Splitwiser

12-14.07.2024 ETHGlobal Brussels Hackathon
TUM Blockchain Club

Splitwiser **1**
Short description 1
Long description 1
Used tools 2
Brainstorming for tracks (Draft) 3
Workplan (from 12.07) 4
Debt simplification algorithm 5

Short description

Split your bills with Web3 power and Web2 ease. Settle the debts peer-to-peer directly in the app.

Long description

Imagine you're going on a trip with a group of friends. One friend pays for the taxi, another covers food, and so on. Your trip lasts seven days, and by the end, some friends might have paid for everything while others haven't paid at all. The sheer number of transactions makes tracking expenses challenging, as well as determining who owes what and to whom. This is where bill-tracking software like Splitwise or Tricount comes in, but they have their limitations. They don't allow you to pay your friends directly through the app, meaning you have to find your friend's PayPal or IBAN and trust that they've settled their bill. Additionally, applications like Splitwise have introduced their paid plans, which limit the number of groups and expenses you can track for free. Moreover, these applications are filled with annoying ads.

We are introducing Splitwiser, a solution to tackle peer-to-peer bill-tracking issues in traditional applications. By leveraging the power of web3 while maintaining the familiar workflows of web2, Splitwiser aims to onboard a new generation of users seamlessly. The smart contracts managing these actions are currently deployed on Ethereum and Base chains but are designed to be cross-chain, with plans to support more in the future. Respecting the transparent nature of web3, we are verifying our smart contracts on the Blockscout Explorer. The native currency is EURC by Circle, with more cryptocurrencies to be supported without additional conversion costs.

In Splitwiser, users can authenticate with both web2 and web3 methods. Once authenticated using the Dynamic system, users can create a new group (e.g., for an event) and add other participants. To simplify adding users, Splitwiser allows direct adding via wallet address and leverages The Graph queries and ENS for easy friend searches. After forming the group, its members can add expenses until the trip ends or when they need to settle the bill. Our debt simplification algorithm minimizes the number of transactions required by recursively matching the biggest debtors and creditors.

All the interactions with the smart contract until the actual debt settlement are free for the users because of the Safe Smart Wallet feature. This feature sets Splitwiser apart from current web2 solutions, as it allows for infinite groups, members, and expenses without any charges, thanks to gas sponsorship by the application. The modern and familiar interface ensures that even common users can use our application without encountering any web3 complexity.

Splitwiser introduces direct on-chain payments within the app to finalize settlements. The smart contract handles the deposits and withdrawals, so users either need to supply their sum debt into the smart contract or wait until the smart contract pays out the promised debt.

Splitwiser's goal is not just to replicate a web2 app on-chain but to demonstrate how a proper UI/UX, account abstraction, gas fee subsidy, and other tools can create applications that leverage web3's potential while maintaining the ease of use of web2 apps. The Splitwiser team, former active users of the Splitwise web2 application, is now switching to their solution. They aim for this project to be widely used, not just another hackathon project.

The roadmap for continuous development includes community governance to decide on future features, business models, and more. We plan to extend our payment method to credit cards and Fiat. We also want to support a wide range of cross-chain transactions, allowing users to freely and easily transact with each other. Splitwiser is ready to onboard the next generation of web3 users and stimulate the adoption of web3 applications worldwide.

Used tools

TLDR;

Backend and Frontend: Scaffold-ETH 2 (NextJs14 and Hardhat)
Smart contracts: Solidity

Backend and Frontend: Scaffold-ETH 2 https://scaffoldeth.io/ (NextJs14 and Hardhat)
Smart contracts: Solidity https://soliditylang.org/
Login: Dynamic https://www.dynamic.xyz/
Sponsored transactions & wallets: Safe smart wallets https://safe.global/
Usernames: ENS https://ens.domains/
User search: The Graph https://thegraph.com/ with ENS
Chains deployed: Sepolia Ethereum and Sepolia Base https://www.base.org/
Native currency: EURC https://www.circle.com/en/eurc

Detailed explanation:
The project was initially set up with the help of Scaffold-ETH 2 (https://scaffoldeth.io/) because of its wide range of functionalities, so we could focus on the business logic. The front end is a NextJS 14 web application that is developed for mobile screen view. Later in the future it is planned to support all screen sizes and to build a mobile app. For the backend, Hardhat (https://hardhat.org/) is used. The smart contract is written in Solidity and handles the functionality of group creation, user addition, expense addition, and debt settlement. It is designed to be cross-chain and is currently deployed on Sepolia Ethereum () and Sepolia Base. We have chosen the Base (https://www.base.org/) chain because of its speed, low cost, and ease of use. For verification and transparency of our smart contracts, we are utilizing Blockscout (https://www.blockscout.com/).

For login functionality, we have used Dynamic (https://www.dynamic.xyz/), which allows us to authenticate users using standard web2 methods such as email or social media SSO while offering web3 capabilities. With that, we were able to define two login paths, one for experienced web3 users and a second for web2 users. The Safe smart wallet (https://safe.global/) is used to allow safe and smooth usage of our application. With the help of the sponsored transaction feature, we can conveniently interact with our smart contracts (create groups, add expenses, etc.) free of charge for the users. For the user's view, we are using ENS domains (https://ens.domains/) to simplify the identification of certain people. We have used an ENS search query by The Graph (https://thegraph.com/) to search for new users and add them to groups not only via their address but also by their ENS name, which is considerably more error-prone and convenient. The Graph also accelerates our search and makes the search experience similar to known web2.

The native currency of the application is EURC by Circle (https://circle.com/), but more tokens are planned to be supported in the future. The current choice for this stablecoin is because of its adoption and legal compliance. EURC is MiCAR compliant, and since we thrive on mass adoption and ease of use for Splitwiser, we need a stable currency with which users can interact without any potential legal or volatility consequences.

