

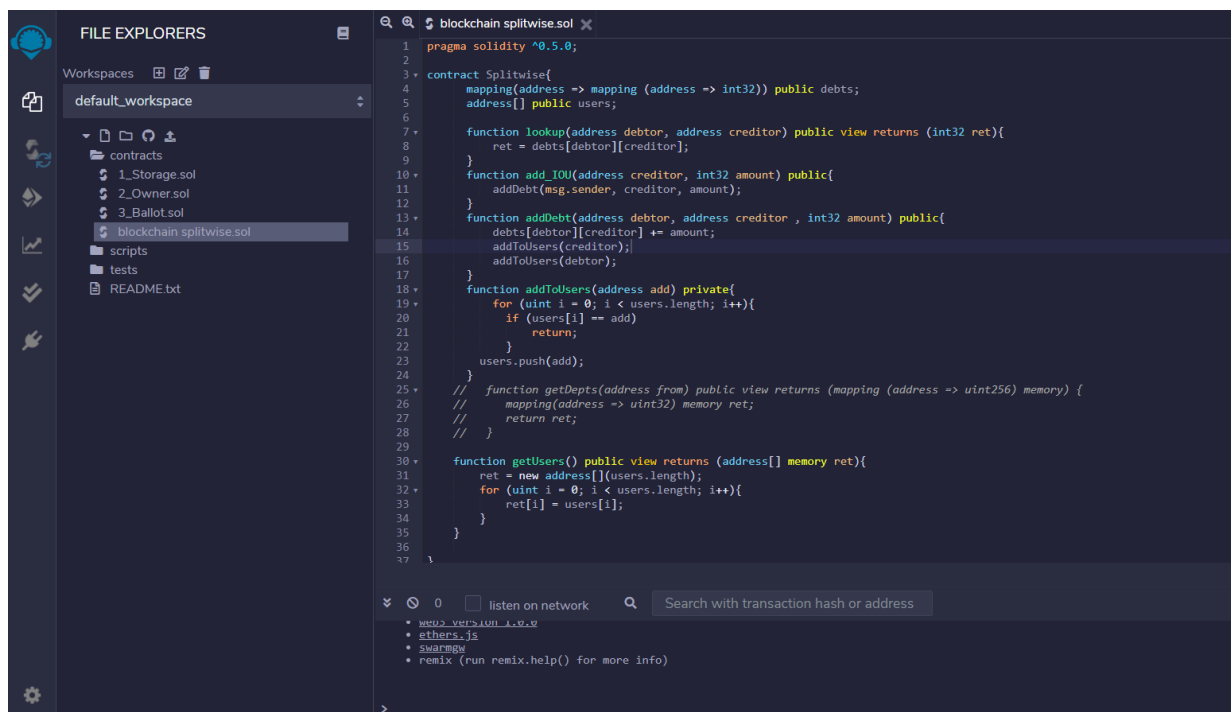
گزارش پروژه سوم ارز های رمز گذاری شده : قرارداد هوشمند

محسن امجدی-810896043

هدف : در این پروژه قصد داریم یک سیستم غیر متمرکز طراحی کنیم که بتواند بدهی و طلب افراد را ثبت کند. در واقع نمونه ای است غیر متمرکز از یک دفترچه حساب. سیستم غیر متمرکزی که با استفاده از قرارداد هوشمند طراحی و پیاده سازی میکنیم ، مشخص میکند هر شخص به چه کسی و چه مبلغی بدهکار است، بدون این که نیاز باشد یک سازمان سومی در میان باشد. به این صورت که اگر کسی به شخص دیگری بدهکار بود آن شخص از او بخواهد که در شبکه ی زنجیره ی بلوک بدهکار بودنش را تصدیق کند. نحوه ی تصدیق کردن بدین شکل است که فرد بدهکار یک تابع را صدا می زند و در آن شخص طلبکار و مبلغ بدهی را مشخص میکند. در کد برنامه این تصدیق نامه(تاییدیه) را IOU می نامیم. فضای موجود و عمومی و همچنین قابل دسترس شبکه ی زنجیره ی بلوک می تواند به عنوان محل ذخیره سازی و یا به عبارتی سمت سرور برای لیست بدهی های هر شخص استفاده شود. برای راحتی کار با این سرور لازم است که یک محیط کاربری نیز طراحی شود که طراحی آن در فایل "index.html" آورده شده است.

