Smart Contract Account Book

Yuxian Chen, yc3840, Zhuoyu Feng, zf2272, Jinxuan Tang, jt3302, Zixuan Xiang, zx2366

May 13, 2022

Abstract

Combining the basic concept of smart contract and Ethereum introduced on lecture four and five, we are going to build a decentralized application (DAPP) called Smart Contract Account Book for our project. The basic functions of our DAPP are one-to-one accounting, billing displaying and the transaction simplification, which can directly display the amount of money of one user owing to other users, and moreover provide a simplified approach for all the users to pay their debts in minimum number of transaction. This project is not only a practice of smart contract coding but also a practical application called Splitwise implemented on Ethereum based on decentralized concept. Our project code is available on https://github.com/RainFZY/Smart-Contract-Account-Book.

1 Introduction

1.1 Basic Concept

1.1.1 Blockchain

The Blockchain is a growing list of blocks which is used for decentralized data management. Each block has a cryptographic hash of the previous block, a timestamp, and transaction data.[1] As each block contains the information of the previous blocks, they form a chain. That is why it is called BlockChain. The main features of Blockchain are its decentralization, security and data consensus. It is the nodes in the net rather than any third-party organizations to control the Blockchain, it is hard to alter the data in the block by easily changing some information in the block and it needs the majority nodes' validation to be accepted by the chain.

1.1.2 Ethereum

The main improvement of Ethereum is to create an alternative protocol for building decentralized applications, supporting fast-pace development time, promising safety for the small and rarely used applications, and allowing short interact time with different applications. Ethereum accomplishes these by constructing a blockchain with a built-in Turing-complete programming language, allowing anyone to write smart contracts and decentralized applications where they can create their own arbitrary rules for ownership, transaction formats and state transition functions.[2]

1.1.3 Dapp

Dapps consist of the frontend part which can be written in any language and the backend part which is the smart contracts. Because Dapps are built based on Blockchain technology, Dapps look like normal apps but have some special and important features. Once it has been deployed to the blockchain, it has no owners and everyone can use the Dapp's feature. It can not be shutted down only if the smart contracts itself turn down the Dapp. Anyone can use the most Dapps anonymously which dapps only need your blockchain account and a wallet[3].

1.1.4 Solidity

Our smart contracts are written by Solidity. Solidity is an object-oriented, high-level language for implementing smart contracts. It used to be partly influenced by JavaScript because of its function-level scoping of variables. It is now most profoundly influenced by C++ which can be seen in the syntax for variable declarations, loops and the concept of overloading functions[4].

1.1.5 Remix

Our smart contracts are deployed at the Remix IDE. Remix IDE is an open source web and desktop application. It fosters a fast development cycle and has a rich set of plugins with intuitive GUIs. Remix IDE has modules for testing, debugging and deploying of smart contracts. It is written in JavaScript and supports both usage in the browser, but run locally and in a desktop version.

1.2 Smart Contract Account Book

Our project aims to realize the one-to-one accounting, billing display and the transaction simplification. One-to-one accounting helps users to record their bills with the specific account. Billing display shows the billing information and the corresponding accounts. Debt simplification shows the shortest path that can pay the bills for all the accounts.

2 Originality/Novelty

We will show the originality of our project of the three main functions of our DAPP.

2.1 One-to-One Accounting

This function is the most basic function of our Smart Contract Account Book DAPP, which is also the foundation of the other functions. This basic one-to-one accounting enables users to clearly record the debt amount to every other user, which the information is stored as a dictionary in the smart contract. The core one-to-one accounting function leads to the other import function which is Billing Display and Debt Simplification.

2.2 Billing Display

This is a original work on the UI designing of our DAPP. We store the billing information after every transaction, record the units of the object user owing to other users. The billing information can be shown on the top-right corner of the UI interaction interface of the lists of users and the units that the object user owing to. This original work of ours makes it easier to have an overview of the billing information.

2.3 Debt Simplification

This is the an idea from the Splitwise's debt simplification feature and we implement it on our Smart Contract Account Book DAPP. The purpose of this function is to find a simplest strategy for every users to pay their debts in minimum number of transaction. Behind this function, we implemented the Maximum-Flow algorithm of Graph Theory on smart contract solidity coding, where the main idea is to iterate the pair of maximum creditor and debtor as the source and sink of graph, compute the maximum cash flow between them and replace the residual graph as the new graph. The challenge of this function is the implementation on the smart contract solidity coding. With this function the DAPP will be more practical in the real world using.

3 Showcase of Smart Contract Account Book

3.1 Showcase by video

The showcase video of how to use our DAPP is available on YouTube https://youtu.be/az5OQGlYjJU.

3.2 Showcase by figures

In the report, we will show the function of one-to-one accounting, billing display and the transaction simplification.

First, figure 1 shows the UI at the beginning when we initialize the DAPP.

