

# 中山大学数据科学与计算机学院本科生实验报告

## (2019 年秋季学期)

课程名称：区块链原理与技术

任课教师：郑子彬

年级	2017	专业（方向）	软件工程
学号	17343100	姓名	仕润昊
电话	13280152626	Email	1056627011@qq.com
开始日期	2019/12/01	完成日期	2019/12/10

### 一、项目背景

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 1 000 万的应收账款单据，承诺 1 年后归还轮胎公司 1 000 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。

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。

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。

## 二、 方案设计

我设计了一套供应链体系，将供应链上的每一笔交易和收款单据上链，引入了第三方信任机构来确认交易，确保交易和单据的真实性。为了减少中小企业的融资难度，每一笔应收账款可以进行转让，使得核心企业的信用可以传递到供应链的下游企业。

### （一）存储设计

我的供应链智能合约的实现是基于表（table）存储的。table 类似于传统的数据库使用，业务逻辑开发较为简单，管理更为方便，性能也更高。而表的存储本质上不会改变区块链的去中心化特征，且有更好的性能，使得开发更简单方便。在本次开发中，我使用了 fisco-bcos 提供的 Table.sol 的接口来实现对数据库表的操作。

#### 注册公司表

在智能合约的实现中，我们使用一个 `company_reg` 表记录所有注册公司，所有交易的欠款人和借款人必须在这个 `company_reg` 表中，否则视为非法交易。当一个新公司想要借款时，需要先在 `company_reg` 表中注册，注册内容包括公司的名称、地址、类型。

企业注册, key : comname, field : comaddress, comkind		
企业名称(主键)	企业地址	企业类型
-----	-----	-----
comname	comaddress	comkind
-----	-----	-----

根据上述分析创建注册公司表 `company_reg`。

```
TableFactory tf = TableFactory(0x1001); // the address of TableFactory is 0x1001
// company table [key: comname]
tf.createTable("company_reg", "comname", "comaddress, comkind");
```

注册公司需要输入公司的名称、地址和类型，并且要求公司不能重名。每次注册时都将进行唯一性检查。通过 Table.sol 提供的 `select` 接口和 `insert` 接口完成注册公司名称的唯一性检查和新公司的注册。

```
function checkCompanyName(string comname) private constant returns(int256) {
    // open table
    Table table = openCompanyTable();
    // query
    Entries entries = table.select(comname, table.newCondition());
    if (0 == uint256(entries.size())) {
```

```

        return -1;
    } else {
        return 0;
    }
}

/*
description : register a company
parameters :
    comname : company name
    comaddress : company address
    comkind : company kind
return values:
    0 register success
    -1 account exists
    -2 other errors
*/
function registerCompany(string comname, string comaddress, string comkind) public
returns(int256) {
    int256 ret_code = 0;
    int256 ret = 0;
    // check whether the company exists
    ret = checkCompanyName(comname);
    // if not exists
    if(ret != 0) {
        Table table = openCompanyTable();
        Entry entry = table.newEntry();
        entry.set("comname", comname);
        entry.set("comaddress", comaddress);
        entry.set("comkind", comkind);
        // insert
        int count = table.insert(comname, entry);
        if (count == 1) {
            // success
            comname_arr[cnt] = comname;
            cnt++;
            ret_code = 0;
        } else {
            // fail
            ret_code = -2;
        }
    } else {
        // if exists or company not exists
        ret_code = -1;
    }

    emit RegisterEvent(ret_code, comname, comaddress, comkind);
    return ret_code;
}

```

## 交易信息表

如果已经注册的 A 公司向已经注册的 B 公司借款 100 元，则将该交易记录到链上，该交易记录字段包括：欠款人、借款人、借款数额、借款日期、应还日期、状态（即是否得到银行确认）。因为 Table.sol 提供的所有操作都是基于唯一的 key，所以为了便于以后的操作，我们实际建立两张完全相同的表（主码不同）：一张是以借款公司（欠款人） comname\_key\_from 为主码的表，另一张是以贷款公司（借款人） comname\_key\_to 为主码的表。

```
receipt_key_from, key : comname_from, field : comname_to, amount, start_date, end_date, status
| 欠款人(主键) | 借款人 | 金额 | 借款时间 | 还款时间 | 状态 |
|-----|-----|-----|-----|-----|-----|
| comname_from | comname_to | amount | start_date | end_date | status |
|-----|-----|-----|-----|-----|-----|
```

根据上述分析创建两张表 receipt\_key\_from 和 receipt\_key\_to。

```
tf.createTable("receipt_key_from", "comname_from", "comname_to, amount, start_date, end_date, status");
// receiptto Table [key comname_to]
tf.createTable("receipt_key_to", "comname_to", "comname_from, amount, start_date, end_date, status");
// the receiptfrom and receiptto are the same, the key of the two tables are different
```

每次收据记录都需要上述所有字段，不允许有前五个属性相同的收据（视为同一收据），同时在创建收据前也会检查欠款人和借款人公司的合法性。如果该收据经过第三方权威机构（如银行）的确认，该信息将记录到 status 属性中，便于以后的信誉证明。

```
function addReceipt(string comname_from, string comname_to, int amount, int start_date, int end_date, string status) private returns (int256){
    int256 ret_code = 0;
    int256 ret = 0;
    // check whether the receipt exists
    ret = checkReceipt(comname_from, comname_to, amount, start_date, end_date);
    // if not exists
    if(ret != 0) {
        Table tablefrom = openReceiptFromTable();
        Table tableto = openReceiptToTable();
        Entry entry = tablefrom.newEntry();
        entry.set("comname_from", comname_from);
        entry.set("comname_to", comname_to);
        entry.set("amount", amount);
        entry.set("start_date", start_date);
        entry.set("end_date", end_date);

        // insert
        int count1 = tablefrom.insert(comname_from, entry);
        int count2 = tableto.insert(comname_to, entry);
        if (count1 == 1 && count2 == 1) {
            // success
            ret_code = 0;
        } else {
```

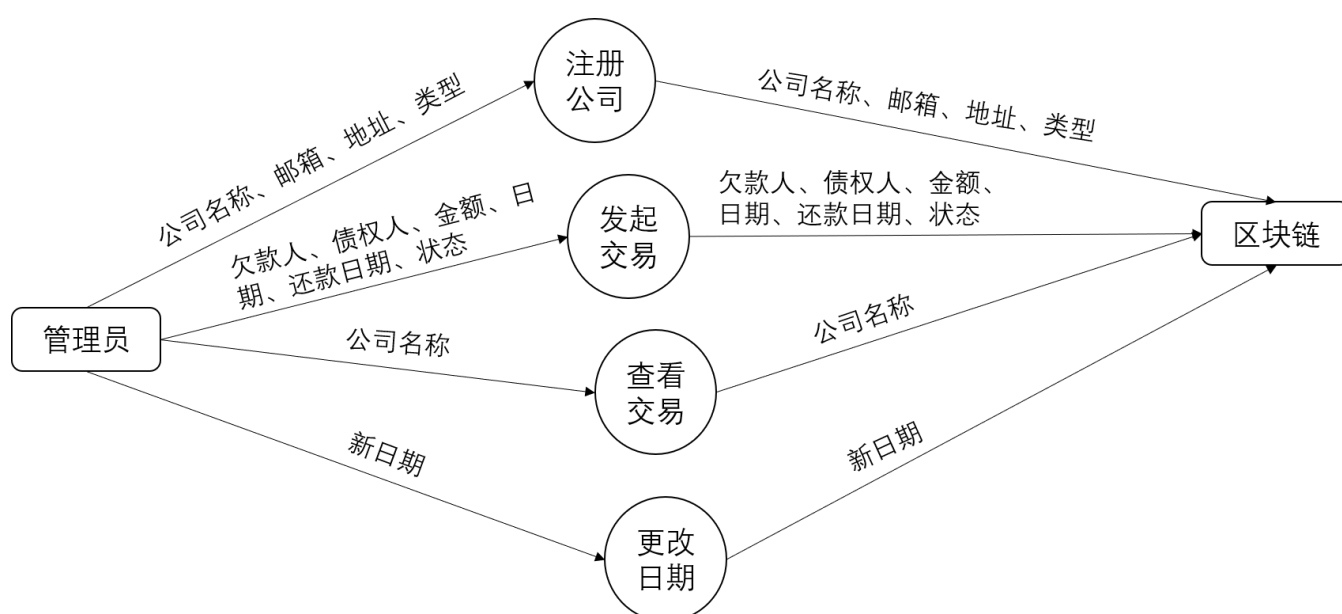
```

        // fail
        ret_code = -2;
    }
} else {
    // if exists
    ret_code = -1;
}

emit TransferEvent(ret_code, comname_from, comname_to, amount, start_date, end_date,
status);
return ret_code;
}

```

## (二) 数据流图



## (三) 核心功能介绍（文字+代码）

### 1. 链端实现

**核心功能一：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。

为了便于解决中小企业融资难的问题，需要将借贷的信用进行传递，这也就意味着，如果 B 向 C 借款 200 元，A 向 B 借款 100 元，这两个收据可以等价转换为 B 向 C 借款 100 元，A 向 C 借款 100 元。

当然，这个在实际操作中也会产生许多问题：比如还款日期的统一。在我的供应链实现中，要求收据转移时，还款日期为两个还款日期中最早的那一个。即如果 B 向 C 借款 200 元，还款日期为 10 号，A 向 B 借款 100 元，还款日期为 8 号时，收据等价转换后，B 向 C 借款 100 元，还款日期不变为 10 号，A 向 C 借款 100 元，还款日期为最近的那一个即 8 号。

