



## Alpha

### Multipurpose Cryptocurrency Wallet Google Chrome Extension

Department of Software Engineering  
Capstone Project Phase B, 22-1-D-1

#### Supervisors

Alexander Keselman

#### Authors

Alla Davydov      317000669      [davydov.alla@gmail.com](mailto:davydov.alla@gmail.com)

Pavel Golikovski      320636574      [pavel.golikovski@gmail.com](mailto:pavel.golikovski@gmail.com)

#### Code Repository

<https://github.com/AceOfSwords/alpha>

## Table of Contents

<b>1</b>	<b>ABSTRACT .....</b>	<b>3</b>
<b>2</b>	<b>INTRODUCTION .....</b>	<b>3</b>
2.1	Motivation .....	3
2.2	Overview .....	4
2.3	Structure .....	4
<b>3</b>	<b>BACKGROUND AND RELATED WORK .....</b>	<b>4</b>
3.1	Blockchain .....	4
3.2	Mining .....	5
3.2.1	<b>Proof of Work .....</b>	<b>5</b>
3.3	Decentralization .....	6
3.4	Ethereum Blockchain .....	6
3.4.1	<b>Gas .....</b>	<b>7</b>
3.4.2	<b>Gas Limit .....</b>	<b>7</b>
3.5	Smart Contracts .....	8
3.6	Cryptocurrency Wallets .....	8
3.6.1	<b>Custodial and Non-Custodial Wallets .....</b>	<b>8</b>
3.6.2	<b>Hierarchical Deterministic (HD) Wallets .....</b>	<b>9</b>
3.6.3	<b>Derivation Paths .....</b>	<b>10</b>
3.6.4	<b>Mnemonic Phrases .....</b>	<b>10</b>
3.7	Decentralized Finances (DeFi) .....	10
3.7.1	<b>Infura .....</b>	<b>11</b>
3.7.2	<b>Nods .....</b>	<b>11</b>
3.7.3	<b>EVM .....</b>	<b>12</b>
3.8	Decentralized Lending and Borrowing .....	12
3.8.1	<b>Compound .....</b>	<b>13</b>
3.9	Decentralized Exchanges (DEX) .....	14
3.9.1	<b>Uniswap .....</b>	<b>14</b>
3.9.2	<b>dYdX .....</b>	<b>15</b>
3.10	Aggregators .....	15
3.11	Web3 .....	16
3.11.1	<b>Web2 vs. web3 .....</b>	<b>17</b>
3.12	Solid.js .....	17
3.12.1	<b>Solid.js vs React.js .....</b>	<b>17</b>
3.12.2	<b>Basic Reactivity .....</b>	<b>19</b>
3.13	Less .....	19
3.14	TypeScript .....	20
<b>4</b>	<b>EXPECTED ACHIEVEMENTS .....</b>	<b>20</b>
4.1	Product Infrastructure and Flow .....	20
4.2	Success Criteria .....	20
4.3	Unique Features .....	20
<b>5</b>	<b>ENGINEERING PROCESS .....</b>	<b>21</b>
5.1	Process .....	21
5.1.1	<b>Methodology .....</b>	<b>21</b>
5.1.2	<b>Research .....</b>	<b>21</b>
5.1.3	<b>Challenges .....</b>	<b>24</b>
5.2	<b>Product .....</b>	<b>24</b>
5.2.1	<b>Requirements .....</b>	<b>25</b>
5.2.2	<b>Architecture .....</b>	<b>26</b>
5.2.3	<b>API .....</b>	<b>26</b>

5.2.4	Flowcharts and Interface .....	28
5.2.5	Deployment .....	56
6	EVALUATION AND VERIFICATION .....	57
6.1	Testing .....	57
6.2	Evaluation .....	58
7	REFERENCES .....	59

# 1 Abstract

As of today, more and more decentralized financial services are being published to complement traditional financial practices. With that being said, many services come with many diverse user interfaces, varying user experiences and different learning curves.

In this project - **Alpha**, we want to assist novice and advanced Ethereum users alike to interface with the Ethereum Blockchain and DeFi services in a coherent and uniform manner by aggregating services such as Compound and Uniswap into a single local Google Chrome extension.

Using our browser extension users may manage their cryptocurrency assets in a built-in cryptowallet, perform transactions and use additional services such as Compound to borrow or lend money or convert it into other currency using Uniswap, *just like your own private banking service*, directly from within the extension.

Alpha incorporates external APIs that communicate with the Ethereum Blockchain via Smart Contracts to accomplish various financial operations, leveraging Blockchain's inherent architecture to substitute centralized databases in order to provide security and verifiable actions.

Our main goal is to contribute to the DeFi ecosystem and to build a better financial landscape made possible by the Internet and the Ethereum Blockchain.

# 2 Introduction

Traditional financial culture still mostly relies on centralized systems and third-party entities in order to perform monetary transactions that are inherently vulnerable. In turn, the centralized nature of such systems gave rise to the modern decentralized Blockchain technology.

## 2.1 Motivation

As Blockchain technologies evolve and increasingly integrate with more and more aspects of our lives, it brings with it a myriad of new issues that are being solved by a magnitude of different developers and parties, which in turn offer a multitude of varying solutions.

In practice, this in itself becomes a strenuous **problem for the end user**.

## 2.2 Overview

Publicly available decentralized finance (DeFi) applications (Dapps) tend to sacrifice user experience (UX) for unique visual fidelity. This forces users to learn to interact with new user interfaces (UI) and systems and creates user friction and unnecessary mental burden.

In this document we propose a browser extension for cryptocurrency owners that aggregates multiple DeFi services based on the extendible Ethereum blockchain technology (such as a wallet, exchanging, staking) into a simple, user-friendly and uniformly designed application.

## 2.3 Structure

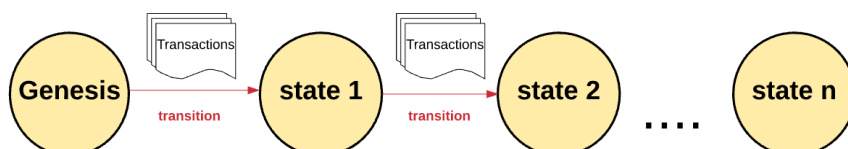
In this document we will further elaborate on the idea of the project and its general use cases, discuss the technologies we use, the way we approach design and our expectations.

# 3 Background and Related Work

## 3.1 Blockchain

A blockchain, is an ever-growing list of records, called blocks, that are linked through cryptographic means. Each block contains the cryptographic hash of the previous block, a timestamp, and transaction data. It is a database of transaction records that is distributed, verified, and maintained by a decentralized network of computers around the world, which means it works like a peer-to-peer network.

Compared to a conventional centralized database, the information cannot be manipulated due to the built-in distributed structure of the blockchain and the confirmed guarantees from the partners. In other words, the timestamp proves that the transaction data existed when the block was posted to get into its hash. Blocks contain the hash of the previous block, forming a chain, with each additional block reinforcing the previous ones. Consequently, blockchains are resistant to modification of their data, because after being written, the data in any given block cannot be changed retroactively without changing all subsequent blocks. Whenever someone completes a transaction, it is sent to the network and computer algorithms determine the authenticity of the transaction. After the transaction is confirmed, this new transaction is linked to the previous transaction, forming a chain of transactions. This chain is simply called a blockchain.



*Figure 1: A very simplistic overview of the way blocks are chained in the system.*

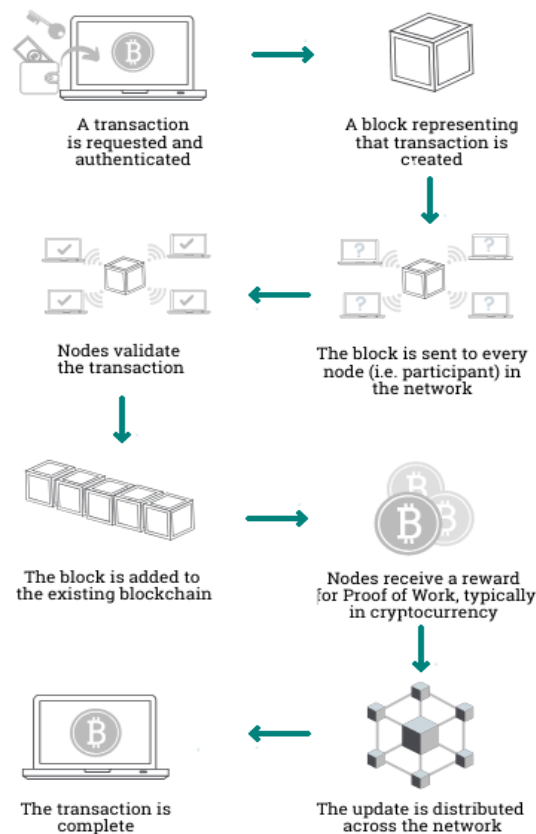


Figure 2: Blockchain transaction overview.