پیاده سازی و اجرا: در ابتدا باید قرارداد هوشمند خود را پیاده سازی کنیم تا بتوانیم آن را بر روی شبکه ی زنجیره ی بلوک مستقر کنیم . طراحی و پیاده سازی قرارداد هوشمند در فایل "BlockchainSplitwise.sol" قرار گرفته است که با توجه به پسوند آن یک برنامه ی نوشته شده با زبان "solidity" می باشد. پیاده سازی تمامی توابع خواسته شده برای قرارداد هوشمند در صورت مسئله، از جمله- "add_IOU" و "lookup" و توابع دیگری مثل "addDebt" یا "getUsers" - که توضیحات کاربرد آن ها در صورت مسئله و در ویدیو ضبط شده آورده شده؛ در این فایل قرار گرفته اند. همچنین پیاده سازی تمامی توابع سمت کاربر که در صورت مسئله خواسته شده با توضیحات مربوط به آن ها به صورت کامنت در کد فایل "script.js" قرار گرفته است.

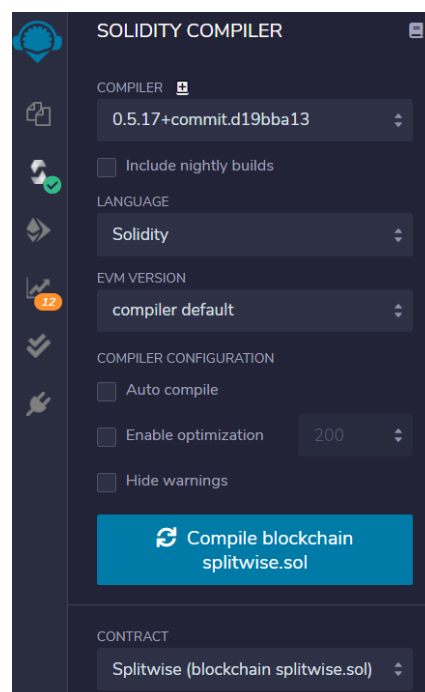
برای اجرای قرارداد در ابتدا وارد سایت "<https://remix.ethereum.org>" می شویم تا قرارداد هوشمند خود را بر روی شبکه زنجیره ی بلوک مستقر کنیم.



The screenshot displays the Remix IDE interface. On the left, the 'FILE EXPLORERS' panel shows a workspace named 'default_workspace' containing a folder 'contracts' with files '1_Storage.sol', '2_Owner.sol', and '3_Ballot.sol'. The file 'blockchain splitwise.sol' is selected. The main editor shows the Solidity code for the 'Splitwise' contract. The code includes a pragma statement for Solidity version 0.5.0, a mapping for debts, an array for users, and functions for adding IOUs, adding debts, and retrieving data. The bottom panel shows the 'Compiler' tab with settings for compiler version (0.5.17+commit.d19bba13), language (Solidity), and EVM version (compiler default). A 'Compile blockchain splitwise.sol' button is visible.

```
1 pragma solidity ^0.5.0;
2
3 contract Splitwise{
4     mapping(address => mapping (address => int32)) public debts;
5     address[] public users;
6
7     function lookup(address debtor, address creditor) public view returns (int32 ret){
8         ret = debts[debtor][creditor];
9     }
10    function add_IOU(address creditor, int32 amount) public{
11        addDebt(msg.sender, creditor, amount);
12    }
13    function addDebt(address debtor, address creditor , int32 amount) public{
14        debts[debtor][creditor] += amount;
15        addToUsers(creditor);
16        addToUsers(debtor);
17    }
18    function addToUsers(address add) private{
19        for (uint i = 0; i < users.length; i++){
20            if (users[i] == add)
21                return;
22        }
23        users.push(add);
24    }
25    // function getDepts(address from) public view returns (mapping (address => uint256) memory) {
26    //     mapping(address => uint32) memory ret;
27    //     return ret;
28    // }
29    function getUsers() public view returns (address[] memory ret){
30        ret = new address[](users.length);
31        for (uint i = 0; i < users.length; i++){
32            ret[i] = users[i];
33        }
34    }
35 }
36
37 }
```

از طریق قسمت کامپایل قرارداد را کامپایل میکنیم.