如果出现 B 向多个企业借款，且 A 的借款总额总是大于某个企业时，该收据将分散到多个 B 的上游贷款企业。即 A 的借款人变为 B 的多家上游贷款企业。

这样的操作使得每个链式关系的最大节点数为 2，实际也是并查集中的路径压缩操作。

具体代码如下所示：

```
/*
description : transfer a receipt
parameters :
    comname_from : borrower name
    comname_to   : creditor name
    amount       : amount
    start_date   : borrow time
    end_date     : return time
    status       : whether recognized by bank
return values:
    0 register success
    -1 account exists
    -2 other errors
*/

function transferReceipt(string comname_from, string comname_to, int amount, int start_date, int
end_date, string status) public {
    Table tablefrom = openReceiptFromTable();
    Entries entriesto = tablefrom.select(comname_to, tablefrom.newCondition());
    Entry tmp;
    int new_end_date;
    int tmp_new_amount;
    for(int i = 0; i < entriesto.size(); i++) {
        if(amount == 0) {
            break;
        }
        tmp = entriesto.get(i);
        new_end_date = tmp.getInt("end_date") < end_date ? tmp.getInt("end_date") : end_date;
        if(tmp.getInt("amount") == amount) {
            // earliest end_date
            addReceipt(comname_from, tmp.getString("comname_to"), amount, start_date,
new_end_date, status);
            break;
        }
        else if(tmp.getInt("amount") > amount) {
            tmp_new_amount = tmp.getInt("amount") - amount;
            updateReceipt(tmp.getString("comname_from"), tmp.getString("comname_to"),
tmp.getInt("amount"), tmp.getInt("start_date"), tmp.getInt("end_date"), tmp_new_amount,
tmp.getInt("end_date"));
        }
    }
}
```

```

        addReceipt(comname_from, tmp.getString("comname_to"), amount, start_date,
new_end_date, status);
        break;
    }
    else {
        amount = amount - tmp.getInt("amount");
        addReceipt(comname_from, tmp.getString("comname_to"), tmp.getInt("amount"),
start_date, new_end_date, status);
        deleteReceipt(tmp.getString("comname_from"),tmp.getString("comname_to"),tmp.getInt("a
mount"),tmp.getInt("start_date"),tmp.getInt("end_date"));

    }
}
if(amount > 0) {
    addReceipt(comname_from, comname_to, amount, start_date, end_date, status);
}
}

```

## 核心功能二：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。

因为所有的收据不存在链式关系，所以对于每个企业，只需要向银行展示所有的借贷信息即可。我设计的供应链除了返回借贷信息之外，另外增加了一个字段记录该企业的借贷是处于“正资产”还是“负资产”，也即，正资产代表借款总额大于欠款总额，负资产代表欠款总额大于借款总额。

```

/*
description : query a company's receipts
parameters :
    comname : company name
return values:
    Entries entriesfrom : borrower receipts
    Entries entriesto  : creditor receipts
    int amount        : amount
*/
function queryReceipt(string comname) public returns (Entries, Entries, int){
    Table tablefrom = openReceiptFromTable();
    Table tableto = openReceiptToTable();
    Entries entriesfrom = tablefrom.select(comname, tablefrom.newCondition());
    Entries entriesto = tableto.select(comname, tableto.newCondition());
    int amount = 0;
    for(int i = 0;i < entriesfrom.size();i++) {
        amount = amount - entriesfrom.get(i).getInt("amount");
    }
    for(int j = 0;j < entriesto.size();j++) {
        amount = amount + entriesto.get(j).getInt("amount");
    }
    return (entriesfrom, entriesto, amount);
}

```

**核心功能三：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。**

当到了还款日期时，企业应该及时还款，所有的到期账单都应该到期销毁。我的供应链通过调用 `addDay` 函数模拟日期增加，当到达一定时间后，到期的账单都将被销毁，核心企业向下游企业支付相应的欠款。

```
/*
description : remove a receipt
parameters : comname : company name
return values
*/
function removeReceiptDate(string comname) private {

    Table tablefrom = openReceiptFromTable();
    Table tableto = openReceiptToTable();
    Condition condition = tablefrom.newCondition();
    condition.LT("end_date", current_date);
    tablefrom.remove(comname, condition);
    tableto.remove(comname, condition);
}

/*
description : add current_date
parameters
return values
*/
function addDay(int new_days) public returns(int){
    current_date = current_date + new_days;
    for(uint k = 0; k < cnt; k++) {
        removeReceiptDate(comname_arr[k]);
    }
    return current_date;
}
```

## 2.后端实现

我采用 Fsico-bcos 提供的 Node.js API 来进行后端的开发。配置完毕环境、证书及 Channel 端口后，使用 Node.js 进行后端开发。



## 调用 Node.js API

首先使用官方提供的编译方法将智能合约编译成 abi 以备后用，然后使用 cli.js 将智能合约部署到链上。

```
./cli.js deploy SupplyChain
```

在 main.js 中使用官方提供的 Node.js API，如下：

```
var Web3jService = require('./nodejs-sdk/packages/api/web3j').Web3jService;
var api = new Web3jService();
```

以注册公司为例，使用官方 API 的调用智能合约的方法如下：

```
app.get('/company_get', function (req, res) {
  var response = {
    "company_name": req.query.company_name,
    "company_address": req.query.company_address,
    "company_kind": req.query.company_kind
  };
  var params=[req.query.company_name,req.query.company_address,req.query.company_kind];
  // [ 'transferReceipt', 'addDay', 'queryReceipt', 'registerCompany' ]
  var functions = abi.filter(value => value.type === 'function').map(value => value.name);
  var functionName = utils.spliceFunctionSignature(abi[3]);
  // send a transcation
  api.sendRawTransaction(SupplyChainAddr,functionName,params).then(function(value) {
    console.log(value);
  });
  ...
});
})
```

## 调用 CRUDService

在后端，我主要想通过 CRUD 接口查询表格。因为官方提供的 CRUD 接口并不好用，而我后端的主要操作是查表，所以我对其 select 函数进行修改和重构。

我使用了 cli 文件目录下的 crud 的 interfaces。

```
var crud = require('./nodejs-sdk/packages/cli/interfaces/crud').interfaces;
let crudService = new CRUDService();
```

根据 cli 目录下的调用方法修改 select 函数，该函数将根据表名、键值和可选项值返回一个 Promise 对象，解析该对象就可以获得查询到的结果。

```
function select(argv) {
  let tableName = argv.tableName;
  let key = argv.key;
  let condition = parseCondition(argv.condition);
  return crudService.desc(tableName).then(tableInfo => {
    let table = new Table(tableInfo.tableName, key, tableInfo.valueFields, tableInfo.optional);
    return crudService.select(table, condition);
  });
}
```