## 3.2 Mining

In decentralized systems like Ethereum, we need to make sure everyone agrees on the order of transactions. Miners help with this by solving computationally complex block-building puzzles, protecting the network from attacks.

Mining is the process of creating a block of transactions that will be added to the Ethereum blockchain.

Ethereum miners - computers running software - use their time and computing power to process transactions and create blocks.

### 3.2.1 Proof of Work

Any node on the network can declare itself as a miner and attempt to create and validate a block. Lots of miners from around the world try and create and validate blocks at the same time. Each miner provides a mathematical proof when submitting a block to the blockchain, and this proof acts as a guarantee: if the proof exists, the block must be valid.

For a block to be added to the main blockchain, the miner must prove it faster than any other competitor miner. The process of validating each block by having it miner provide a mathematical proof is known as a “proof of work”.

### 3.3 Decentralization

Most systems and services today are centralized. All data is stored in one object, and you must interact exclusively with this object to get the information you need. Centralized systems have served us well for many years, but they have several vulnerabilities such as being an easy target for potential hackers, updating a centralized system will stop the entire system, and shutting down at a centralized facility means no information is available. In a decentralized system, all data is distributed, so these vulnerabilities do not pose a threat. Also, in a decentralized system, no third party is required for interaction.

### 3.4 Ethereum Blockchain

The Ethereum Blockchain is an open source, decentralized blockchain with smart contract functionality. Ether (ETH) is the cryptocurrency of the Ethereum network. It is the second largest cryptocurrency by market capitalization after Bitcoin.

Ethereum is the most actively developed blockchain protocol as of today. The network allows developers to build and deploy decentralized applications (Dapps).

They are also stored on the Ethereum blockchain along with transaction records.

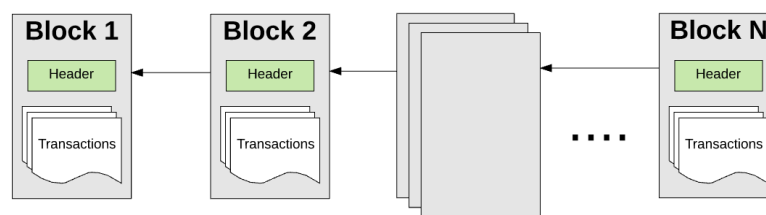


Figure 3

Dapps are open-source software using smart contracts and do not need an intermediary to operate.

### 3.4.1 Gas

Gas is required for the Ethereum network. It is the fuel that allows it to work just like a car needs gas to function.

One very important concept in Ethereum is the concept of fees. Every computation that occurs as a result of a transaction on the Ethereum network incurs a fee. This fee is paid in a denomination called “gas”.

Gas is the unit used to measure the fees required for a particular computation. Gas price is the amount of Ether you are willing to spend on every unit of gas and is measured in “gwei”. “Wei” is the smallest unit of Ether, where  $1^{-18}$  Wei represents 1 Ether. One gwei is 1,000,000,000 Wei.

### 3.4.2 Gas Limit

With every transaction, a sender sets a *gas limit* and *gas price*. The product of gas price and gas limit represents the maximum amount of Wei that the sender is willing to pay for executing a transaction.

For example, a sender sets the gas limit to 50,000 and gas price to 20 gwei. This implies that the sender is willing to spend at most  $50,000 \cdot 20 \text{ gwei} = 0.001 \text{ Ether}$  to execute that transaction.



Figure 4

Where does that gas money go. All the money spent on gas by the sender is sent to the beneficiary address, which is typically the miner’s address. Since miners expending the effort to run computations and validate transactions, miners receive the gas fee as a reward.

It is also important to note, the sender is refunded for any unused gas at the end of the transactions, exchanged at the original rate.

Inversely, if the sender runs out of gas, none of the gas is refunded to the sender.

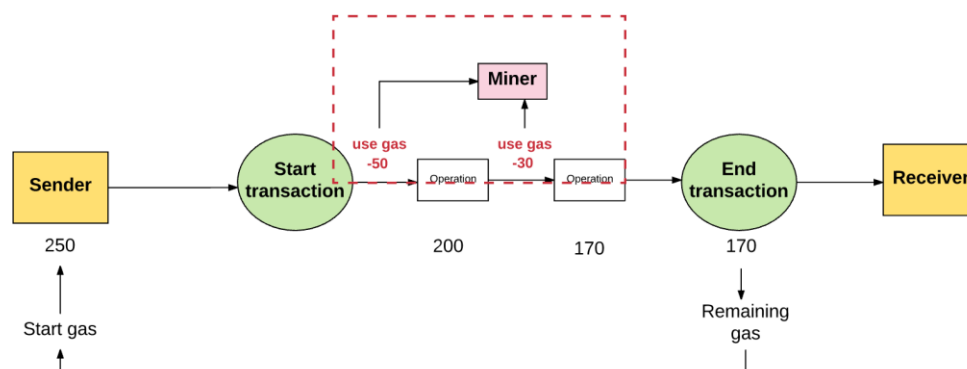


Figure 5: A flowchart depicting the way a transaction consumes gas.

### 3.5 Smart Contracts

Smart contracts are code scripts that facilitate the exchange of money, stocks, content, or anything of value. Smart contracts are formed using the Ethereum Virtual Machine (EVM). When a smart contract runs on the blockchain it acts like a self-acting computer program, without censorship, downtime, or the influence of third parties.

A smart contract works automatically according to the terms and conditions defined in its specific protocol - when the conditions of the transaction are met (for example, a specific date or a predetermined share price), the contract is executed immediately and automatically. An “if and then” mechanism that does not require external intervention. Both parties to the contract remain anonymous, but the contract itself is in the public network log (Blockchain) and is completely transparent - anyone can check the contract and get all the existing information about it. Smart contracts are written in Solidity, so technical knowledge is required to read the actual contract, however the information is available to everyone.

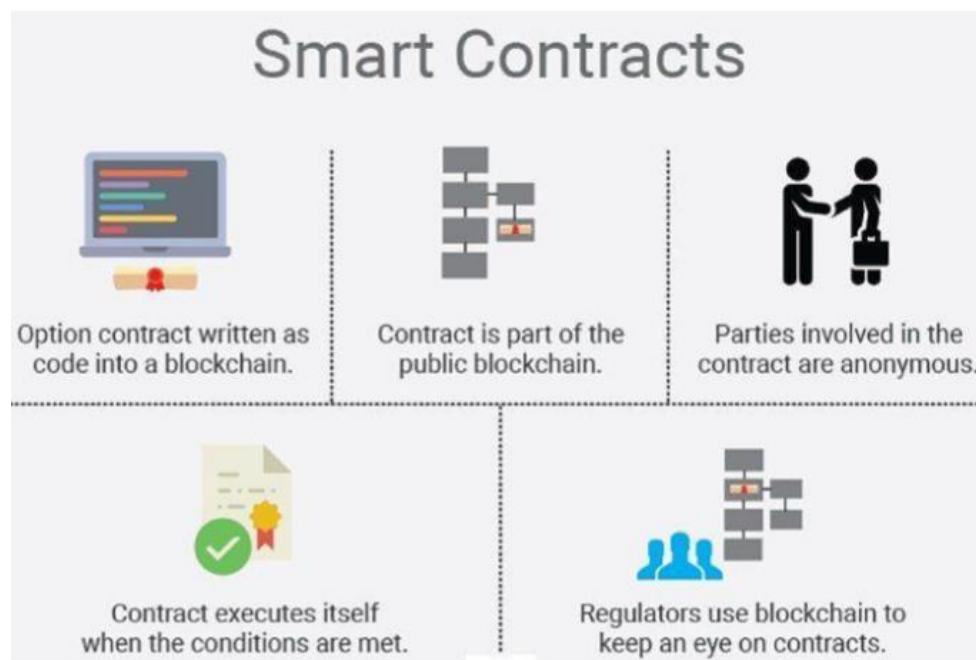


Figure 6

### 3.6 Cryptocurrency Wallets

A wallet is a user-friendly interface to the blockchain network. It manages your private keys, which are basically keys to the lock on your cryptocurrencies' vault. Wallets allow you to receive, store and send cryptocurrencies.