حال برای ادامه ی کار باید شبیه ساز شبکه اتریوم “ganache-cli” را نصب و اجرا کنیم. با این کار یک سرور به نام “ganache” روی سیستم نصب می شود که با اجرا شدن آن تعدادی اکانت اتریوم (پیش فرض 10 عدد) با موجودی های تعیین شده (100 اتر) ایجاد میشود که از آن ها میتوانیم استفاده کنیم.

```
Ganache CLI v6.12.2 (ganache-core: 2.13.2)
```

Available Accounts

=====

```
(0) 0x9f7eEE88cb06cA17aA2d50AeaDa412210D86209A (100 ETH)
(1) 0x389e281D900CB9E513643D15bd71b3DE15A37630 (100 ETH)
(2) 0x6Bf5254049ddb4b0269B42A6D8153459ce19DbcB (100 ETH)
(3) 0x91A541a6917bFD51E1A93fb87faaB27dB3819017 (100 ETH)
(4) 0xf08c9abAB21E0a3A5379C2CC5A34A811616f2e53 (100 ETH)
(5) 0x56f1415651f1b1f77E397da870a56C5f332c4430 (100 ETH)
(6) 0x806e016c614Cf1B0ee1ec65d5486522FDFb39482 (100 ETH)
(7) 0x52b1E6bCEEdDffdf3Ef3df5716DCa304d7f04F8b5 (100 ETH)
(8) 0x013E59b9b4478714295C1B65D56B2aD36846b3FF (100 ETH)
(9) 0x79Cc34897bd0F361637DddF1bC2DAE2bb9d6d312 (100 ETH)
```

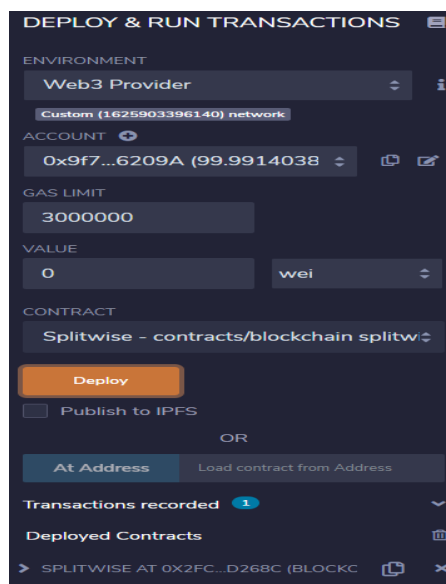
Private Keys

=====

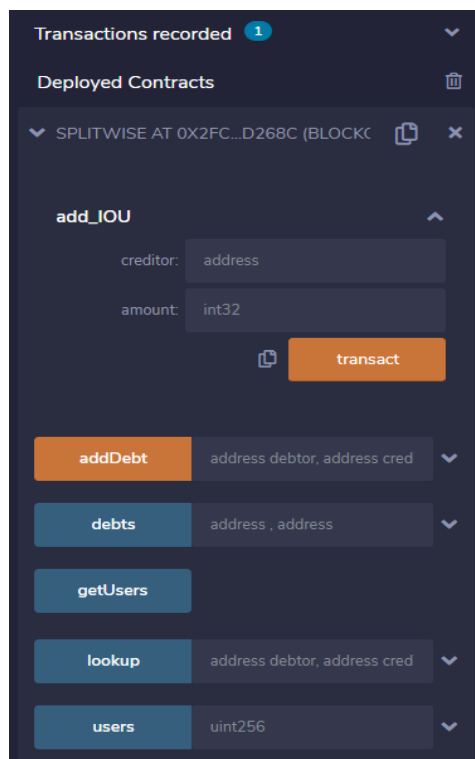
```
(0) 0x6367a5b5f279f9280316091eb1bd74b7c52832e502fa09fa1d0939a6e8f78d62
(1) 0x7eb8210dc20495a55a16d807577130375dbc5231491994233857e26f0976d9af
(2) 0x21858bc0abf37c8861ee659802cdf601d3df06a3d9b492a9b42dce8f4a5c225
(3) 0x6241715c3437d6dd0d9a0ae8e98edcb839ef8b1b9f72de4cf98e2b1275fa917d
(4) 0x554cd2f42f5364c9dff2300a118343d26021bd67f6fd7fb17e5b1ea34b6e9fee
(5) 0x9282594719fcdad1d457d7d53dac90cf97f9bbdfbb93772b8765fdb0c50bbcf2
(6) 0x780b977009d67054c62e3a5a84c805d5174542ca1a3a7829b40c4dc3acbdb259
(7) 0x3093b493eaf781a8d6c244600156a2cdf0741031069a602c47f21de37a8181f
(8) 0x5fac77bb0296f9656faa5b45d7916c52df3e8220aa3461df372e4e9bdf39c93f
(9) 0x25bcebc7cc8abfd946cec9cff0f341b5b03a94a43b8381f85b32ccaed11b2ab1
```