## 后端与链端、前端交互

以记录一条交易为例，展示供应链的后端与链端、前端的交互过程。当交易内容填完后，用户点击网页上的 `Finish` 提交交易。此时后端收到前端发送的 GET 请求以及相应的表格参数，进行处理。后端首先利用之前写好的 `select` 函数查询数据库，查询该数据库中是否已经存在相同交易，如果不存在这样的交易，就调用链上的智能合约接口，发送交易，记录该条交易，并检测该交易是否部署到链上。如果该交易成功部署到链上，就向前端发送交易信息；如果失败则向前端发送 `error` 信息。这样就完成了供应链的后端与链端、前端的一次交互。

```
app.get('/receipt_get', function(req,res) {
  // 输出 JSON 格式
  var response = {
    "comname_from":req.query.comname_from,
    "comname_to":req.query.comname_to,
    "amount":req.query.amount,
    "start_date":req.query.start_date,
    "end_date":req.query.end_date,
    "status":req.query.status
  };

  var start_date = DateDiff(BaseTime,req.query.start_date);
  var end_date = DateDiff(BaseTime,req.query.end_date);
  var
  params=[req.query.comname_from,req.query.comname_to,req.query.amount,start_date,end_date,req.query.status];
  var functions = abi.filter(value => value.type === 'function').map(value => value.name);
  var functionName = utils.spliceFunctionSignature(abi[0]);

  var argv = {
    "tableName":"receipt_key_from",
    "key": req.query.comname_from,
    "condition": "comname_to="+ req.query.comname_to
  }
  var table_value = select(argv);
  var table_len;
  var flag = true;
  table_value.then(function(ret) {
    table_len = ret.length;
```

```

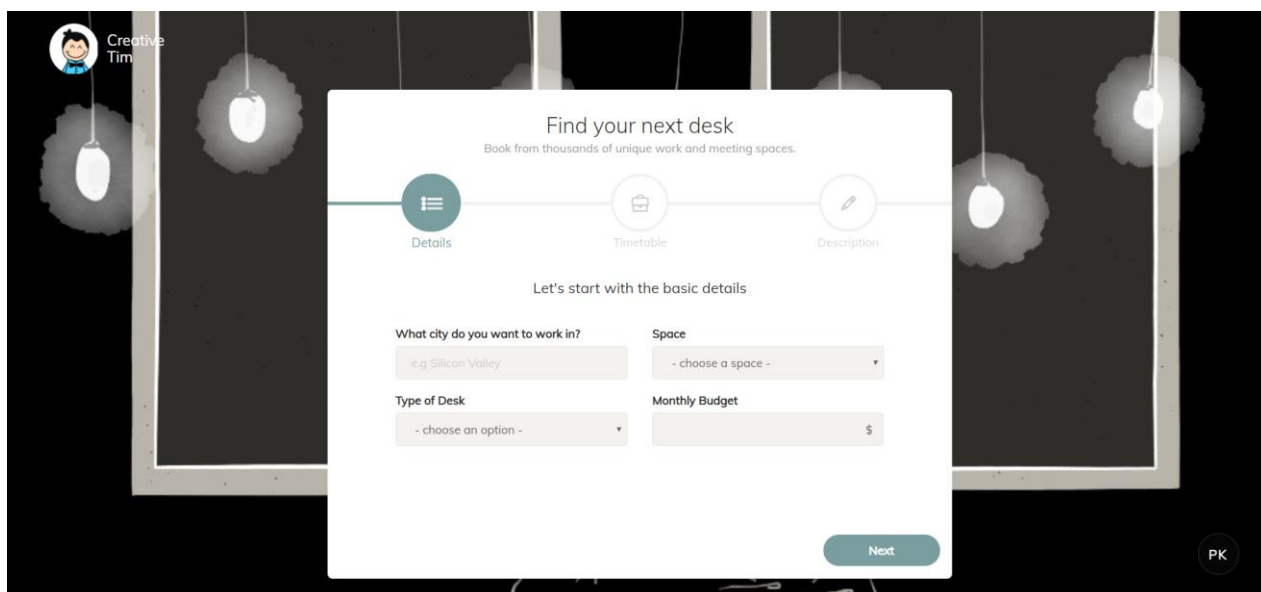
    for(var i = 0;i < ret.length;i++) {
        if(parseInt(ret[i].end_date) == end_date && parseInt(ret[i].start_date) == start_date &&
parseInt(ret[i].amount) == parseInt(response.amount)) {
            flag = false;
            break;
        }
    }
})

if(response.status == "yes" && flag == true) {
    // send a transcation
    api.sendRawTransaction(SupplyChainAddr,functionName,params).then(function(value) {
        var table_value = select(argv);
        table_value.then(function(ret) {
            if(ret.length > table_len) {
                var htmlfile = part1 + response.comname_from + part2 + response.comname_to + part3 +
response.start_date + part4 + response.end_date + part5 + response.amount + part6;
                res.send(htmlfile);
            }
            else {
                var wrong = "<p style=\"color: red\">wrong</p>"
                var htmlfile = part1 + wrong + part2 + wrong + part3 + wrong + part4 + wrong + part5 + wrong +
+ part6;
                res.send(htmlfile);
            }
        })
    });
} else {
    var wrong = "<p style=\"color: red\">wrong</p>"
    var htmlfile = part1 + wrong + part2 + wrong + part3 + wrong + part4 + wrong + part5 + wrong +
part6;
    res.send(htmlfile);
}
})

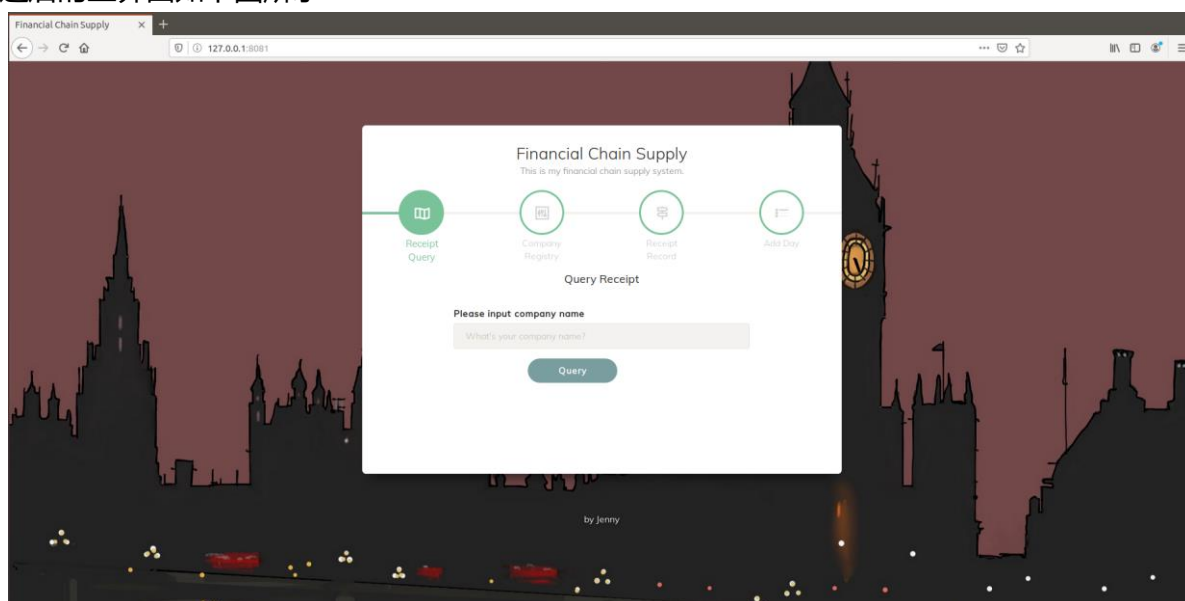
```

### 3. 前端实现

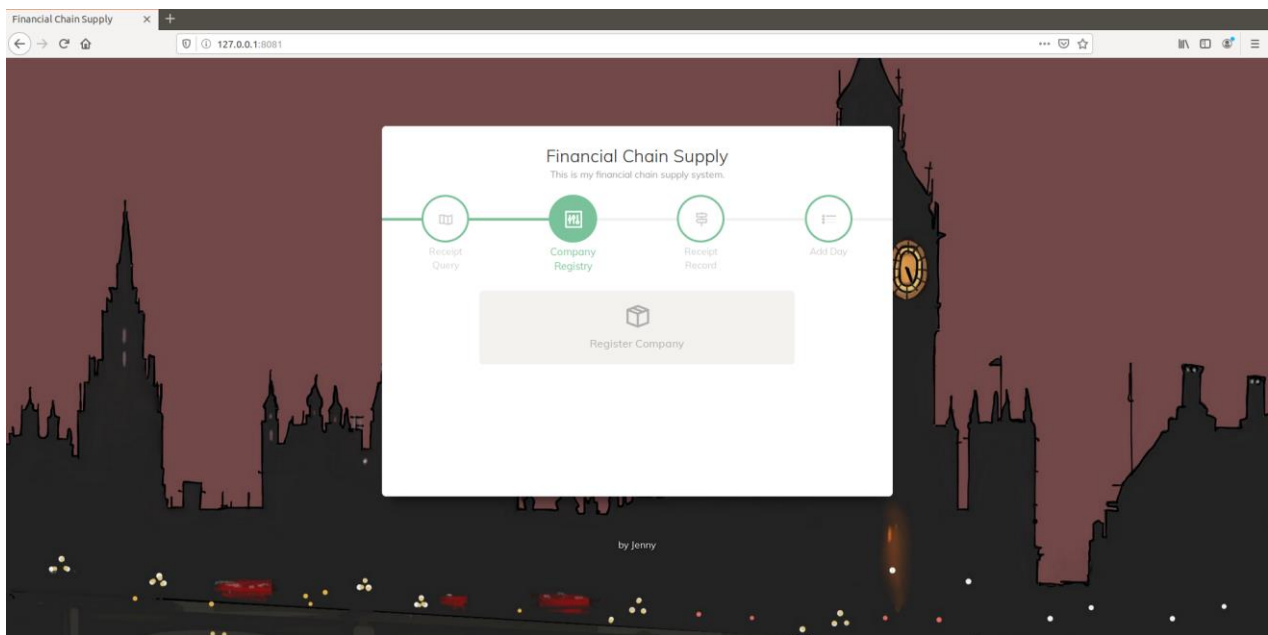
为了更好的用户体验，我借用了 github 上的 html [前端界面](#)，原界面如下图所示：



改造后的主界面如下图所示：



以注册公司为例，进行前端代码的讲解。在主界面中点击 Company Registry 的 Tab bar 进行跳转。



切换到 Company Registry 一栏后，点击 Register Company 组件进行跳转，该组件将会跳转到公司注册/`company_reg.html` 界面，该组件的 html 代码如下所示。

```
<div class="tab-pane" id="type">
  <div class="row">
    <div class="col-sm-18">
      <div class="col-sm-8 col-sm-offset-2">
        <div class="choice" data-toggle="wizard-checkbox"
onclick="location='/company_reg.html'">
          <input type="checkbox" name="jobb" value="Design" >
          <div class="card card-checkboxes card-hover-effect">
            <i class="ti-package"></i>
            <p>Register Com
              pany</p>
          </div>
        </div>
      </div>
    </div>
  </div>
</div>
```

公司注册的布局代码（主要部分）如下：

```
<div class="tab-content">
  <div class="tab-pane" id="about">
    <div class="row">
      <h5 class="info-text"> Please tell us more about yourself.</h5>
      <div class="col-sm-10 col-sm-offset-1">
        <div class="form-group">
          <label>Company Name <small>(required)</small></label>
          <input name="company_name" type="text" class="form-control" placeholder="Company
Name">
        </div>
      </div>
    </div>
  </div>
</div>
```

```

</div>
<div class="col-sm-10 col-sm-offset-1">
  <div class="form-group">
    <label>Email</label>
    <input name="email" type="email" class="form-control" placeholder="jenny@xxx.com">
  </div>
</div>
</div>
</div>
<div class="tab-pane" id="account">
  <h5 class="info-text"> Is your business a bank? (checkbox) </h5>
  <div class="row">
    <div class="col-sm-8 col-sm-offset-4">
      <div class="col-sm-5">
        <div class="choice" data-toggle="wizard-checkbox">
          <input type="checkbox" name="checkbox">
          <input type="text" style="display:none" name="company_kind" value="common">
          <div class="card card-checkboxes card-hover-effect">
            <i class="ti-pencil-alt"></i>
            <p>Bank</p>
          </div>
        </div>
      </div>
    </div>
  </div>
</div>
</div>
</div>
<div class="tab-pane" id="address">
  <div class="row">
    <div class="col-sm-12">
      <h5 class="info-text"> Please input your Company Address </h5>
    </div>
    <div class="col-sm-10 col-sm-offset-1">
      <div class="form-group">
        <label>Company Address <small>(required)</small></label>
        <input type="text" name="company_address" class="form-control" placeholder="XXXX">
      </div>
    </div>
  </div>
</div>
</div>
<div class="wizard-footer">
  <div class="pull-right">
    <input type="button" class="btn btn-next btn-fill btn-warning btn-wd" name='next' value='Next'
  />
    <input type="submit" class="btn btn-finish btn-fill btn-warning btn-wd" name='finish'
value='Finish' />
  </div>
  <div class="pull-left">
    <input type="button" class="btn btn-previous btn-default btn-wd" name='previous'
value='Previous' />
  </div>
  <div class="clearfix"></div>
</div>