#### 3.6.1 Custodial and Non-Custodial Wallets

There are two kinds of wallets, custodial and non-custodial wallets.

Custodial wallets are wallets where third-parties keep and maintain control over your cryptocurrencies on your behalf. Non-custodial wallets are wallets where you take full control and ownership of your cryptocurrencies.



### **3.6.2 Hierarchical Deterministic (HD) Wallets**

HD Wallets are the most advanced type of deterministic wallet. They contain keys in a tree structure, in which parent keys can produce children keys, which can produce grandchildren keys, and so on, infinitely. The cryptocurrency holder can use the tree structure to organize transactions by type of transaction or by entity involved, such as departments or subsidiaries.

Like simple deterministic wallets, all HD wallets are created from a single master root seed, usually represented by a mnemonic word sequence, which makes it easier for account holders to transcribe and store. But HD wallets also offer the option of creating public keys without having to access the corresponding private keys. This means they can be used on insecure servers or in a receive-only mode.

### 3.6.3 Derivation Paths

If you use an HD wallet to store your crypto assets, you will encounter the phrase “derivation path.” Simply put, a derivation path defines a consistent method for generating the same set of accounts and wallets for a given private key even if you switch to another device or wallet provider.

To avoid confusion (e.g., using the same wallet address on multiple cryptocurrency networks), most popular cryptocurrency communities have established a conventional derivation path to be used exclusively by that cryptocurrency network. Some wallet providers have additional constraints, for example, on the Ethereum network, the Ledger and Ledger Live wallets use different derivation paths than the conventional Ethereum derivation path.

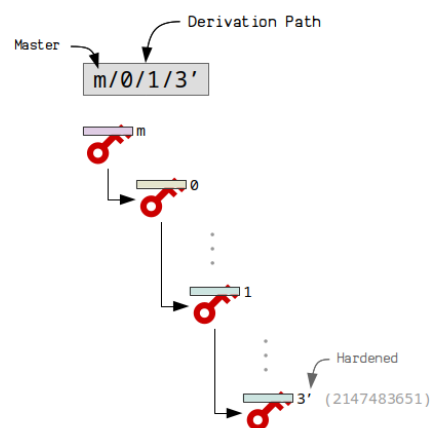


Figure 7: A derivation path includes a master key from which an infinite theoretical number of private keys can be derived.

### 3.6.4 Mnemonic Phrases

BIP39 or Bitcoin Improvement Proposal 39 is one of the many design ideas that was approved by an economic majority of the Bitcoin community and became a standard for many popular wallets.

BIP39 is the use of a mnemonic phrase – a group of easy to remember words – to serve as a back up to recover your wallet and coins in the event your wallet becomes compromised, lost, or destroyed. This is also known as a mnemonic seed, seed phrase, recovery phrase, wallet back up, master seed, etc.

These words aren’t just any words. They are pulled from a specific list of 2048 words known as the BIP39 wordlist. Upon start up, wallets that utilize the BIP39 standard will provide the user with a 12–24-word phrase randomly chosen from the wordlist.

## 3.7 Decentralized Finances (DeFi)

DeFi is currently one of the fastest growing sectors in the blockchain and cryptocurrency space. It is an ecosystem of decentralized applications (Dapps) that provide financial services built on distributed networks without any governing authority.

The DeFi sector is a collection of products and services that replace various institutions, including banking, insurance, bonds, and money markets. These financial services are

delivered through decentralized applications (Dapps), allowing users to combine their services (such as borrowing, lending and trading) to unlock many possibilities.

### 3.7.1 Infura

✚ Across the Ethereum network, utilities are needed to lower the barrier to entry and make it easier to access Ethereum data. Among the most important of these are Infrastructure as a Service (IaaS) products. Leading the way is Infura, which offers developers, application development teams, and enterprises across industries a suite of tools to connect their applications to the Ethereum network and other decentralized platforms.

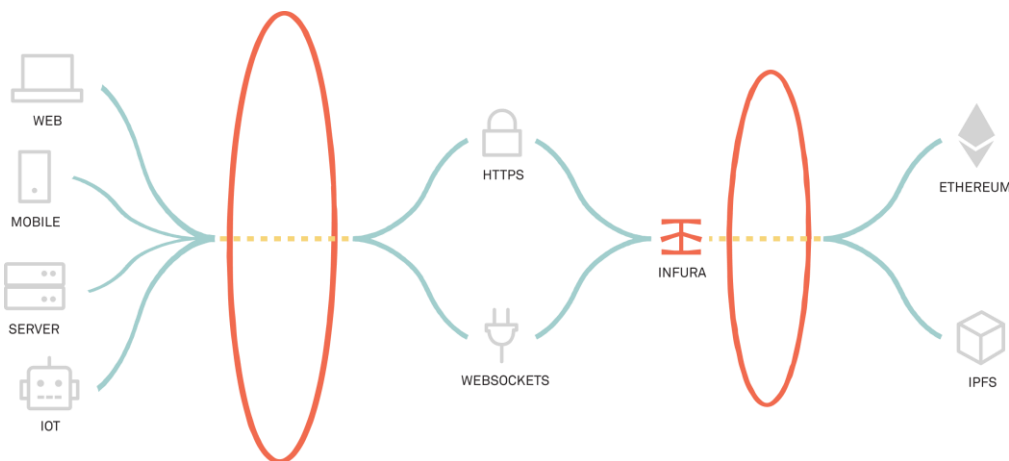


Figure 8: The Infura service is capable of interfacing between many technologies and the Ethereum Blockchain using HTTPS or Web Sockets.

Many of the most prominent Web3 projects - MetaMask, Aragon, Gnosis, our Alpha project and many more - use the Infura API to connect their applications to the Ethereum network. In doing so, Infura provides the fundamental infrastructure needed to handle both short-term spikes, which can often occur during token launches, and important long-term scaling solutions.

Infura Transactions (ITX) are a simplified way to send Ethereum transactions. ITX handles all edge cases for transaction delivery and takes care of transaction mining, eliminating the need for developers to tackle the complexities of gas management.

### 3.7.2 Nods

Real machines storing EVM state. Nodes interact with each other to propagate information about the EVM state and new state changes. Any user can also request code execution by submitting a code execution request from a node. The Ethereum network itself is a collection of all Ethereum nodes and their communications.

### 3.7.3 EVM

The Ethereum Virtual Machine is a global virtual computer, the state of which is maintained and agreed upon by each participant in the Ethereum network. The Ethereum protocol itself exists solely to support the continuous, uninterrupted, and unchanging operation of this special state machine; It is the environment in which all Ethereum accounts and smart contracts live. For any given block on the chain, Ethereum has one and only one "canonical" state, and the EVM is what defines the rules for calculating the new allowed state from block to block.

## 3.8 Decentralized Lending and Borrowing

One of the most common services offered by the financial industry is lending and borrowing. In the DeFi world, there are no such barriers as intermediaries since banks are no longer needed. With enough collateral, anyone can have access to capital and do whatever they want, anyone can contribute to Compound's decentralized liquidity pool from which borrowers can borrow and repay at an algorithmically determined interest rate.


Rates are adjusted automatically based on supply and demand.

Entrepreneurs can borrow the capital they need to start a business by collateralizing the business, and families can get a mortgage on a house that would otherwise be too expensive for them, using the house as collateral.

On the other hand, accumulated wealth can be lent and become capital for creditors.

This system reduces the risk that borrowers will flee with borrowed funds.