HD Wallet

در ادامه از طریق قسمت “DEPLOY & RUN TRANSACTIONS” قرارداد را روی شبکه زنجیره بلوک مستقر میکنیم که آدرس قرارداد هوشمند نیز به ما داد می شود. در این قسمت محیط اجرایی را روی “web3 provider” تنظیم میکنیم. “web3.js” مجموعه از کتابخانه هایی هستند که به ما امکان تعامل با یک نود اتریوم به صورت “local” یا “remote” را میدهند. “provider” ها نحوه ی تعامل “web3” با بلاکچین هستند، که درخواست های “JSON-RPC” را دریافت و پاسخ را برمیگردانند. این کار معمولاً با ثبت درخواست ها به یک سرور مبتنی بر سوکت با “HTTP” یا “IPC” انجام میشود.



با مستقر شدن قرارداد هوشمند ، امکان استفاده از توابع پیاده سازی شده از طریق همین سایت فراهم میشود. در تصویر زیر مشاهده میکنید که تمام توابع نوشته شده در قرارداد هوشمند آماده ی اجرای تراکنش ها هستند.



بنابراین در این لحظه می توانیم توابع مورد نظر را تست کنیم. برای شروع از یک تراکنش تصدیق بدهی “IOU” استفاده میکنیم.

The screenshot shows a web application interface for adding an IOU. The left sidebar contains a menu with the following items: 'addDebt' (selected), 'debts', 'getUsers', 'lookup', and 'users'. The main area displays a transaction confirmation for 'add_IOU'. The transaction details are as follows:

- status: true Transaction mined and execution succeed
- transaction hash: 0x30dd41aecd182fd552a86b460c049e425657e6edf3e0104d109bcdc3e8a7805
- from: 0x9f7eEE88cb06cA17a2d50aead412210086209A
- to: Splitwise.add_IOU(address,int32) 0x2fce2A3D1B3fD421B4fe77bCA9fa2686C76d268C
- gas: 112602 gas
- transaction cost: 112602 gas
- hash: 0x30dd41aecd182fd552a86b460c049e425657e6edf3e0104d109bcdc3e8a7805
- input: 0xada...000005
- decoded input: { "address creditor": "0x389e281D900CB9E513643D15bd71b3DE15A37630", "int32 amount": 5 }
- decoded output: -
- logs: []
- value: 0 wei

در حال حاضر آدرس اکانت ما، که با آن تراکنشی قرار است انجام دهیم به صورت دیفالت بر روی اولین آدرس قرار داده شده از “ganache” می باشد. در تابع “add_IOU” آدرس دومی که از “ganache” دریافت کرده بودیم را قرار می‌دهیم تا با استفاده از این تابع مشخص کنیم مبلغ 5 اتر به این آدرس بدهکار هستیم. در تصویر تایید تراکنش را مشاهده میکنیم همچنین میتوانیم جزئیات تراکنش را نیز مشاهده کنیم.

در این قسمت با استفاده از تابع “lookup” می توانیم با مشخص کردن دو آدرس، مقدار بدهی که بین آن ها وجود دارد را مشاهده کنیم.

The screenshot shows a web application interface for looking up an IOU. The 'lookup' button is selected. The main area displays a form with the following fields:

- debtor: 0x9f7eEE88cb06cA17a2d50aead412210086209A
- creditor: 0x389e281D900CB9E513643D15bd71b3DE15A37630

The 'call' button is highlighted. Below the form, the result is displayed as '0: int32: ret 5'.

همچنین جزئیات آن نیز قابل مشاهده است.

The screenshot shows a web application interface for looking up an IOU. The 'lookup' button is selected. The main area displays a form with the following fields:

- debtor: 0x9f7eEE88cb06cA17a2d50aead412210086209A
- creditor: 0x389e281D900CB9E513643D15bd71b3DE15A37630

The 'call' button is highlighted. Below the form, the result is displayed as '0: int32: ret 5'.