```

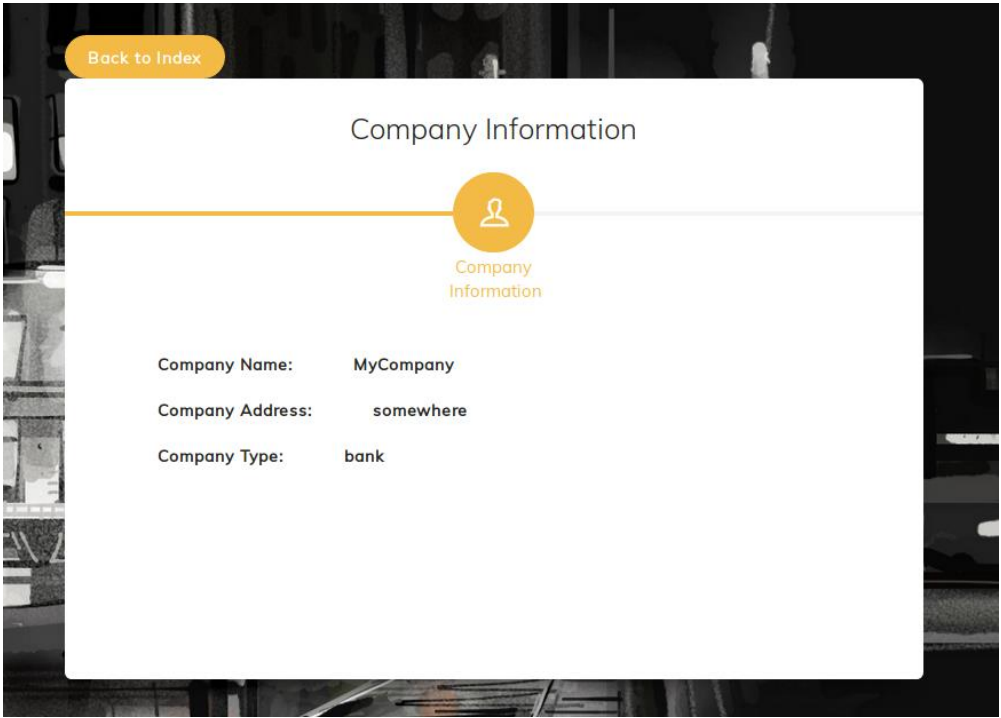
公司注册界面如下所示：

The screenshot shows a web form titled "Create your company profile" with the subtitle "This information will let us know more about you company." The form is divided into three steps: "Company Name", "Company Profile", and "Address". The "Company Name" step is currently active, indicated by an orange circle and line. Below the step indicators, the text "Please tell us more about yourself." is displayed. The form contains two input fields: "Company Name (required)" with the placeholder text "Company Name", and "Email" with the placeholder text "jenny@xxx.com". A "Next" button is located at the bottom right of the form. A "Back to Index" button is located at the top left of the form.

注册公司需要：公司名称、公司邮箱、公司类型以及公司地址的信息。其中必须要填写的是公司名称和公司地址，且公司名称不能重名，如果注册重名公司将会显示 WRONG。公司名称不得少于三个字符，前端有公司和邮箱的合法性检查。同时公司类型默认是 common，如果该公司是银行，则需要点击下图的 CheckBox，选中 Bank

The screenshot shows the same web form, but now the "Company Profile" step is active, indicated by an orange circle and line. The "Company Name" and "Email" fields are now disabled. The text "Is your business a bank? (checkbox)" is displayed. Below this text is a checkbox labeled "Bank" with an orange icon of a bank building. A "Previous" button is located at the bottom left of the form, and a "Next" button is located at the bottom right of the form. The "Back to Index" button remains at the top left.




注册成功后将显示注册信息：



三、 功能测试

注册公司

注册三家公司，分别为 AAA,BBB,CCC 用来进行测试

Company Information	Company Information	Company Information
		
Company Name: AAA	Company Name: BBB	Company Name: CCC
Company Address: AAA	Company Address: BBB	Company Address: CCC
Company Type: common	Company Type: common	Company Type: common

交易上链

BBB 公司向 CCC 公司借款 200 元，借款时间是 2019-05-01，还款日期是 2020-05-01。



**Make out a receipt**  
We are committed to solving the financing difficulties of smes.

Back to Index

Details Status

Let's start with the basic details

Name of Borrowing Company:

Name of Loan Company:

Start Date of Receipt:

End Date of Receipt:

Amount:  \$

Next

---

**Receipt Information**

Receipt

Name of Borrowing Company: BBB

Name of Loan Company: CCC

Start Date of Receipt: 2019-05-01

End Date of Receipt: 2020-05-01

Amount: 200 \$

Back to Index

## 应收账款的转让上链

AAA 公司向 BBB 公司借款 100 元，借款时间是 2019-10-01 号，还款日期是 2019-12-01 号。在此之前，BBB 公司向 CCC 公司借款 200 元，借款时间是 2019-05-01 号，还款日期是 2020-05-01 号。按照我的供应链的逻辑，此时的借贷发生了转移，AAA 公司向 BBB 公司借款 100 元，而 BBB 公司向 CCC 公司借款 200 元，这两个借贷关系等价于 AAA 公司向 CCC 公司借款

100 元，还款日期应该为两个还款日期更早的那一个，而 BBB 公司向 CCC 公司借款 100 元，还款日期不变。

Back to Index

Make out a receipt

We are committed to solving the financing difficulties of smes.

Details

Status

Let's start with the basic details

Name of Borrowing Company

AAA

Name of Loan Company

BBB

Start Date of Receipt

10/01/2019

End Date of Receipt

12/01/2019

Amount

100

\$

Next

可以看到 CRUD 查询中，CCC 的账单中，两个欠款人分别为 AAA 和 BBB，欠款数额均为 100，还款日期分别为 2019-12-01 号和 2020-05-01 号。

Back to Index

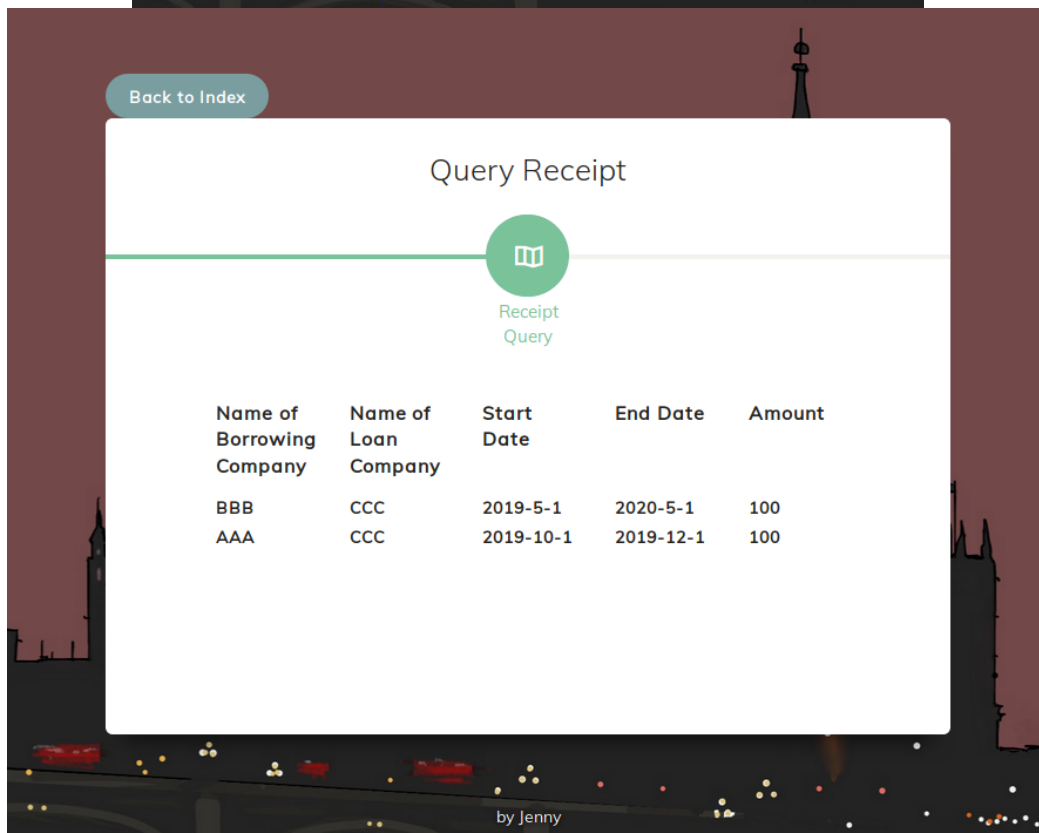
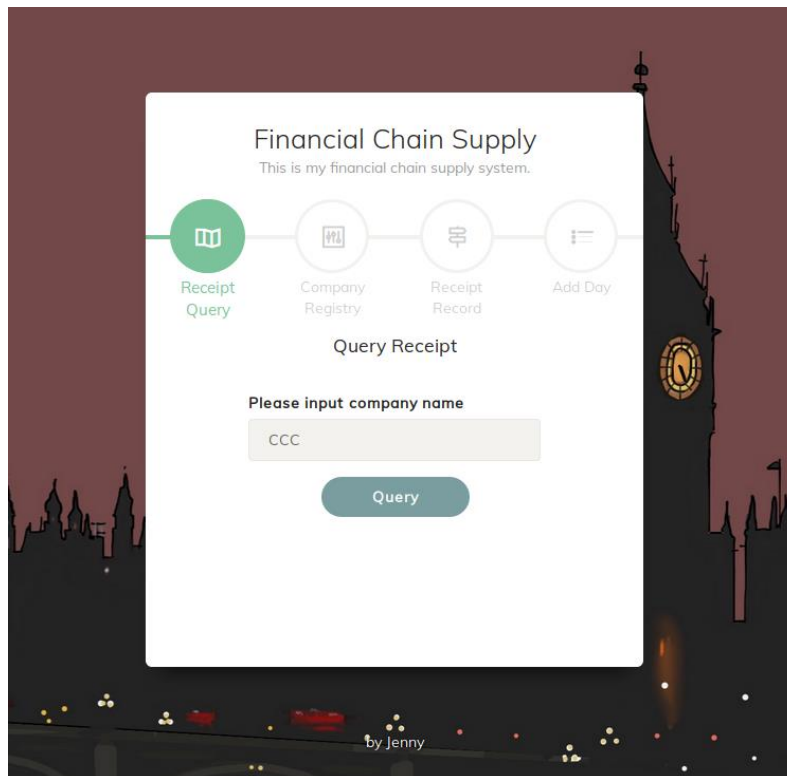
Query Receipt