### 3.8.1 Compound

 *Compound* is an algorithmic money market protocol on Ethereum where anyone can supply or borrow cryptocurrencies frictionlessly against collateral, lets users earn interest or borrow.

Compound acts as a pool of liquidity.

Providers provide assets to the pool and receive interest, and borrowers receive loans from this pool and pay interest on their debts.

The interest rates are calculated for each asset using algorithms that take into account the current supply and demand of that asset.

Essentially, Compound reduces lending / borrowing friction by allowing lenders / borrowers to interact directly with the protocol on interest rates without having to negotiate loan terms.

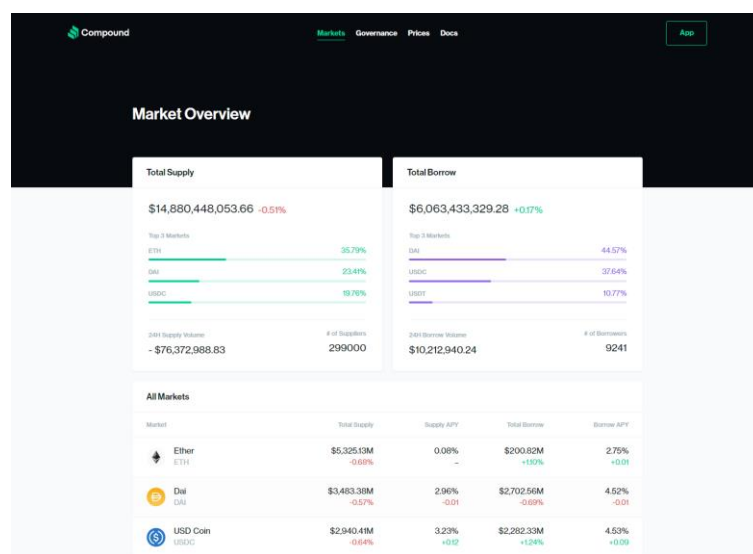


Figure 9: Compound web user interface.

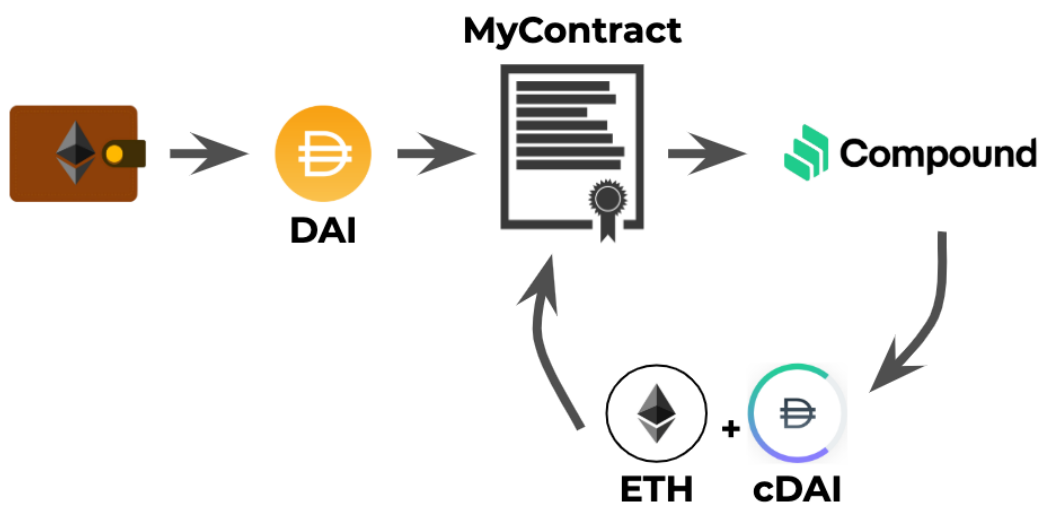


Figure 10: Compound process using cryptocurrency wallet

## 3.9 Decentralized Exchanges (DEX)


More and more people are aware of the risks and are turning to decentralized exchanges (DEX). Since users of centralized exchanges do not own their assets and there are cases of theft of registration data and, as a result, cryptocurrency.

DEX works using smart contracts and on-chain transactions to reduce or eliminate the need for an intermediary. Popular decentralized exchanges include Kyber Network, Uniswap, Dex.Blue, and dYdX.

Liquidity pools are essentially reserves of tokens in smart contracts, and users can instantly buy or sell tokens that are available in these pools. The token price is determined algorithmically and increases for large transactions. Liquidity pools can be used by several DEXs at once, which increases liquidity on each of these DEXs. Examples of DEXs based on liquidity pools are Kyber Network, Bancor and Uniswap.

In the case of DEX, users' assets can be stored in their own wallets.

### 3.9.1 Uniswap

 **Uniswap** Exchange is a decentralized token exchange protocol built on Ethereum that allows direct exchange of tokens without the need for a centralized exchange.

At Uniswap, you can simply exchange your tokens directly from your wallet.

All you have to do is send tokens from your wallet to the Uniswap smart contract address, and you will receive the desired token in return directly to your wallet. There is no order book, and the token exchange rate is determined algorithmically. All this is achieved through liquidity pools and an automatic market maker mechanism.

Liquidity pools are reserves of tokens that are in smart contracts.

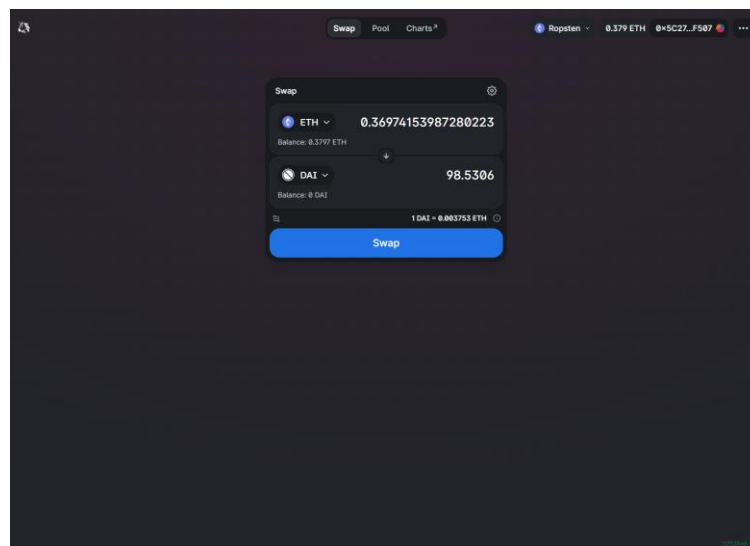


Figure 11: Uniswap web user interface.

### 3.9.2 dYdX

**X** *dYdX* is a decentralized exchange protocol for lending and borrowing and margin trading. It currently supports 3 assets - ETH, USDC and DAI. By combining an off-chain order book with on-chain settlement, the dYdX protocol seeks to create efficient, fair, and trustless financial markets that are not governed by any central authority.

dYdX runs on audited smart contracts on Ethereum, which eliminates the need to trust a central exchange while trading. The exchange combines the security and transparency of a decentralized exchange, with the speed and usability of a centralized exchange.

At first glance, dYdX has some similarities to Compound - users can deposit assets (lend) to receive interest and borrow assets (borrow) after posting collateral. However, dYdX goes one step further by introducing margin trading. dYdX allows you to trade ETH up to 5x leverage using DAI or USDC.

