完成前端开发,前端后端这样分层的开发是目前的主要应用开发模式,之后我们需要重点去学习的开发方式.其三是在测试过程中发现很多不够鲁棒的地方, 比如判断一个企业是否已经注册过,还有是否这个应收账款等,这些判断都需要在测试的过程中慢慢加入,完善合约的内容.

经过这一次项目设计,我们发现要实现一个可用的系统并不容易,前端像是皮肉,捏捏换换形状了,看似随便点击一下出现各式各样的信息,其实都是后端的骨架和链端的血脉在运作的结果. 尤其是设计后端哪些部分调用哪些,怎样调用之类的,绑定到前端是怎样的效果,都是需要小组成员细细斟酌好好思考的.

区块链使得这样的函数上链称为合约,区块链的不可篡改性使得和金融相关的应用安全性有了保障,每一笔交易都不可篡改,不会出现金融的骗局,我们认为在未来的社会,区块链去中心化,没有中心的权威是有长足的发展潜力的,相信未来基于区块链的应用也将从如今的凤毛麟角到妇孺皆知. 所谓玉不琢不成器,我们相信这一新兴的技术只要好好加以利用,有更多人能够投入时间精力去加以研究和完善,便是为人类带来新的便利的好技术.

参考

http://fisco-bcos.org/zh/assets/docs/FISCO%20BCOS%20-%20Featured%20Cases-Cn.pdf

https://github.com/FISCO-BCOS/python-sdk

https://www.cnblogs.com/xsmile/p/13491508.html

https://www.cnblogs.com/xsmile/p/13581425.html