Receipt Query

Name of Borrowing Company	Name of Loan Company	Start Date	End Date	Amount
BBB	CCC	2019-5-1	2020-5-1	100
AAA	CCC	2019-10-1	2019-12-1	100

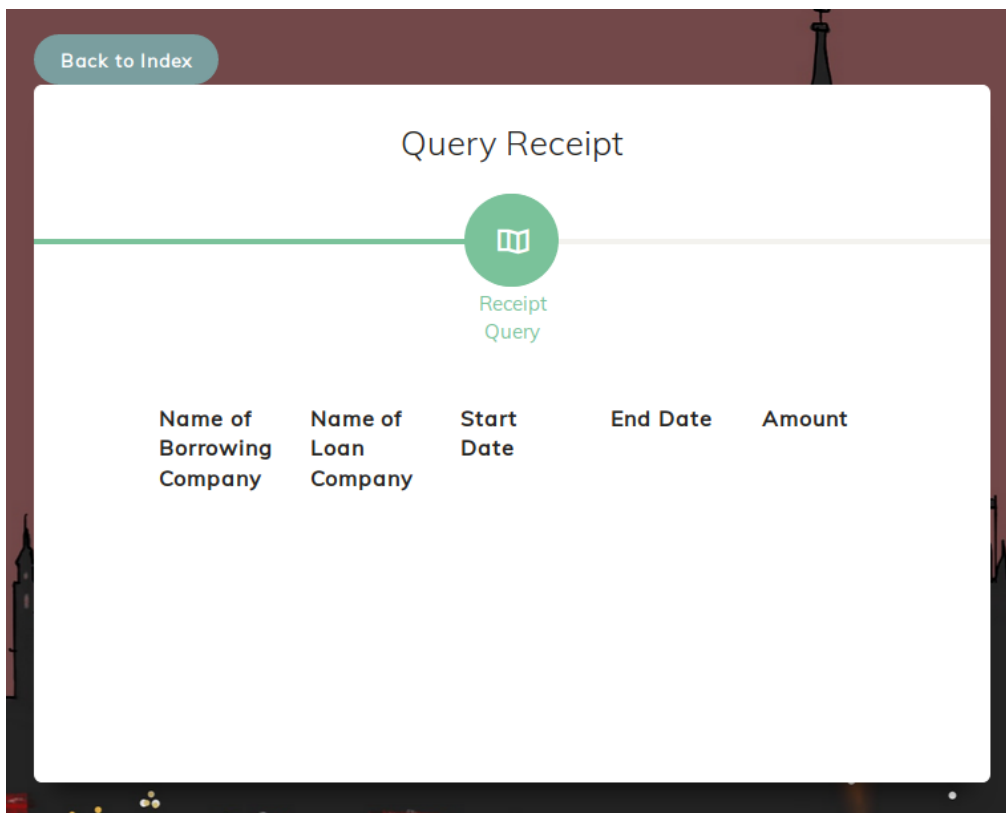
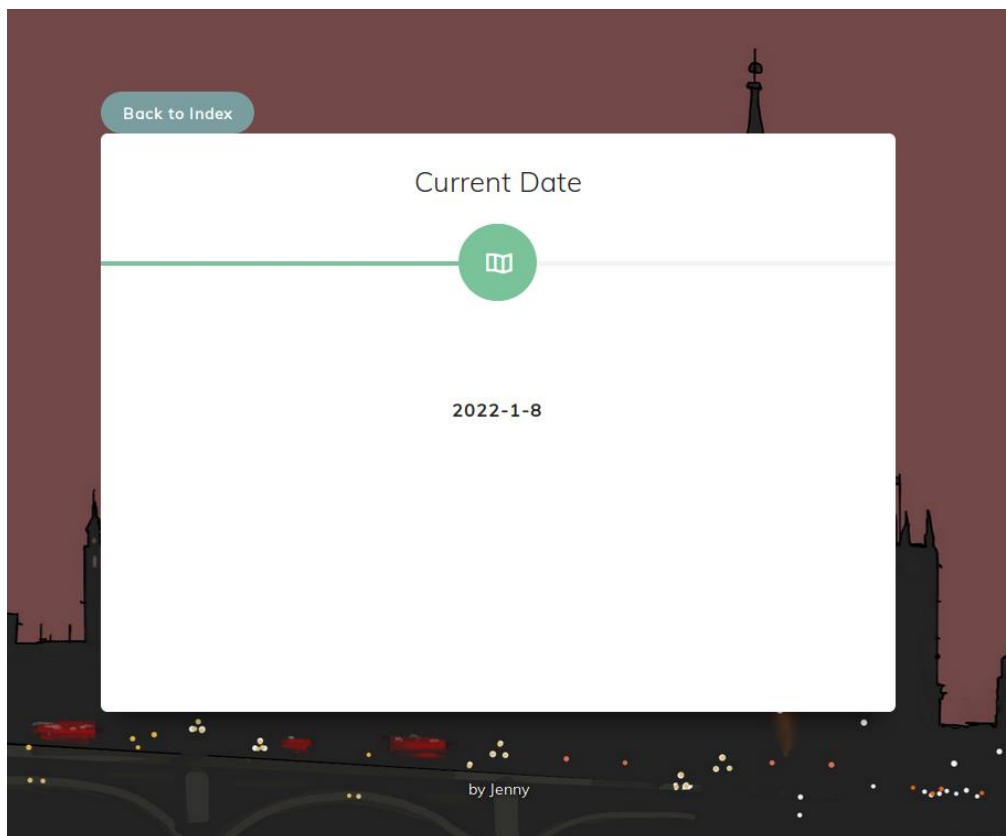
查询应收账款

查询 CCC 公司的借贷情况。



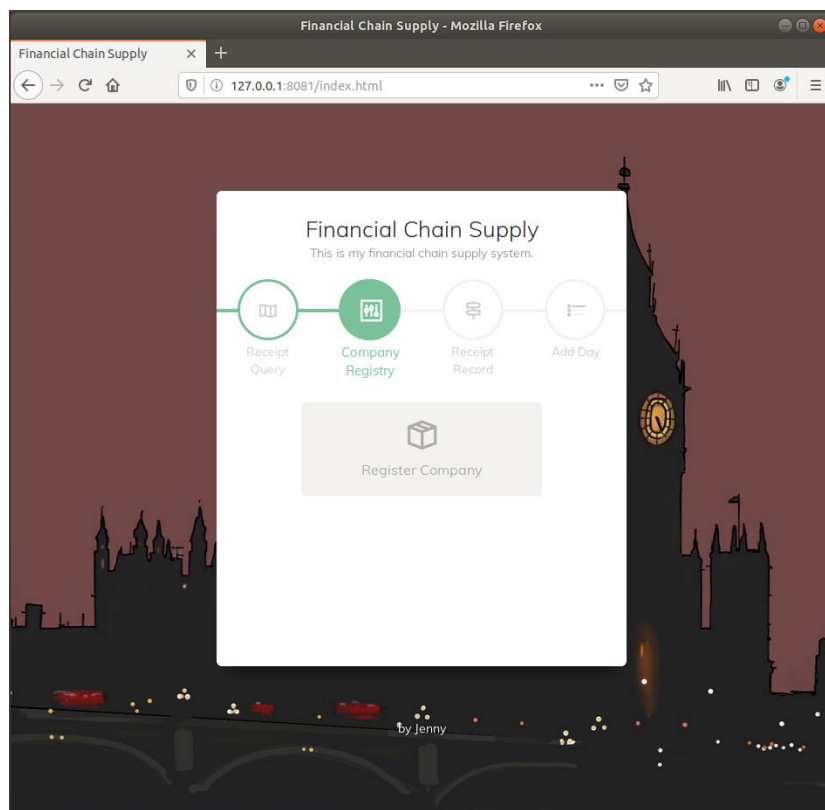
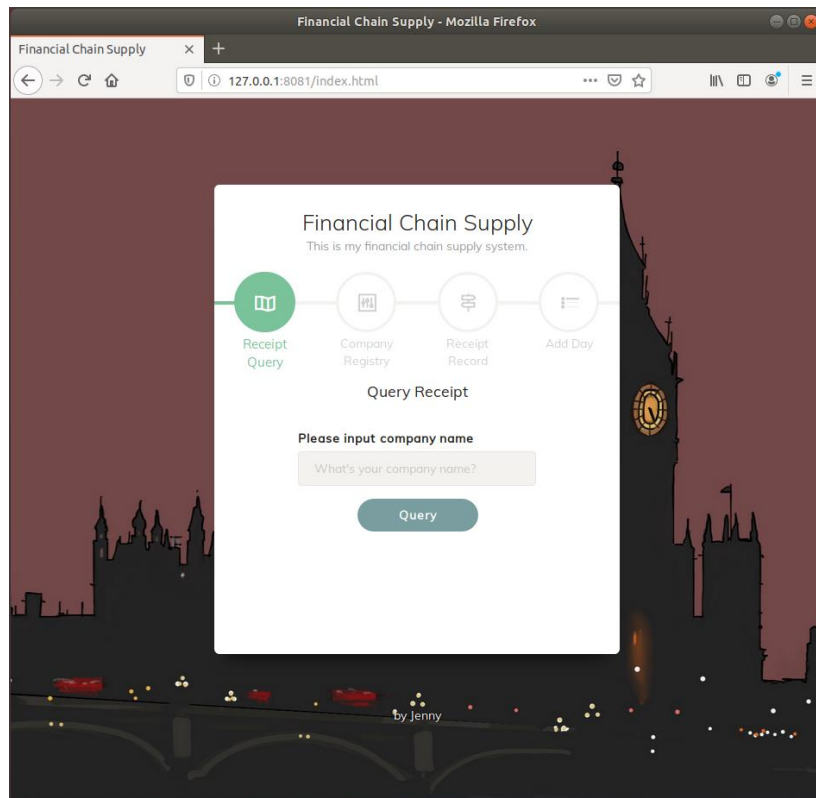
## 应收账款支付结算上链