با استفاده از تابع “addDebt” می توانیم برای شخصی به غیر از کسی که در حال حاضر وارد حساب شده ، تراکنش بدهی ایجاد کنیم. به این صورت که آدرس شخص بدهکار و بستانکار و همچنین مبلغ بدهی را برای تابع مشخص میکنیم و تراکنش را به انجام می رسانیم.

addDebt

debtor:


0x6Bf5254049ddb4b0269B42

creditor:

0x91A541a6917bFD51E1A93f

amount:


3

 **transact**

همچنین برای مشاهده ی جزئیات تراکنش، خروجی تراکنش تایید شده را گسترش میدهیم:

✓ [block:3 txIndex:0] from: 0x91A...19017 to: Splitwise.addDebt(address,address,int32) 0x2fc...d268C value: 0 wei data: 0xfe9...00003 logs: 0 hash: 0x12b...0cc3a

Debug

status	true Transaction mined and execution succeed
transaction hash	0x12b6db70632eba9fa3ce04ad687d723ab67cb655f391af5f9d98c758eba0cc3a 
from	0x91A541a6917bFD51E1A93fb87faaB27d83819017 
to	Splitwise.addDebt(address,address,int32) 0x2fce2A3D1B3fD421B4fe77bCA9fa2686C76d268C 
gas	108461 gas 
transaction cost	108461 gas 
hash	0x12b6db70632eba9fa3ce04ad687d723ab67cb655f391af5f9d98c758eba0cc3a 
input	0xfe9...00003 
decoded input	{ "address debtor": "0x6Bf5254049ddb4b0269B42A6D8153459ce19DbcB", "address creditor": "0x91A541a6917bFD51E1A93fb87faaB27d83819017", "int32 amount": 3 } 
decoded output	- 

همچنین با استفاده از تابع “getUsers” می توانیم کاربرانی که در شبکه، تراکنش های بدهی به هم زده اند را مشاهده کنیم.

getUsers

0: address[]: ret 0x389e281D900CB9E513643D15bd71b3DE15A37630,0x9f7eEE88cb06cA17aA2d50AeaDa412210D86209A,0x91A541a6917bFD51E1A93fb87faaB27dB3819017,0x6Bf5254049ddb4b0269B42A6D8153459ce19DbcB

در ادامه برای استفاده از توابع قرارداد از برنامه ی کلاینتی که نوشته ایم استفاده میکنیم. برای استفاده از آن باید آدرس قرارداد “deploy” شده را به فایل “script.js” بدهیم. بنابراین در کد برنامه در قسمت زیر آدرس قرارداد را که از “remix” گرفته بودیم قرار میدهیم:

```
var contractAddress = '0xED800e31ec623bC7456fb4CA75C444D9607f883a'
```

سپس فایل “index.html” را باز کرده و مشاهده میکنیم که آدرس های ایجاد شده از “ganache” در “wallet addresses” در آن وجود دارند. همچنین قسمتی برای استفاده از تابع “add_IOW” و قسمتی از صفحه اکانت فعلی برنامه و قسمت دیگر کاربرانی که به یکدیگر تراکنش بدهی زده اند نشان داده می شود.

Blockchain Splitwise

Add IOU

Address of person you owe:

Amount you owe them:

Add IOU

Users

My Account

0x355df90f9663d078af36d72b35f93371dd89aebc

Total Owed: \$0

Last Activity: unknown

Wallet Addresses

- 0x355df90f9663d078af36d72b35f93371dd89aebc
- 0x160776071548630d9ce101211d3b554e3b0fe
- 0x040e811c04648af600002776a040c76121d1de
- 0x040e811c04648af600002776a040c76121d1de
- 0x040e811c04648af600002776a040c76121d1de
- 0x040e811c04648af600002776a040c76121d1de
- 0x040e811c04648af600002776a040c76121d1de
- 0x040e811c04648af600002776a040c76121d1de
- 0x040e811c04648af600002776a040c76121d1de
- 0x040e811c04648af600002776a040c76121d1de

در این قسمت برای این که نشان دهیم مشکل وجود حلقه های بدهی بر طرف شده است تراکنش هایی را به قرار زیر انجام می دهیم:

به عنوان مثال اگر $A \rightarrow B$ 15 و $B \rightarrow C$ 11 نگاه اگر $C \rightarrow A$ 16 بخواند تراکنش $C \rightarrow A$ را اضافه کند لازم است که نرم افزار ما گراف را به صورت زیر تغییر دهد.