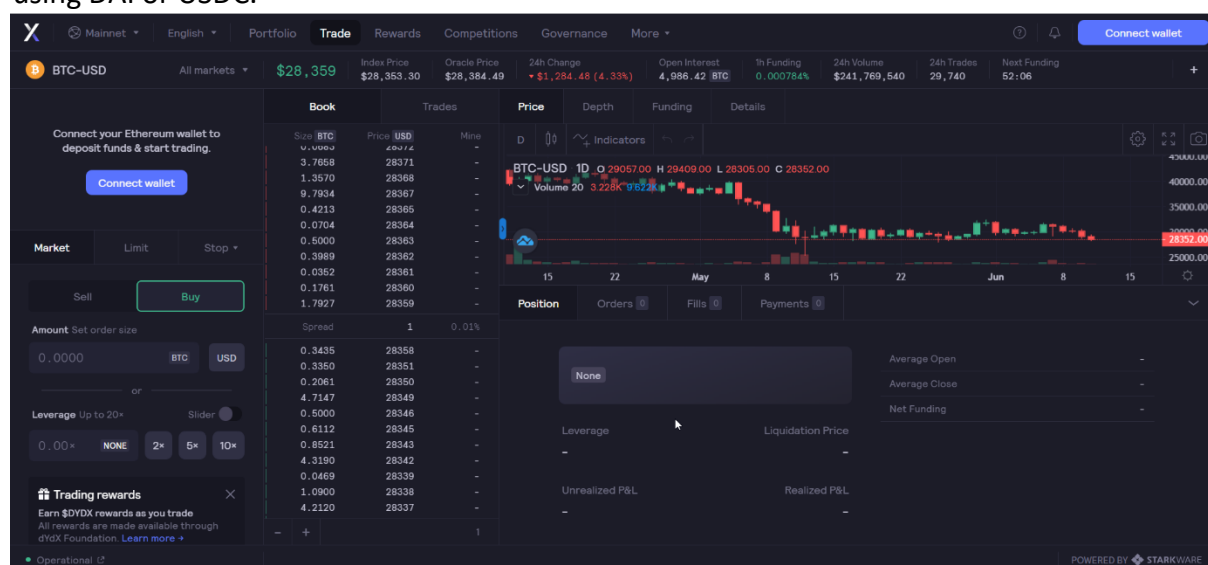


Figure 12: dYdX web user interface.

## 3.10 Aggregators

A DeFi aggregator brings together trades across various decentralized finance platforms into a single location, saving users time and increasing efficiency for cryptocurrency trades. As the name suggests, DeFi is spread out across different blockchains such as Ethereum and Binance Smart Chain. Within each blockchain is an ecosystem of isolated financial protocols.

DeFi aggregators siphon the very best prices from DEXs, lending services and liquidity pools into one place so that users can optimize their trades. Without an aggregator, users need to go to each platform on an individual basis to compare prices that will generate the best deal for them. Then, the user must manually execute each transaction using smart contracts. While this strategy may be fine for casual crypto trading, it severely limits those looking to implement advanced trading strategies.

Aggregators *put UX/UI at the forefront*, offering a far superior experience to the traditional way of manually interacting with liquidity layers. As a result, this helps users who are not as crypto savvy as trading experts navigate the world of DeFi with ease.

### 3.11 Web3

Web2 is the version of the Internet that most of us know today. An internet dominated by companies that provide services in exchange for your personal information. Web3 in the context of Ethereum refers to decentralized applications running on the blockchain. These are applications that allow anyone to participate without monetizing their personal data.

Status of Web3 Implementation in 2019:

Since Bitcoin launched the world's first blockchain in 2009, blockchain technology has improved in several key areas. As of mid-2019, Ethereum has performed the best of any blockchain ecosystem in existence today. Despite this success, adoption of Web3 applications still lags well behind Web2. Web3's goal of becoming the de facto global standard for the Internet remains an elusive goal for Ethereum as well as other blockchain communities.

Many Web3 developers have chosen to build decentralized applications due to Ethereum's inherent decentralization:

- Anyone who is online has permission to use the service - or, in other words, no permission is required.
- No one can block you or deny you access to the service.
- Payments are made through its own token, ether (ETH).
- Ethereum is Turing complete, which means you can program just about anything.

Web3 currently has some limitations:

- Scalability - transactions in web3 are slower because they are decentralized. State changes, like payment, must be processed by the miner and propagated across the network.
- UX - Interacting with web3 applications can require additional steps, software, and training. This can be a barrier to adoption.
- Accessibility - The lack of integration into modern web browsers makes web3 less accessible for most users.
- Cost - Most successful decentralized applications put very small chunks of their code on the blockchain because it is expensive.



## Web2 vs. web3

Web2	Web3
Twitter can censor any account or tweet	Web3 tweets would be uncensorable because control is decentralized
Payment service may decide to not allow payments for certain types of work	Web3 payment apps require no personal data and can't prevent payments
Servers for gig-economy apps could go down and affect worker income	Web3 servers can't go down – they use Ethereum, a decentralized network of 1000s of computers as their backend

Figure 13

### 3.12 Solid.js

Solid is a JavaScript framework for building interactive web applications. With Solid, you can use your existing knowledge of HTML and JavaScript to create components that can be reused in your application. Solid provides tools to make your components more reactive: declarative JavaScript code that links the user interface to the data it consumes and creates.

Mix of HTML and JavaScript is called JSX. Solid ships with a compiler that turns these tags into DOM nodes later.

JSX allows you to use most HTML elements in our app, but it also lets you create new elements.

#### 3.12.1 Solid.js vs React.js

One of the most striking differences between Solid and React is the absence of a virtual DOM. The key to its strong performance is that it interacts directly with the DOM (no virtual DOM) and it performs “fine-grained” DOM updates. So, when your state changes, SolidJS updates only the code that depends on it.

In the past, JavaScript libraries made use of the virtual DOM because it was comparatively faster than the DOM itself. JavaScript libraries make changes to the virtual DOM and then compare the changes against the real DOM, and merge both. In the result the Virtual DOM slows down an application.

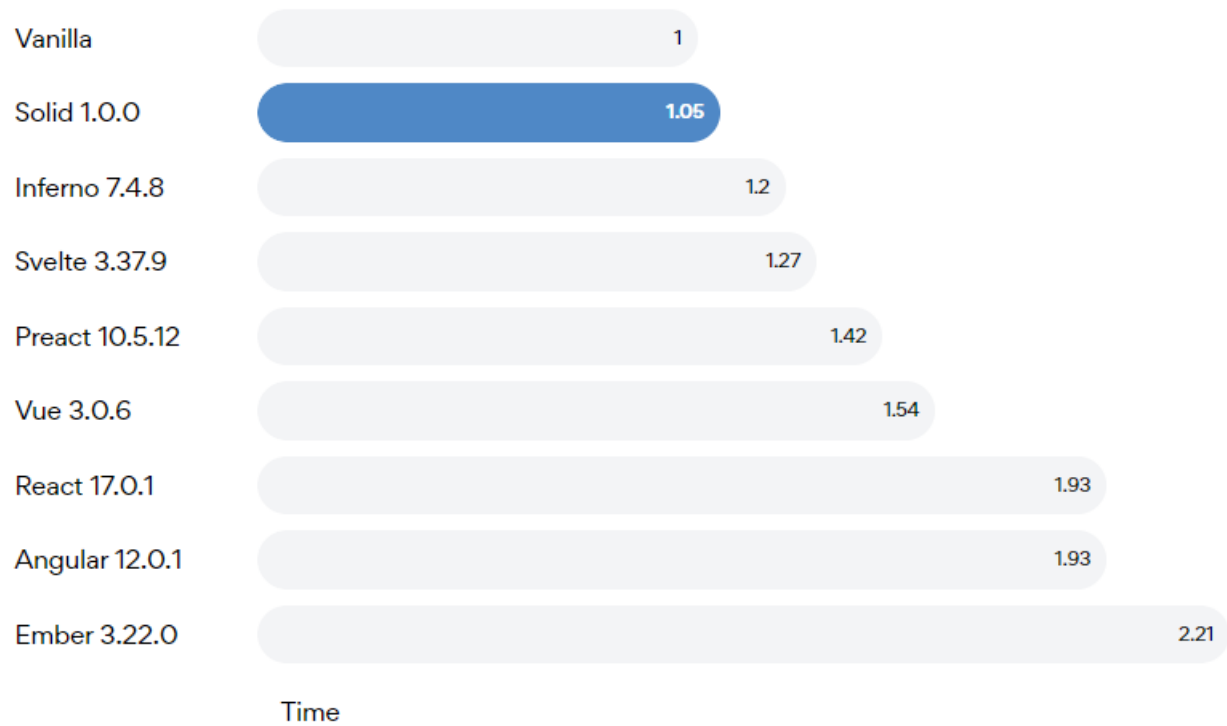


Figure 14

Benchmarks criteria will be performance profiling in Chrome Dev Tools and final bundle sizes. Let's start with Performance profiling. The performance tab shows an overall breakdown of CPU activity into four categories:

1. Loading: Making network requests and parsing HTML
2. Scripting: Parsing, compiling, and running JavaScript code, also includes Garbage Collection (GC)
3. Rendering: Style and layout calculations
4. Painting: Painting, compositing, resizing and decoding images

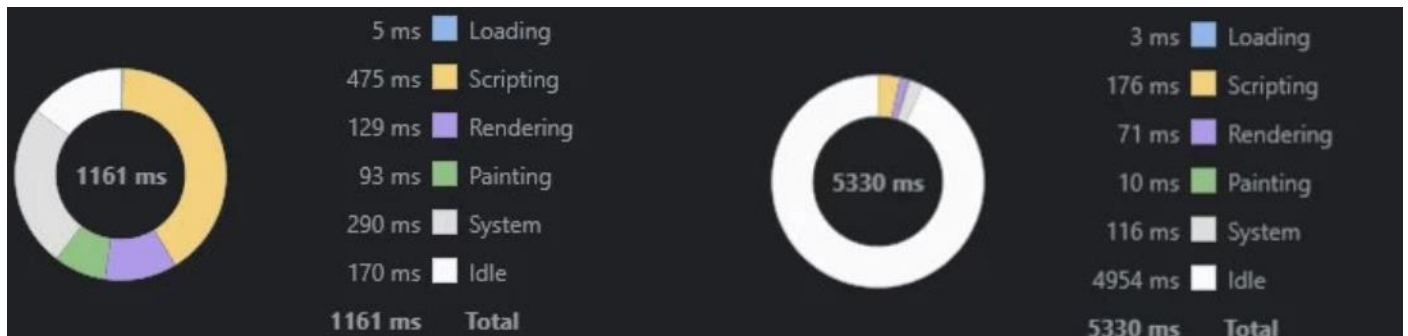


Figure 15: The left side is React, and the right is Solid. As you can see Scripting part is almost 3x faster, Rendering almost 2x faster, Painting part is abnormally faster.

### 3.12.2 Basic Reactivity

Solid's overall approach to reactivity is to wrap any reactive computation in a function and rerun that function when its dependencies update.

The Solid JSX compiler also wraps most JSX expressions (code in braces) with a function, so they automatically update (and trigger corresponding DOM updates) when their dependencies change. More precisely, automatic rerunning of a function happens whenever the function gets called in a tracking scope, such as a JSX expression or API calls that build "computations" (createEffect, createMemo, etc.).

By default, the dependencies of a function get tracked automatically when they're called in a tracking scope, by detecting when the function reads reactive state (e.g., via a Signal getter or Store attribute).

As a result, you generally don't need to worry about dependencies yourselves. (But if automatic dependency tracking ever doesn't produce the results you want, you can override dependency tracking.) This approach makes reactivity composable: calling one function within another function generally causes the calling function to inherit the dependencies of the called function.

### 3.13 Less

#### `{less}`

Less (which stands for Leaner Style Sheets) is a backwards-compatible language extension for CSS. This is the official documentation for Less, the language and Less.js, the JavaScript tool that converts your Less styles to CSS styles.

A CSS preprocessor is a program that lets you generate CSS from the preprocessor's own .unique syntax

There are many CSS preprocessors to choose from, however most CSS preprocessors will add some features that don't exist in pure CSS, such as mixin, nesting selector, inheritance selector, and so on. These features make the CSS structure more readable and easier to maintain.

### 3.14 TypeScript

TypeScript is JavaScript with syntax for types. TypeScript is a strongly typed programming language that builds on JavaScript, giving you better tooling at any scale.

TypeScript code converts to JavaScript, which runs anywhere JavaScript runs: In a browser, on Node.js or Deno and in your apps.

TypeScript understands JavaScript and uses type inference to give you great tooling without additional code.

## 4 Expected Achievements

Our product, Alpha, aims to provide a fast, stable, simple and responsive user experience for basic and advanced users alike.

### 4.1 Product Infrastructure and Flow

We plan on using the Google Chrome extension API, backed by the SolidJS frontend framework, which we will develop locally using Node.js.

Launching the extensions the user will be prompted by either a login process or the content of the extension which will include: a convenient dashboard with relative user information, frontend services based on DeFi APIs, and a settings panel in which the user will be able to modify his experience using the extension.

### 4.2 Success Criteria

- I. Implementation of multiple initial decentralized finance services (*Uniswap, Compound*)
- II. Self-explanatory user interface
- III. Smooth and responsive user experience
- IV. Positive (>75%) user feedback

### 4.3 Unique Features

- I. User experience above all
- II. Module-based and extensible – advanced users can implement their own services given an API
- III. Non-profit, open-source extension
- IV. No internal fees

## 5 Engineering Process

This section details the challenges we faced and expected to encounter, as well as techniques we used during the product engineering process.

### 5.1 Process

The research and development process began by inquiring about the Blockchain field in general, the Ethereum Blockchain, cryptocurrency diversity, DeFi services, and more. This is a new and evolving type of technology which contains many new concepts that required us to study many related ideas and approaches, out of which many are new to us.

Alongside formulating the idea and the use-cases of our project we had also started developing a prototype which helped us to come up with new ideas and discard less successful ones.

During the said process, we also had to learn about web development, mainly Node.js, Web3.js and Solid.js, in order to build the prototype. We also got to learn about basic financial concepts as well, which helped to shape our project.

#### 5.1.1 Methodology

As for our process methodology we chose to implement fundamental parts of our project as a simple prototype extension, in order to quickly iterate over ideas and to obtain hands-on experience with the technologies we had learned about.

#### 5.1.2 Research

To maintain chronology for all the studies that have been carried out, you can see in the graphs below, in addition, the technical aspects and technologies are listed as following:

- The Blockchain network and the Ethereum realm
- MetaMask wallet
- Hierarchical Deterministic (HD) wallets
- Web3.js
- Node.js
- Cryptography and Cryptographic Signatures
- Solid.js
- Less
- TypeScript
- Infura
- Compound
- Uniswap
- DYDX (optional)

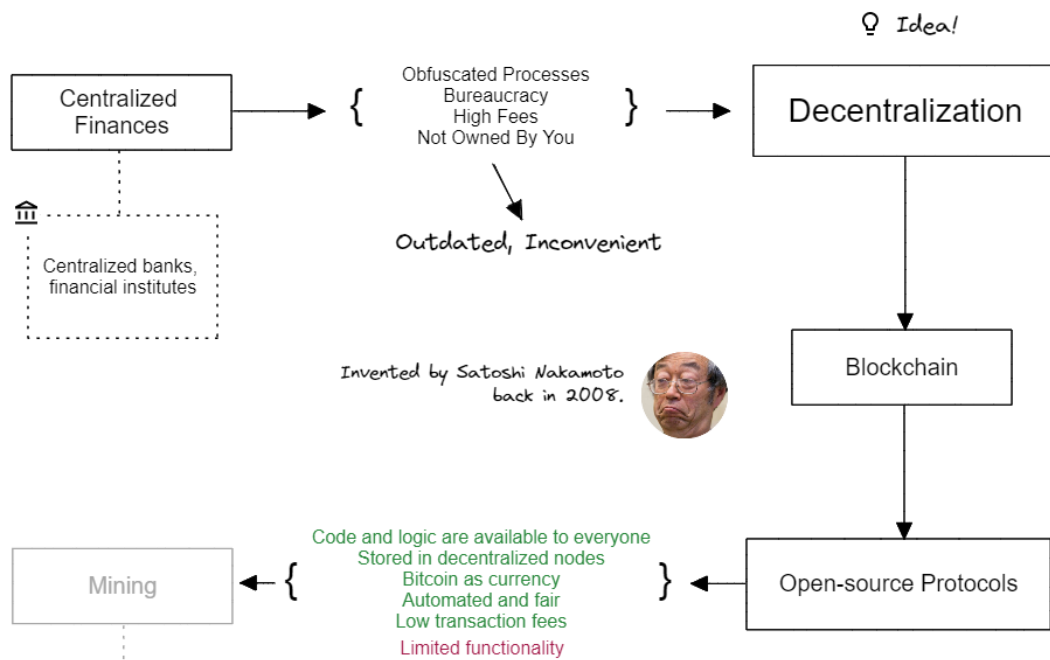


Figure 16: From centralized finances to creating new blocks of information on the Blockchain.