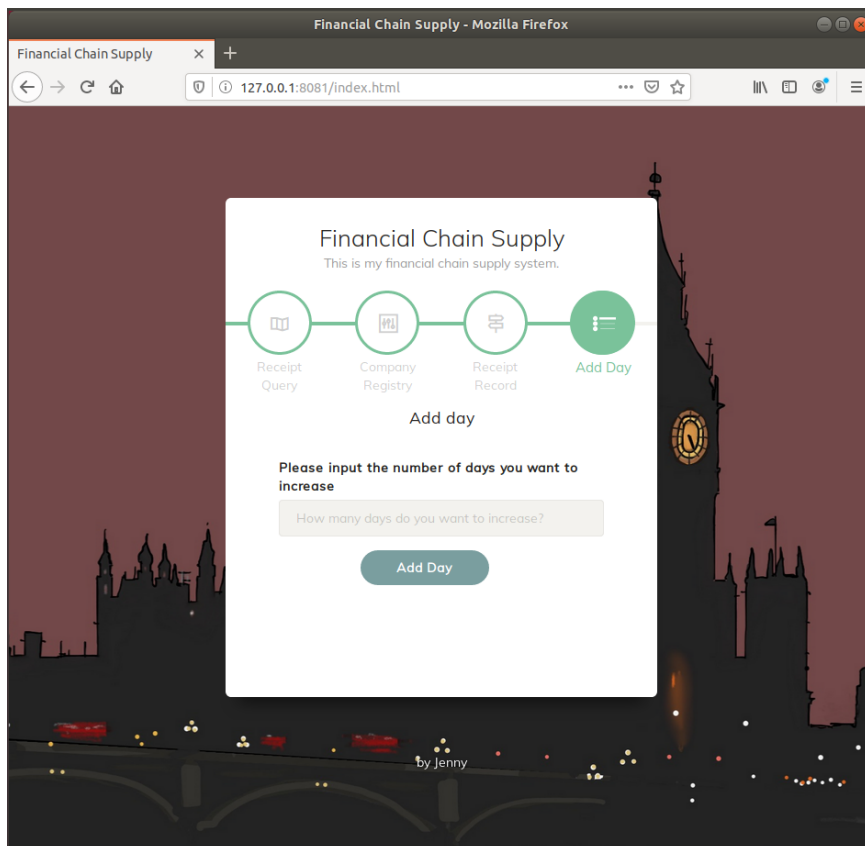
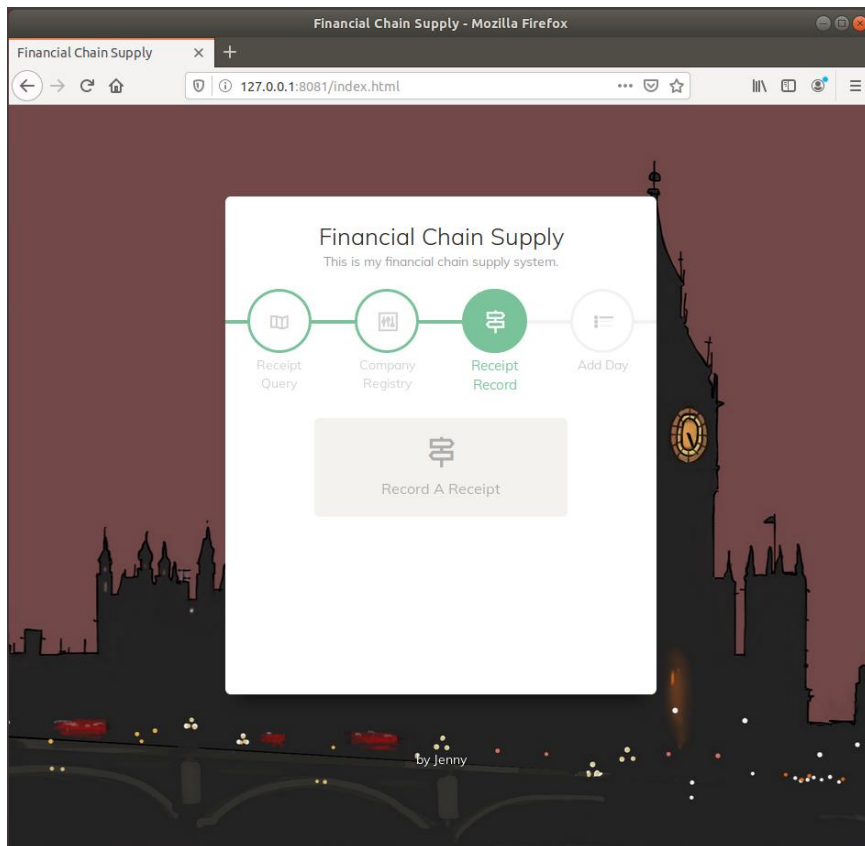
修改日期为 2022-01-08，此时 CCC 的交易应该已经结算，所以账单被清除。



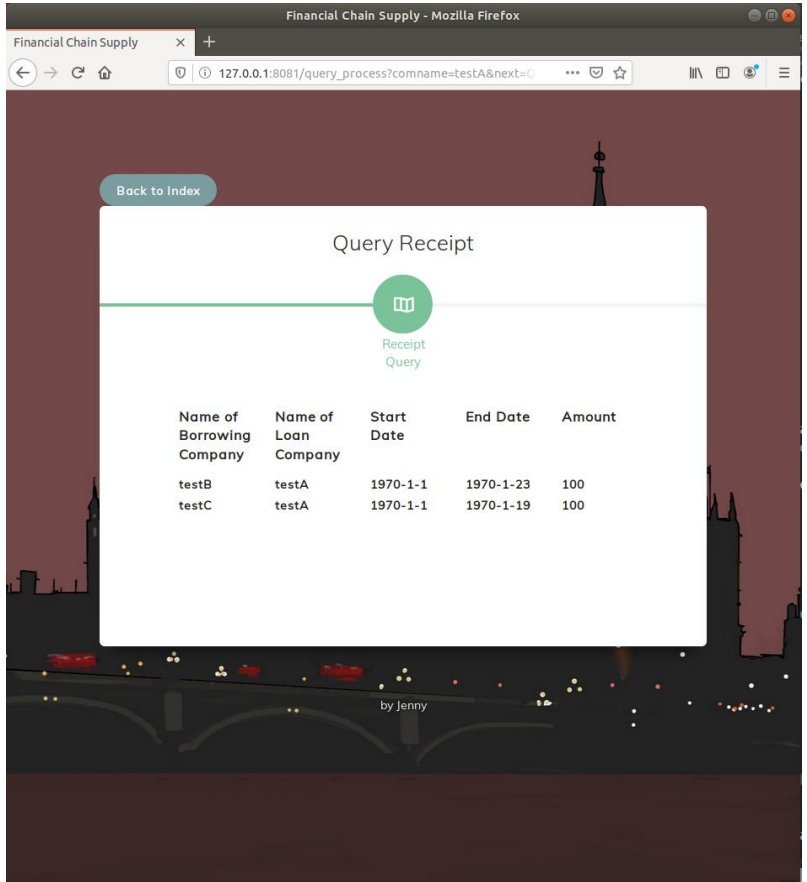
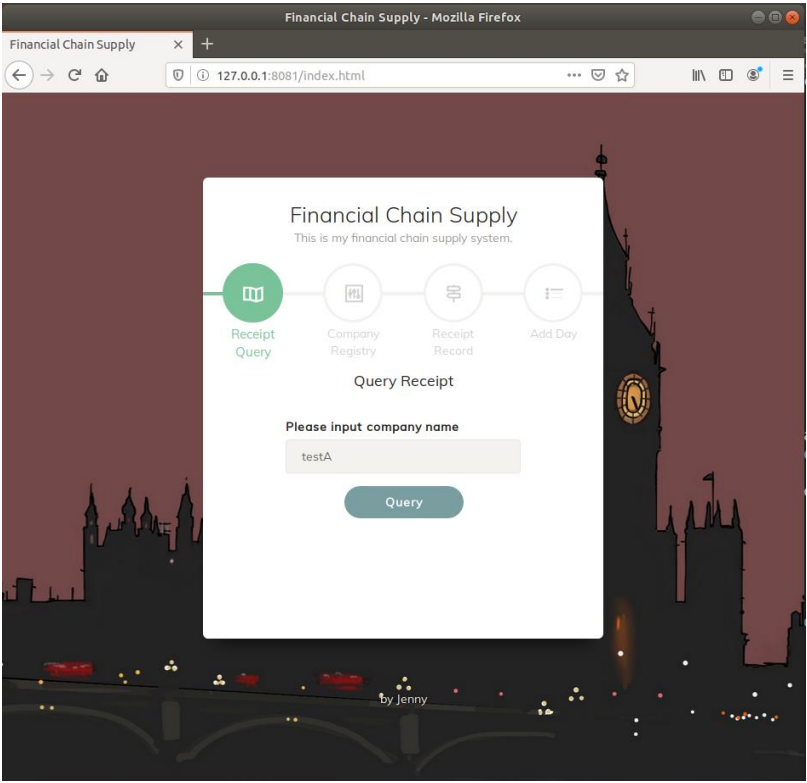
## 四、 界面展示