بنابراین در ابتدا از آدرس اول به آدرس دوم تراکنش بدهی ای به مقدار 15 اتر را انجام می دهیم:

Blockchain Splitwise

Add IOU

Address of person you owe:

0x16bf7fd0f15f49930d9ce1021d11d3b5b5a08d

Amount you owe them:

15

Add IOU

Users

My Account

0x355df90f9663d078af36d72b35f93371dd89abeb

Total Owed: \$0

Last Activity: unknown

Wallet Addresses

- 0x355df90f9663d078af36d72b35f93371dd89abeb
- 0x16bf7fd0f15f49930d9ce1021d11d3b5b5a08d
- 0x4b0e81c0b045f9c000273ea306c7581c151d8
- 0x404c55f95950f7f7076ac5a35394b044b493
- 0x6cb7f9f9c0790ab06166d9f72ba7401130932
- 0x6af0724200504e6d7913984d1819047183792
- 0x7aaacae6134832d95d47d58a78d06c232d584
- 0x11729b720477804050f6785a36d2c9380b7139
- 0x07f9a05d8864c5e1139ae9fad9fb04907d57
- 0x0789f9eae33d6739578814a40640675ae3294a

با انجام این تراکنش مبلغ بدهی آدرس اول برابر با 15 می شود:

Blockchain Splitwise

Add IOU

Address of person you owe:

Amount you owe them:

Add IOU

Users

- 0x16bf7fd0f15f49930d9ce1021d11d3b5b5a08d
- 0x355df90f9663d078af36d72b35f93371dd89abeb

My Account

0x355df90f9663d078af36d72b35f93371dd89abeb

Total Owed: \$15

Last Activity: unknown

Wallet Addresses

- 0x355df90f9663d078af36d72b35f93371dd89abeb
- 0x16bf7fd0f15f49930d9ce1021d11d3b5b5a08d
- 0x4b0e81c0b045f9c000273ea306c7581c151d8
- 0x404c55f95950f7f7076ac5a35394b044b493
- 0x6cb7f9f9c0790ab06166d9f72ba7401130932
- 0x6af0724200504e6d7913984d1819047183792
- 0x7aaacae6134832d95d47d58a78d06c232d584
- 0x11729b720477804050f6785a36d2c9380b7139
- 0x07f9a05d8864c5e1139ae9fad9fb04907d57
- 0x0789f9eae33d6739578814a40640675ae3294a

سپس از آدرس دوم به آدرس سوم تراکنش بدهی ای به مقدار 11 اتر را انجام می دهیم:

Blockchain Splitwise

Add IOU

Address of person you owe:

0x5ebde81cbd645ef8c0600b273ea36dc7561c8

Amount you owe them:

11

Add IOU

Users

- 0x16bf7fd0f15f49930d9ce1021d11d3b5b5a08d0e
- 0x355df90f9663d078af36d72b35f93371dd89abeb

My Account

0x16bf7fd0f15f49930d9ce1021d11d3b5b5a08d0e

Total Owed: \$0

Last Activity: unknown

Wallet Addresses

- 0x355df90f9663d078af36d72b35f93371dd89abeb
- 0x16bf7fd0f15f49930d9ce1021d11d3b5b5a08d0e
- 0x5ebde81cbd645ef8c0600b273ea36dc7561c81de
- 0x5402ec33f95d9d7678ac5a3b595b4d44a493
- 0x6cb7f99a0778bba91616a69f72ba740113050c
- 0xc4a8f7242009a46d79159fa4d818ac471837d2
- 0x7aaacee134892496b47d583a78d56ce132308d
- 0x11729b720a7780440f9765a36d2c95b0b7189
- 0xd07f9a8048846c5e1139ceff8a9f8b4e07c27
- 0x4789f9ea935d739f78814a49f458674a3294a

با انجام این تراکنش مبلغ بدهی کاربر دوم نیز به 11 تغییر میکند:

Blockchain Splitwise

Add IOU

Address of person you owe:

Amount you owe them:

Add IOU

Users

- 0x16bf7fd0f15f49930d9ce1021d11d3b5b5a08d0e
- 0x355df90f9663d078af36d72b35f93371dd89abeb
- 0x5ebde81cbd645ef8c0600b273ea36dc7561c81de

My Account

0x16bf7fd0f15f49930d9ce1021d11d3b5b5a08d0e

Total Owed: \$11

Last Activity: unknown

Wallet Addresses

- 0x355df90f9663d078af36d72b35f93371dd89abeb
- 0x16bf7fd0f15f49930d9ce1021d11d3b5b5a08d0e
- 0x5ebde81cbd645ef8c0600b273ea36dc7561c81de
- 0x5402ec33f95d9d7678ac5a3b595b4d44a493
- 0x6cb7f99a0778bba91616a69f72ba740113050c
- 0xc4a8f7242009a46d79159fa4d818ac471837d2
- 0x7aaacee134892496b47d583a78d56ce132308d
- 0x11729b720a7780440f9765a36d2c95b0b7189
- 0xd07f9a8048846c5e1139ceff8a9f8b4e07c27
- 0x4789f9ea935d739f78814a49f458674a3294a