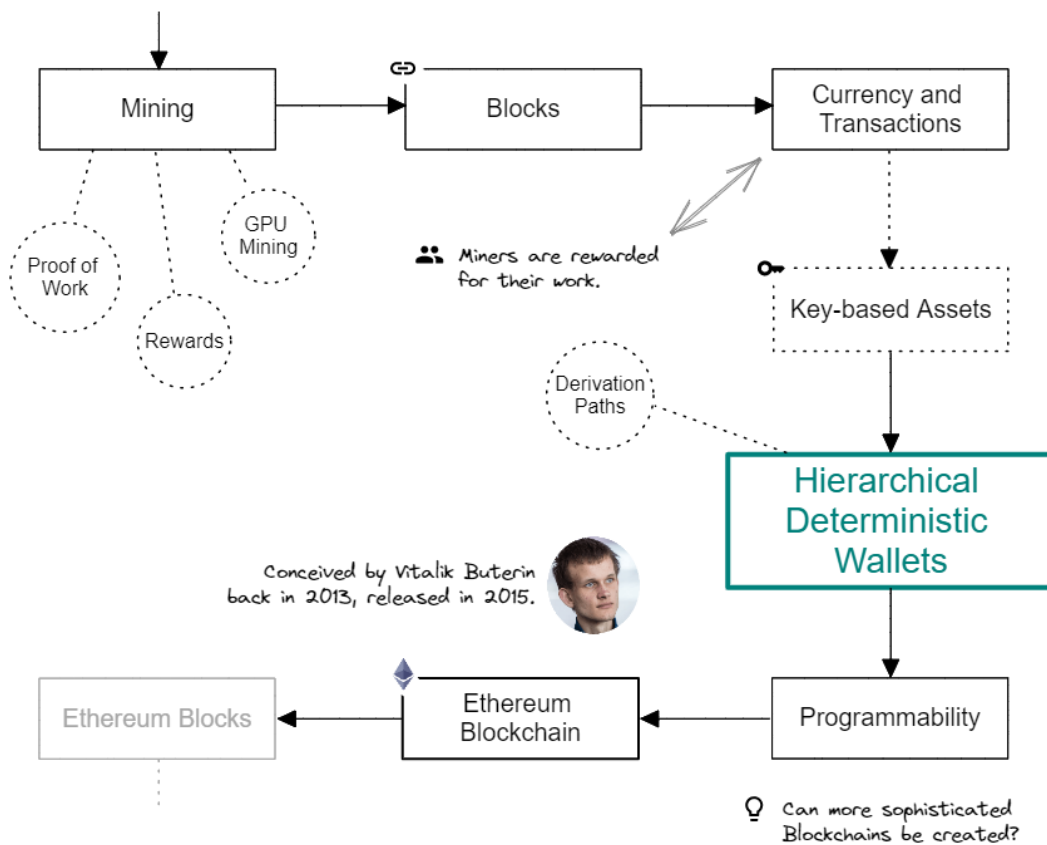


Figure 17: From transactions using cryptographic keys to developing DeFi services on the Ethereum Blockchain.

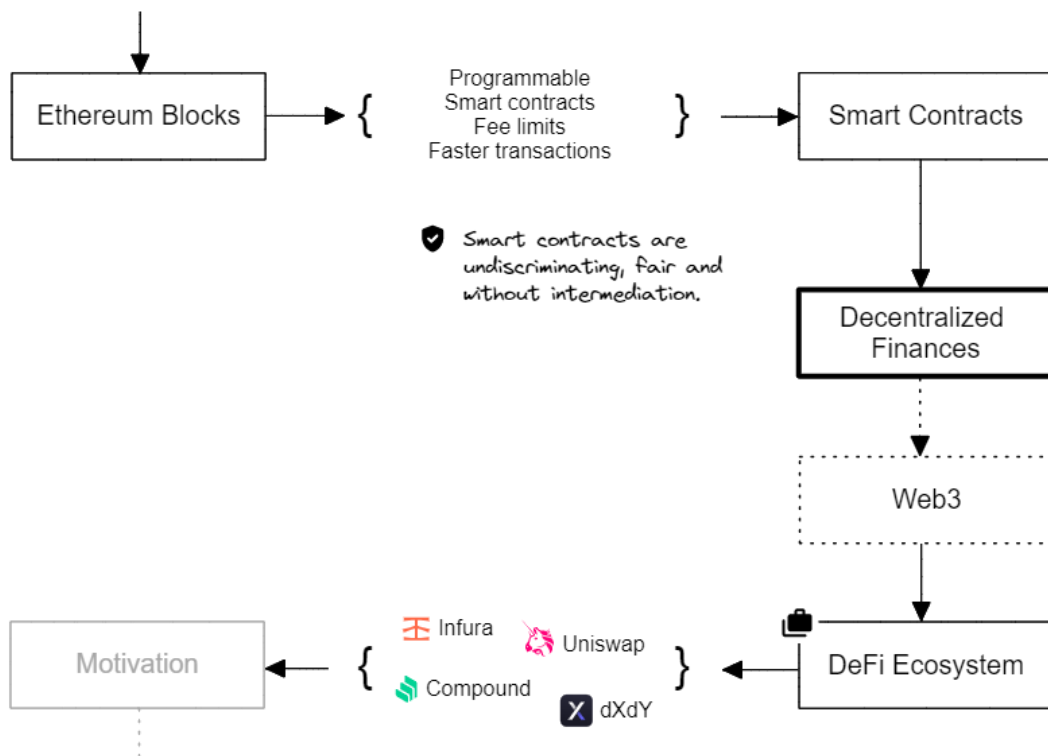


Figure 18: From the benefits of programmable blocks to the motivation behind our project.

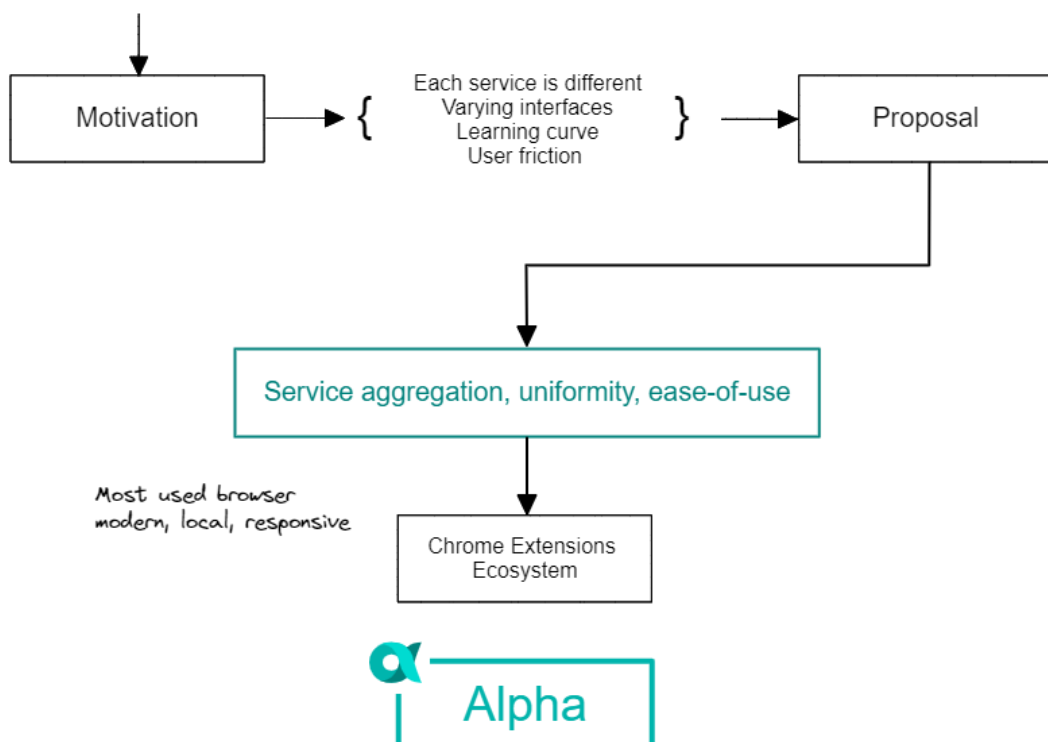


Figure 19: The motivation and framework of our project.