### 主界面

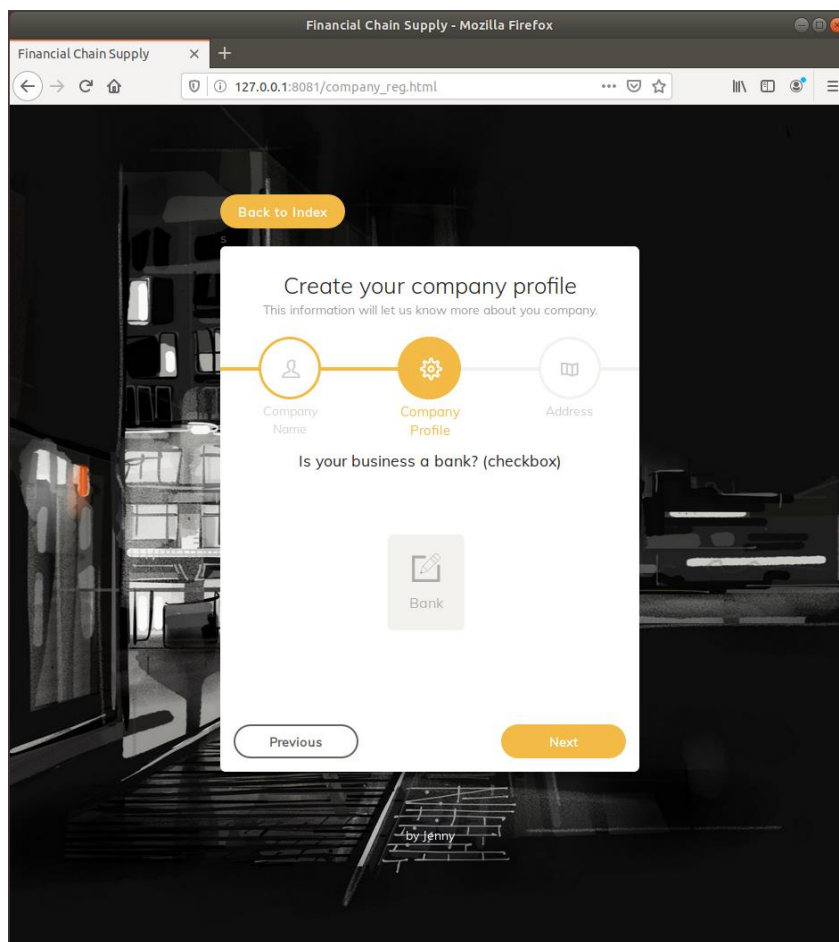
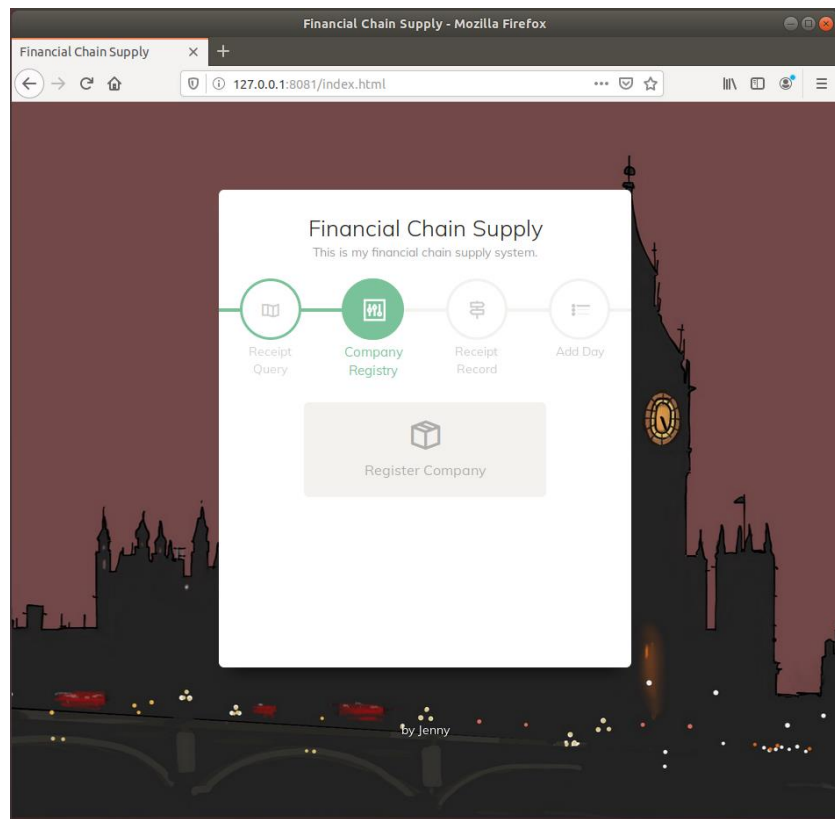




查询界面



## 企业注册界面





Financial Chain Supply - Mozilla Firefox


Financial Chain Supply x +


127.0.0.1:8081/company\_reg.html


Back to Index

### Create your company profile

This information will let us know more about you company.

  
Company Name

  
Company Profile

  
Address

Please tell us more about yourself.

**Company Name (required)**

**Email**

Next

Financial Chain Supply - Mozilla Firefox


Financial Chain Supply x +


127.0.0.1:8081/company\_reg.html


Back to Index

### Create your company profile

This information will let us know more about you company.

  
Company Name

  
Company Profile

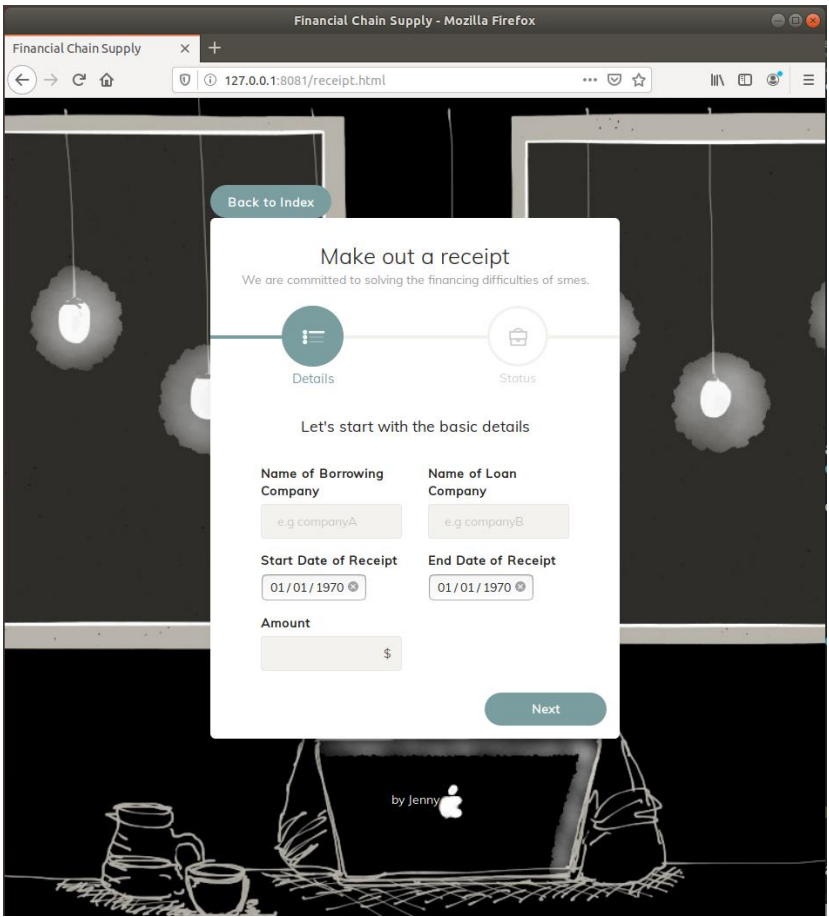
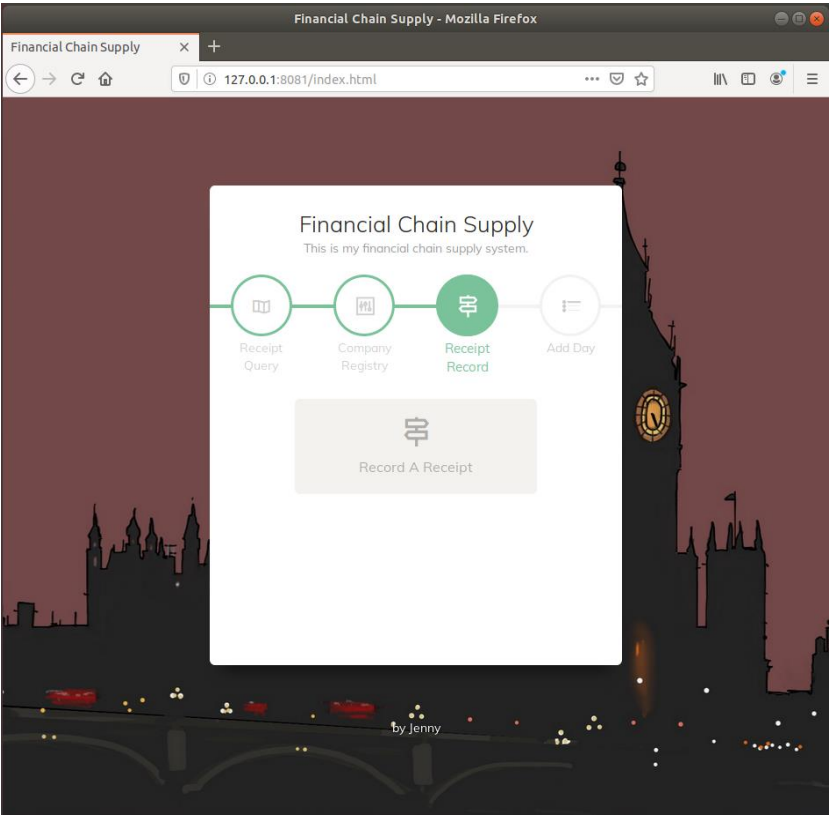
  