در تراکنش سوم می خواهیم تراکنش بدهی ای به مقدار 16 اتر را از آدرس سوم به آدرس اول انجام دهیم. در این حالت برنامه باید دور تشکیل شده در گراف را تشخیص دهد و حلقه ی ایجاد شده در گراف را از بین ببرد. برای این کار لازم است که کوچکترین پال در حلقه پیدا شود و مقدار آن از تک تک پال ها کم شود به صورتیکه یکی از پال ها به ارزش صفر برسد. بنابراین با انجام این تراکنش بدهی کاربر دوم به سوم حذف می شود و مقدار بدهی کاربر اول به دوم به 4 اتر تغییر میکند. همچنین با این کار بدهی اصلی آدرس سوم به اول نیز برابر 5 اتر می شود:

Blockchain Splitwise

Add IOU

Address of person you owe:

0x355df90f9663d078af36d72b35f93371dd89ab

Amount you owe them:

16

Add IOU

Users

- 0x16bf7fd0f15f49930d9ce1021d11d3b5b5a08d0e
- 0x355df90f9663d078af36d72b35f93371dd89abeb
- 0x5ebde81cbd645ef8c0600b273ea36dc7561c81de

My Account

0x5ebde81cbd645ef8c0600b273ea36dc7561c81de

Total Owed: \$0

Last Activity: unknown

Wallet Addresses

- 0x355df90f9663d078af36d72b35f93371dd89abeb
- 0x16bf7fd0f15f49930d9ce1021d11d3b5b5a08d0e
- 0x5ebde81cbd645ef8c0600b273ea36dc7561c81de
- 0x4242c53f995f99f7878ac5b3394a4a9a93
- 0x66cb75f9ac77d0ba9e1e6c9f728a74013b092c
- 0xab9f7242030e86a791395ad28159a471817f2
- 0x7a9a9ee1548d2d98a78883a78596ac05280a4
- 0x11728672da471804a5a9b788a36dc9f8a9b7199
- 0xd07794dc08f8dc1ae113a9ff8a99f9e971d57
- 0xd776f5eaf39d739778814a406428679a3294a

My Account

0x355df90f9663d078af36d72b35f93371dd89abeb

Total Owed: \$4

Last Activity: unknown

My Account

0x5ebde81cbd645ef8c0600b273ea36dc7561c81de

Total Owed: \$5

Last Activity: unknown

بنابراین مبالغ بدهی های هر سه آدرس به قرار زیر می شود:

My Account

0x355df90f9663d078af36d72b35f93371dd89abeb ▼

Total Owed: \$0

Last Activity: unknown

My Account

0x16bf7fd0f15f49930d9ce1021d11d3b5b5a08d0e ▼

Total Owed: \$0

Last Activity: unknown

My Account

0x5ebde81cbd645ef8c0600b273ea36dc7561c81de ▼

Total Owed: \$5

Last Activity: unknown

امتیازی

در این بخش برای کامپایل کردن و ثبت کردن قرارداد بر روی شبکه بلاک چین به جای استفاده از سایت “remix” از ابزاری به نام “truffle” استفاده میکنیم که فرایند قبل را برای ما راحت تر و هوشمندانه تر انجام می دهد.

نحوه ی کار ترافل به این شکل است که لازم است سه پوشه با نام های “contracts”، “migrations” و “test” در پوشه اصلی که پروژه ما در آن است موجود باشد. در پوشه اول کدهای “solidity” قرار میگیرد، قراردادهای اصلی که لازم است کامپایل و ثبت شوند. در پوشه دوم و سوم فایل های به فرمت .js قرار میگیرند که اصطلاحا اسکریپت نام دارند.