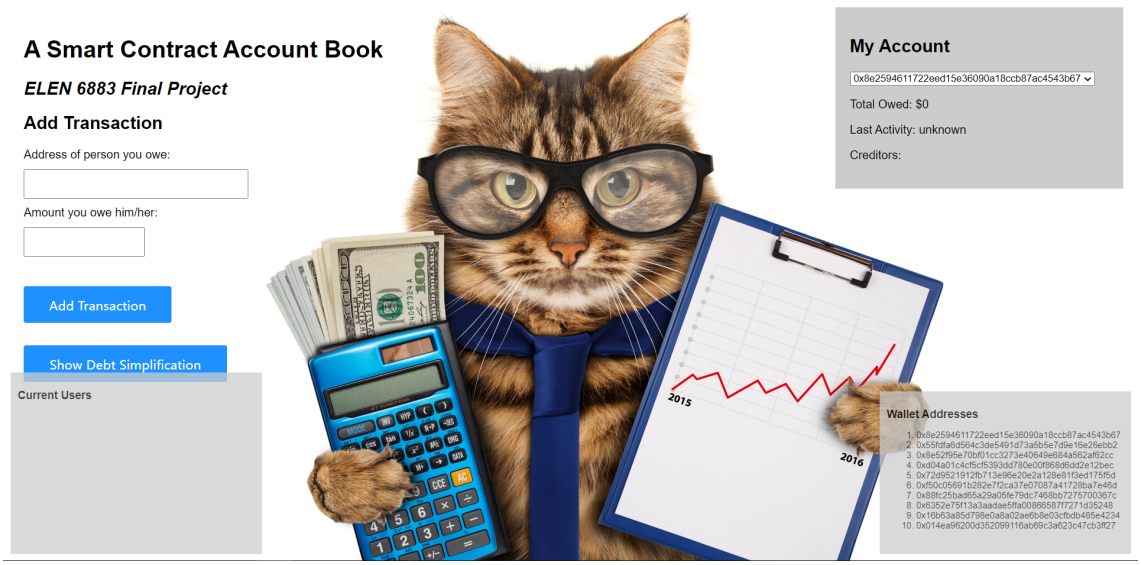

Fig1: Initial UI

Now we assume the user 1 owes user 2 20 units (can be any token), owes user 3 30 units, owes user 4 40 units, therefore user 1 owes other users 90 units in total. The Account Book then will show the one-to-one accounting on the top-right corner, which is also the billing displaying, shown in the figure 2.

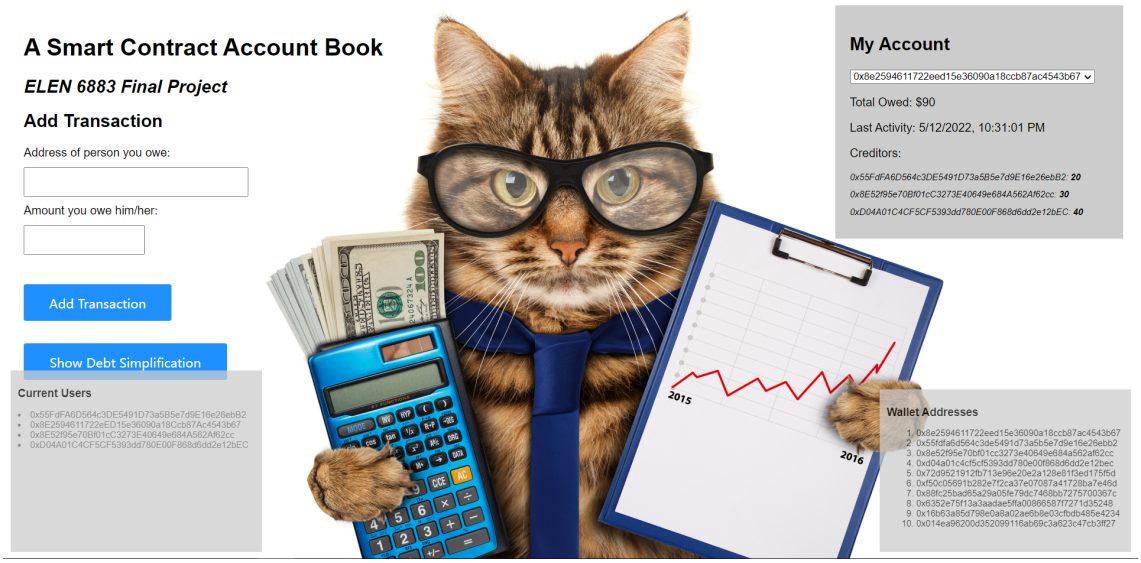

Fig2: The billing of user 1 displayed on UI

Next we assume user 2 owes user 3 20 units, owes user 4 30 units, owes user 5 40 units, therefore user 1 owes other users 70 units in total (subtracted the amount user 1 owing). Then the billing of user 2 displayed as figure 3.