Address

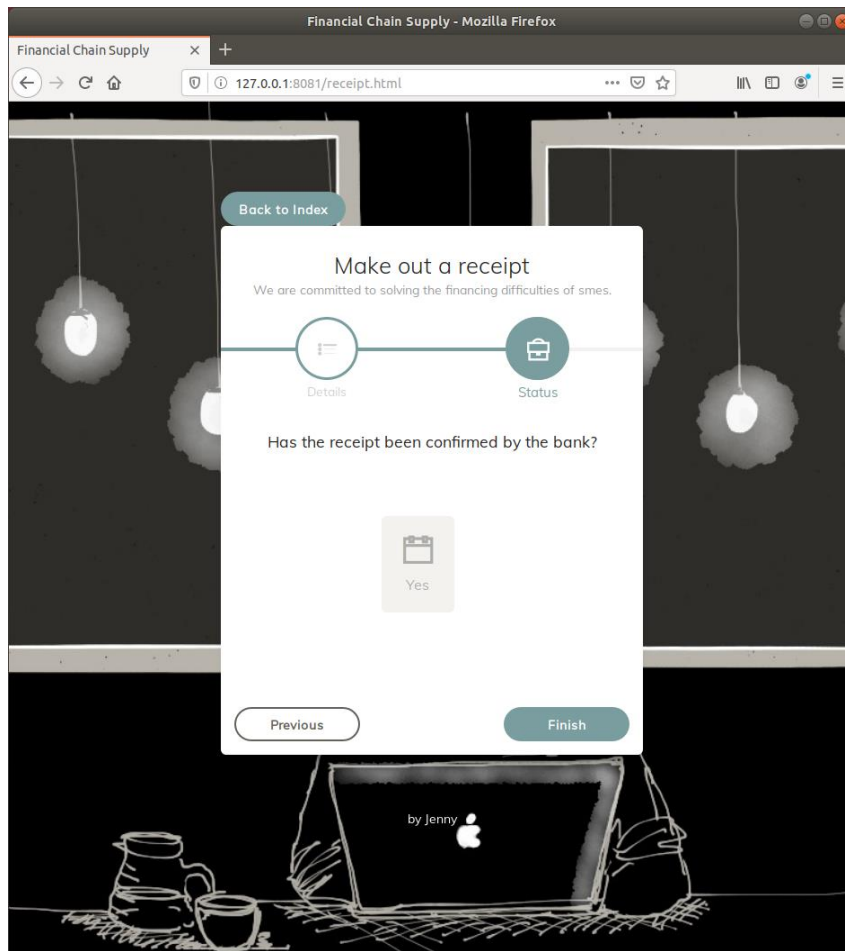
Please input your Company Address

**Company Address (required)**

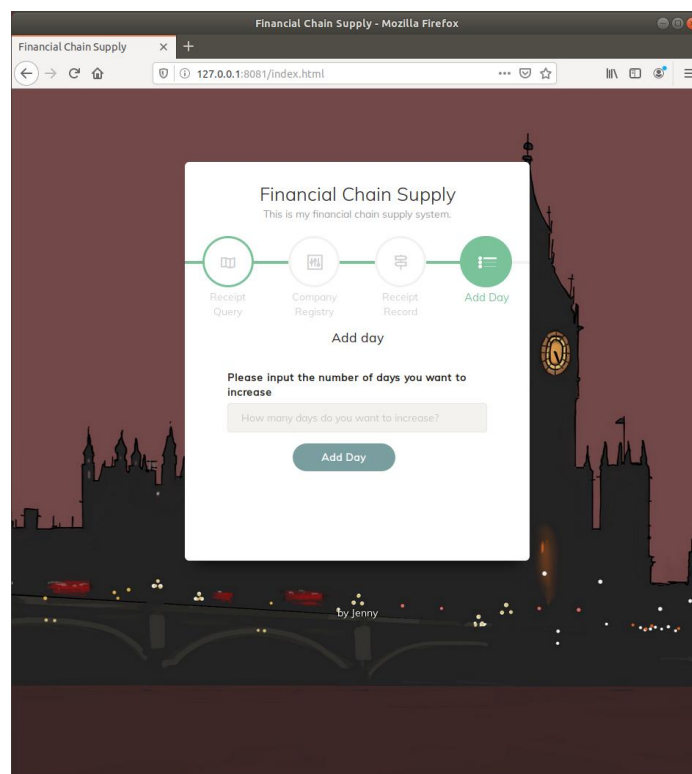
Previous Finish

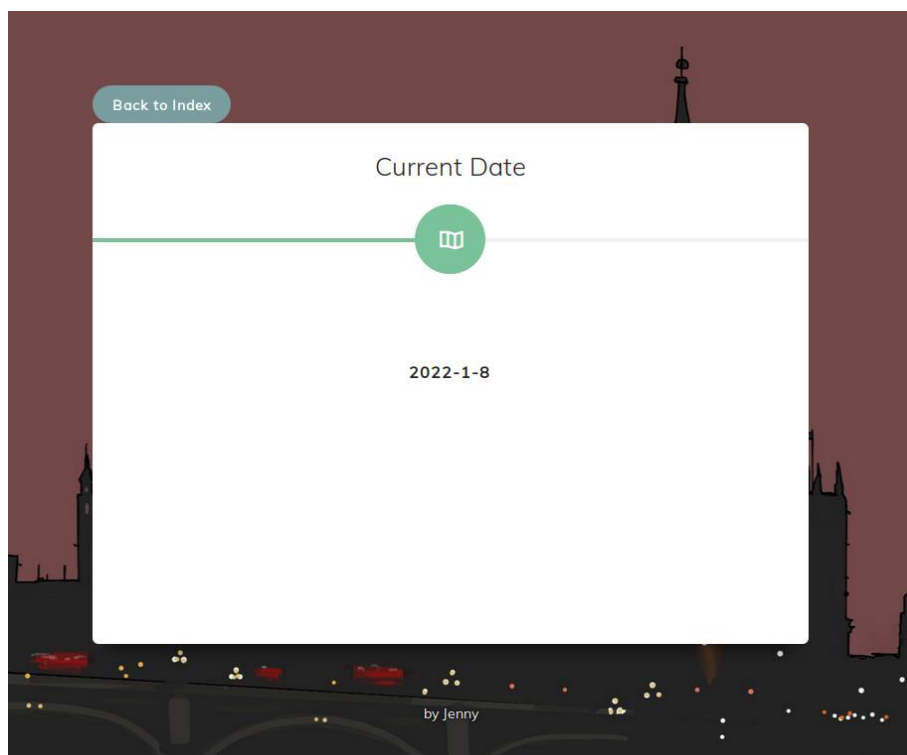
记录交易界面





修改日期界面





## 五、 心得体会

本次大作业，我完成了一个区块链的在供应链金融方面的小项目。从开始对区块链完全陌生，到现在已经可以独立完成一个较为完整区块链项目，中间经历了三个月的时间，不可谓收获不大。初始对智能合约一窍不通，通过编写智能合约，部署到区块链上，熟悉了智能合约的编写规范。前期的铺垫较长，到第二阶段时，已经可以使用数据库进行数据的增删改查，提高了编写的效率和规范性。在第二个阶段，我考虑很多情况，但发现实际情况确实太过于复杂，只能简化处理。饶是如此，仍然写了三百多行的智能合约。

因为在第二阶段的智能合约写的较为完整，所以这第三阶段并没有更改上一阶段写的智能合约，而是使用了官方的 API 进行前端和后端的开发，真正使得应用可以落地。因为使用的是 Node.js 的 SDK，官方的说明很少，全靠自己看命令行样例 cli.js 来摸索接口的使用方法。我看了两天的代码，终于搞懂了 API 的用法。因为官方的 Node.js API 还在开发阶段，使用起来并不方便，所以我也更改了 API 部分的代码，使得开发更方便。前端界面在 github 上找了一个较为精致的排版布局，在这个排版基础上进行改进，最后的效果还是很好的。

总而言之，本次大作业不仅仅学习了区块链的相关知识，还拓展了视野，锻炼了全栈开发能力，收货巨大。

## 六、 加分项

我设计了比较友好高效的界面。除了排版美观之外，还有很多用户提示，使得用户使用

更为方便。如注册公司时，如果不填写必填项，会有红色字体提示；邮箱格式错误也会有提示。如下图。

The screenshot shows a mobile app interface for creating a company profile. At the top, there is a yellow button labeled "Back to Index". Below it, the title "Create your company profile" is displayed, followed by the subtitle "This information will let us know more about you company.". A progress bar with three icons (person, gear, book) indicates the steps: "Company Name", "Company Profile", and "Address". The "Company Name" step is currently active. The main content area asks "Please tell us more about yourself." and contains two input fields. The first field is labeled "Company Name (required)" and contains the placeholder text "Company Name"; below it, a red error message states "This field is required.". The second field is labeled "Email" and contains the placeholder text "dasdf"; below it, a red error message states "Please enter a valid email address.". A yellow "Next" button is located at the bottom right of the form. The background of the app shows a dark, stylized image of a building at night.

默认项的填写通过 CheckBox 实现，便于用户的输入操作。

The screenshot shows the same mobile app interface, but now on the "Company Profile" step. The progress bar shows the "Company Profile" step (gear icon) as active. The main content area asks "Is your business a bank? (checkbox)". Below this question is a checkbox control with a pencil icon and the label "Bank". At the bottom of the form, there are two buttons: a white "Previous" button on the left and a yellow "Next" button on the right. The background remains the same dark, stylized image of a building at night.

拥有较为精美的交易界面，日期可以通过选择日期填入，简洁明了。

Back to Index

## Make out a receipt

We are committed to solving the financing difficulties of smes.

Details

Status

Let's start with the basic details

**Name of Borrowing Company**  
e.g companyA

**Name of Loan Company**  
e.g companyB

**Start Date of Receipt**  
01/01/1970

**End Date of Receipt**  
01/01/1970

**Amount**  
\$

< January 1970 >

Sun	Mon	Tue	Wed	Thu	Fri	Sat
28	29	30	31	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31
1	2	3	4	5	6	7

by Jenn