Brainstorming for tracks (Draft)

Dynamic: Abstract away crypto and best socialfi
- Login for web2 and web3

Base: best corporate/business app UX

- fast, cheap chain. Use together with Ethereum

ENS: Best use of ENS
- Use ENS for profiles

The Graph: Best Use of Subgraph
- Simplify search of users

Safe: Best app integrating ERC-7579 Safe Smart Accounts
- Implement gas sponsoring system

Blockscout: Best use of Blockscout Block Explorer
- Use for linking of transactions
- Use for verification of transactions

Circle: ???
- EURC or USDC
- Use their dev tool stack

For more see: https://ethglobal.com/events/brussels/prizes/

Workplan (from 12.07)

TODOS:

Tonight:

Jesco & Dragos: basic Smart Contract Depoyed at base

Alex: **Pitch**

Yudhis & Fibi: Front end launch

Tomorrow:

- Integration of dynamic plus extra feature email login
- Creation of theam Ens on sepolia with real person picures.
- **Pitch**: add some topics to slides we must do (there we will also understand what we need to implement)
  - add explanation on algorithm to count final debts etc.
- Circle repo feedback in github issue(ask our or circle) + in our readme

Sunday:

- Recording of pitch (use of tools in cirlce)
- Check EF Price pool
- Feeback from circle https://circlefinancial.qualtrics.com/jfe/form/SV_0SZpLwIVQHmMXXM
- Readme (chalanges circle), Roadmap
- Live Demo test for one person in Deployment:
- Video Recoding of live demo

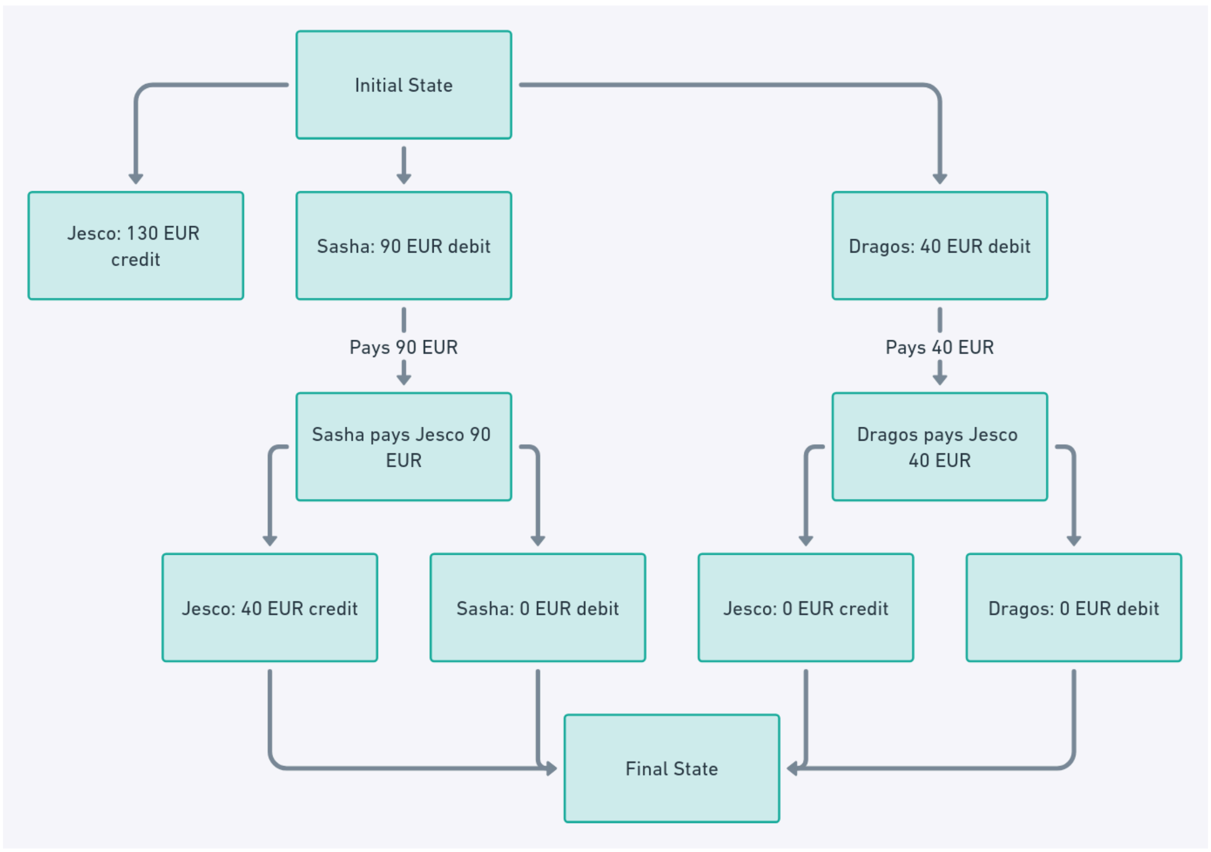
So overall we have 2 videos (live demo clicker and explanation video and pitching video)

Debt simplification algorithm

- Is used to minimize the amount of required transaction to cover all the debts
- Sustainable since less transactions needed

How does it work:

We sort by which person has borrowed the most and by which person has lended out the most. We match biggest creditor with biggest debtor. And continue like this.



Sources:
https://medium.com/@mithunmk93/algorithm-behind-splitwises-debt-simplification-feature-8ac485e97688
https://medium.com/@howoftech/how-does-the-splitwise-algorithm-work-dc1de5eaa371
https://stackoverflow.com/questions/15723165/algorithm-to-simplify-a-weighted-directed-graph-of-debts