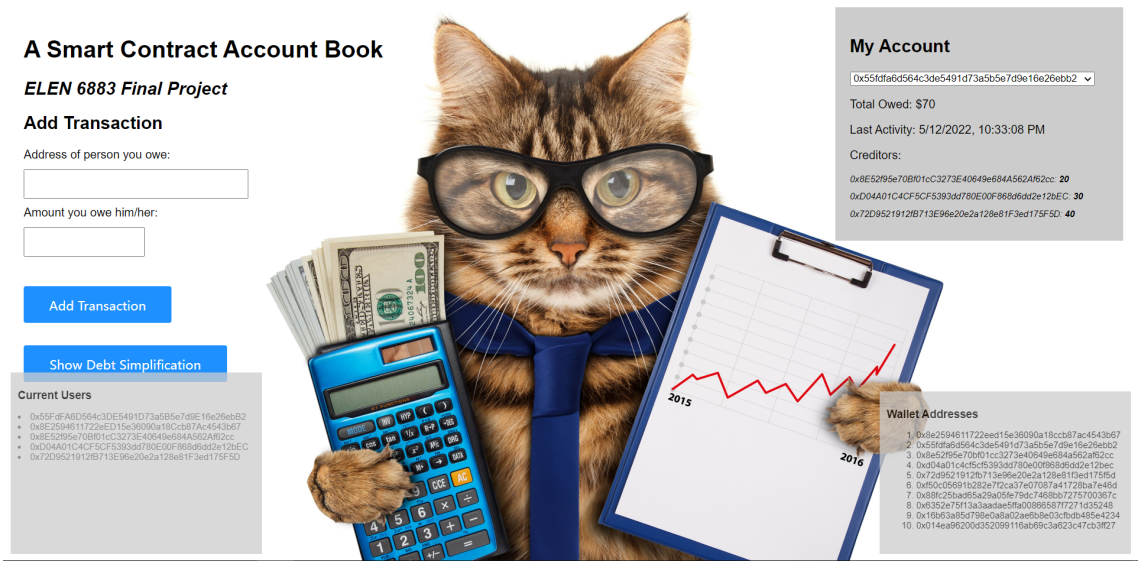

Fig3: The billing of user 2 displayed on UI

Then given the billing of user 1 and user 2, the simplified can be given by the DAPP shown in figure 4, which has the minimum number of transaction.

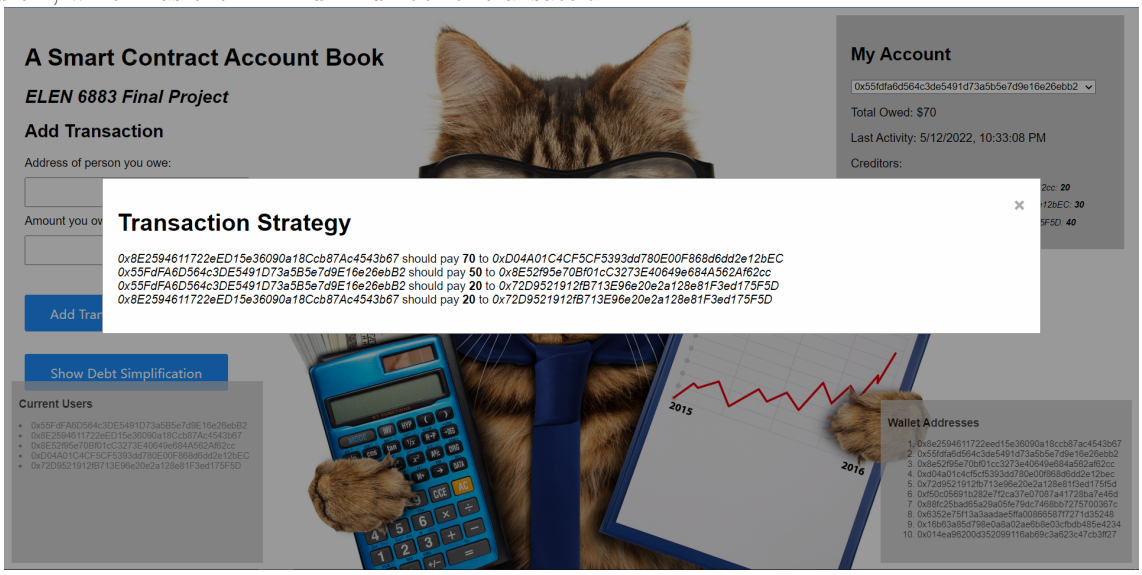

Fig4: The Simplified Debts

4 Conclusion

We use the Splitwise idea from the application on our phones and tried to implement on Ethereum using the decentralized application, and got the positive results from running our toy examples of ten-user transactions. We believe with these functions on our DAPP, we can achieve transparent and efficient transaction from one to another with complicated debt history. In the future, we hope to add other functions to our DAPP for instance share the expense to several participants and combining this new feature to billing display and transaction simplification. Furthermore we hope to implement our idea with these features on other chains for instance like CCN.

References

[1] "Blockchain," https://en.wikipedia.org/wiki/Blockchain, Apr 2022.

[2] V. Buterin *et al.*, "Ethereum white paper."

[3] "Decentralized applications (dapps)," https://ethereum.org/en/dapps/#what-are-dapps.

[4] "Solidity," https://docs.soliditylang.org/en/v0.8.13/.