با اجرای دستور “truffle compile”، قرارداد های موجود در پوشه اول کامپایل می شوند و یک پوشه به نام “build” ساخته می شود و چکیده قراردادها به فرمت .json در آن ثبت می شود.

```
F:\docs\term8\CryptoCurrency\projects\project3\splitwise\truffle>truffle compile

Compiling your contracts...
=====
> Compiling .\contracts\Migrations.sol
> Compiling .\contracts\Splitwise.sol
> Artifacts written to F:\docs\term8\CryptoCurrency\projects\project3\splitwise\truffle\build\contracts
> Compiled successfully using:
   - solc: 0.5.16+commit.9c3226ce.Emscripten.clang
```

با اجرای دستور “truffle migrate” اسکریپت های موجود در پوشه دوم به ترتیب اسم توسط “Node.js” کامپایل و اجرا می شوند، این اسکریپت ها به نحوی نوشته می شوند که با اجرای آن ها، قرارداد اصلی موجود در پوشه اول کامپایل و در صورت درستی در شبکه ثبت شوند. در واقع هدف اسکریپت های موجود در پوشه دوم انجام کاری است که ما تا الان با “remix” انجام میدادیم.

قبل از اجرای این دستور باید “ganache-cli” را اجرا کرده باشیم.

```
F:\docs\term8\CryptoCurrency\projects\project3\splitwise\truffle>truffle migrate

Compiling your contracts...
=====
> Everything is up to date, there is nothing to compile.

Starting migrations...
=====
> Network name:      'development'
> Network id:        1626021505309
> Block gas limit: 6721975 (0x6691b7)

1_initial_migration.js
=====

Deploying 'Migrations'
-----
> transaction hash:  0x46cffffe129a93b8bcf89ffe8699ff6e0f38daf5a7c504c4ada591eecee0084e8
> Blocks: 0         Seconds: 0
> contract address: 0x7178F3F563c6C2e4f93274532E9D3eB8cD3F98a4
> block number:     1
> block timestamp:  1626021522
> account:          0x6ebb6c227f9B1DC20b654f687E9b4bE8C0193435
> balance:          99.99663428
> gas used:         168286 (0x2915e)
> gas price:        20 gwei
> value sent:       0 ETH
> total cost:       0.00336572 ETH

> Saving migration to chain.
> Saving artifacts
-----
> Total cost:       0.00336572 ETH
```

در ادامه ی آن اسکرپیت های مربوط به فایل دوم اجرا می شود:

```
2_deploy_contracts.js
=====
Deploying 'Splitwise'
-----
> transaction hash: 0x9f45f995a4f26d433b2487ad50bf8d3f4906d70ded88c201a394bc80a3863649
> Blocks: 0
> contract address: 0x8FCc197e342a073ac9Ae4C4a6a30663404821dB8
> block number: 3
> block timestamp: 1626021523
> account: 0x6ebb6c227f9B1DC20b654f687E9b4bE8C0193435
> balance: 99.99020408
> gas used: 279231 (0x442bf)
> gas price: 20 gwei
> value sent: 0 ETH
> total cost: 0.00558462 ETH

> Saving migration to chain.
> Saving artifacts
-----
> Total cost: 0.00558462 ETH

Summary
=====
> Total deployments: 2
> Final cost: 0.00895034 ETH
```

در ادامه تست هایی که برای توابع موجود در قرارداد هوشمند هستند را میبینیم

یک تست برای تابع “add_IOW” و یک تست برای تابع “addDebt” به صورت زیر:

```
const Splitwise = artifacts.require("Splitwise");

contract('Splitwise', (accounts) => {

  it("should add 100 credits from accounts[0] to accounts[1] with add_IOW", function() {
    return Splitwise.deployed().then(function(instance) {
      console.log("account 1 : ", accounts[1].valueOf());
      instance.add_IOW(accounts[1], 100);
      return instance.lookup(accounts[0], accounts[1])
    }).then(function(balance) {
      console.log(+balance.valueOf());
      assert.equal(balance.valueOf(), 100, "100 wasn't in the account 1");
    });
  });

  it("should add 50 credits from accounts[1] to accounts[2] with addDebt", function() {
    return Splitwise.deployed().then(function(instance) {
      instance.addDebt(accounts[1], accounts[2], 50);
      return instance.lookup(accounts[1], accounts[2])
    }).then(function(balance) {
      assert.equal(balance.valueOf(), 50, "50 wasn't in the account 2");
    });
  });

});
```

اجرای دستور “truffle test” توابع موجود در فایل تست را صدا می زنند و خروجی های آن ها را نمایش می دهند.

همچنین با کامپایل و مستقر شدن قرارداد هوشمند بر روی بلاکچین با قرار دادن آدرس قرارداد در فایل “script.js” می توانیم توابع قرارداد را از طریق برنامه ی کلاینتی که داشتیم تست کنیم.