### 5.1.3 Challenges

As we started researching the Blockchain realm, we quickly realized we entered a very alien and vast world unbeknown to us that which have to delve deep to understand the underlying concepts, which usually implement traditional concepts in the virtual Blockchain technology. Alongside the Blockchain technology we also had to learn about fundamental web technologies, the difference between the Web2 and Web3 approaches to the modern internet, the Blockchain-based decentralized financial services (namely Infura, Compound and Uniswap).

We also expect to encounter various hurdles during Phase 2 of the project, where we might run into different implementation problems such as using Solid.js, different outdated API documentations, and testing our product using the Ethereum testing networks.

**We faced number of challenges during development:**

#### 1. Application Architecture


- We had no backend servers to work with, so we had to rely on storing and managing local files.
- We changed our frontend framework mid-development (React to Solid) to manage application states more easily.

#### 2. Blockchain

- DeFi services often had API documentation inconsistencies making it harder to implement
- The Blockchain protocol is not HTTP-friendly and has to use intermediate libraries to interact with

## 5.2 Product

Originally, our product is a browser extension. However, by the nature of Chrome-based extensions, it can function as a local or hosted standalone website where users can create or import a cryptocurrency wallet.

Once a user creates or import a cryptocurrency wallet into  Alpha, the user can take actions as if they own a small private bank of their own.

Among the aforementioned actions we can expect operations such as: checking account balance, perform transactions between different wallets, borrow and lend money and coin and token conversions.

All the actions available to the users will use external APIs of various services such as Infura, Compound and Uniswap, albeit can be extended by advanced users as mentioned before.

Providing these APIs to the user means that our extension implements these API endpoints in our extension as frontend graphical user interfaces using HTML, CSS, JavaScript and high-level frameworks such as Solid.js and TypeScript.

It is also important to note that using Blockchain technologies and external APIs we do not need a backend server and the extension client can run completely independently from traditional web services that rely on the typical HTTP request-response paradigm.



### 5.2.1 Requirements

#### **Functional:**

- The system is based on the Ethereum blockchain.
- The system supports cryptocurrencies.
- The system supports the Infura API.
- The system supports the Compound API.
- The system supports the Uniswap API.
- The system can create new wallet.
- The system can import exist wallet (by a 12-words seed phrase or a private key).
- The system can transfer money to different addresses.
- The system supports borrowing and lending money.
- The system supports making conversation between different tokens
- The system supports ERC-20 token conversion.
- The system supports adjusting various extension settings.
- The system is based on an internet connection.

#### **Non-functional:**

- A secure system for using cryptocurrencies.
- User friendly interface.
- User friendly experience.
- The system is always readily available.
- The system can be used anywhere with Internet access using Google Chrome.
- All services are condensed and utilized in a single extension.
- The system optimized the interest.
- The system can be used 24/7.

## 5.2.2 Architecture

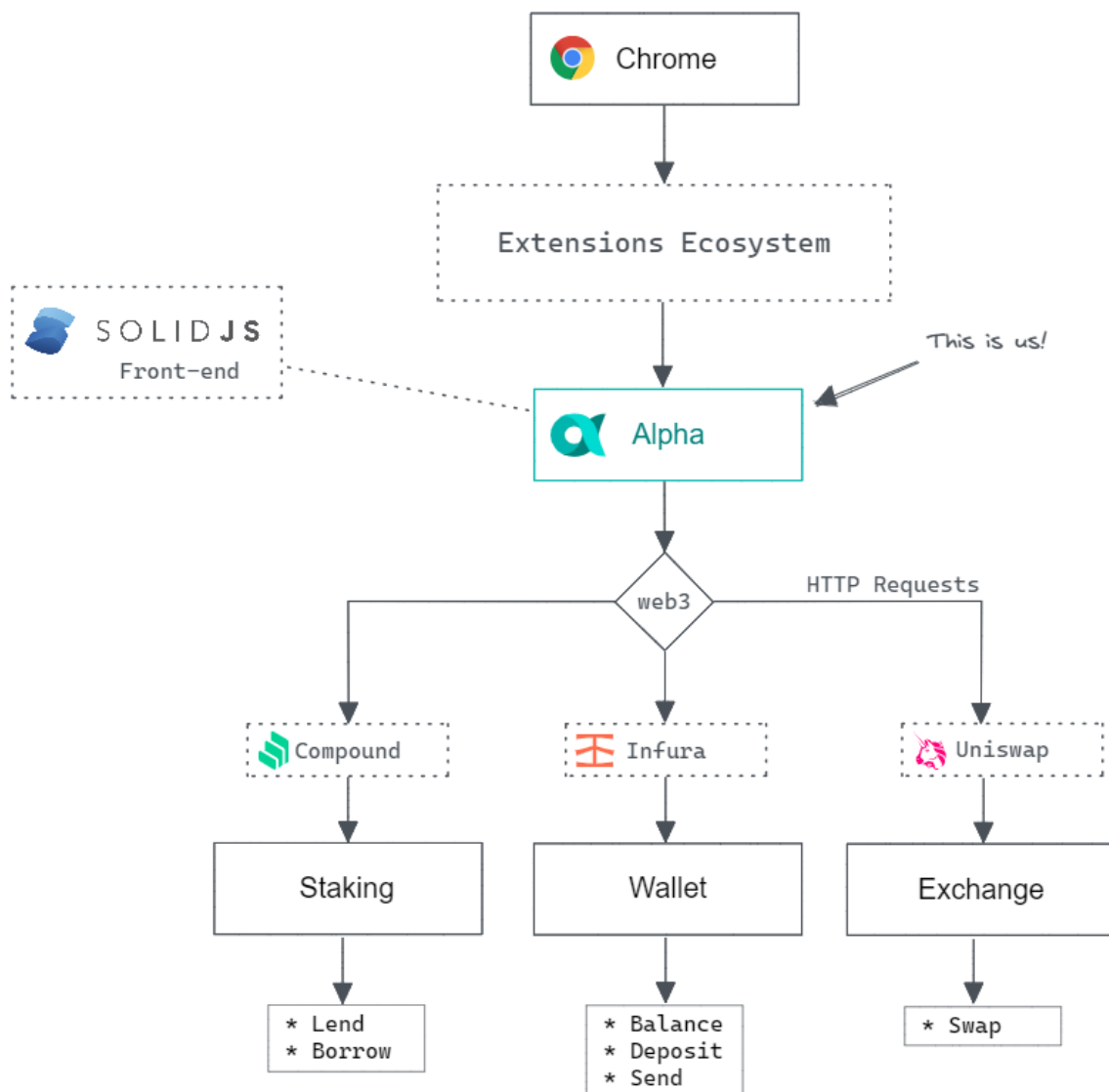


Figure 20: An overview of the previously mentioned architecture as a flowchart.

## 5.2.3 API

A prominent feature of our extension would be extensibility. Advanced users and programmers should be able to easily modify and add new service modules to the extension using methods and classes we would provide, which we will also be using to implement the basic services (Infura, Compound, Uniswap).

Adding a module might be as simple as modifying a JSON list of objects, specifying metadata and available operations:

```

{
  "header": "Exchange",
  "items": [{
    "name": "Uniswap",
    "internal": "uniswap",
    "img": "../img/logos/uniswap.svg"
  }]
},
{
  "header": "Staking",
  "items": [{
    "name": "Compound",
    "internal": "compound",
    "img": "../img/logos/compound.svg"
  }]
}
}

```

Followed by implementing a class which inherits from a base Module class and a Solid.js component with its inner logic, hooking the module to the extension in an elegant manner:

```

class Compound extends Module {
  constructor() {
    // Initializes Compound specific parameters such as an API key
  }
}

```

```

const App = () => {
  Modules.add(new Infura());
  Modules.add(new Compound());
  Modules.add(new Uniswap());
  return <div></div>;
}

```

Resulting in the interface updating accordingly:

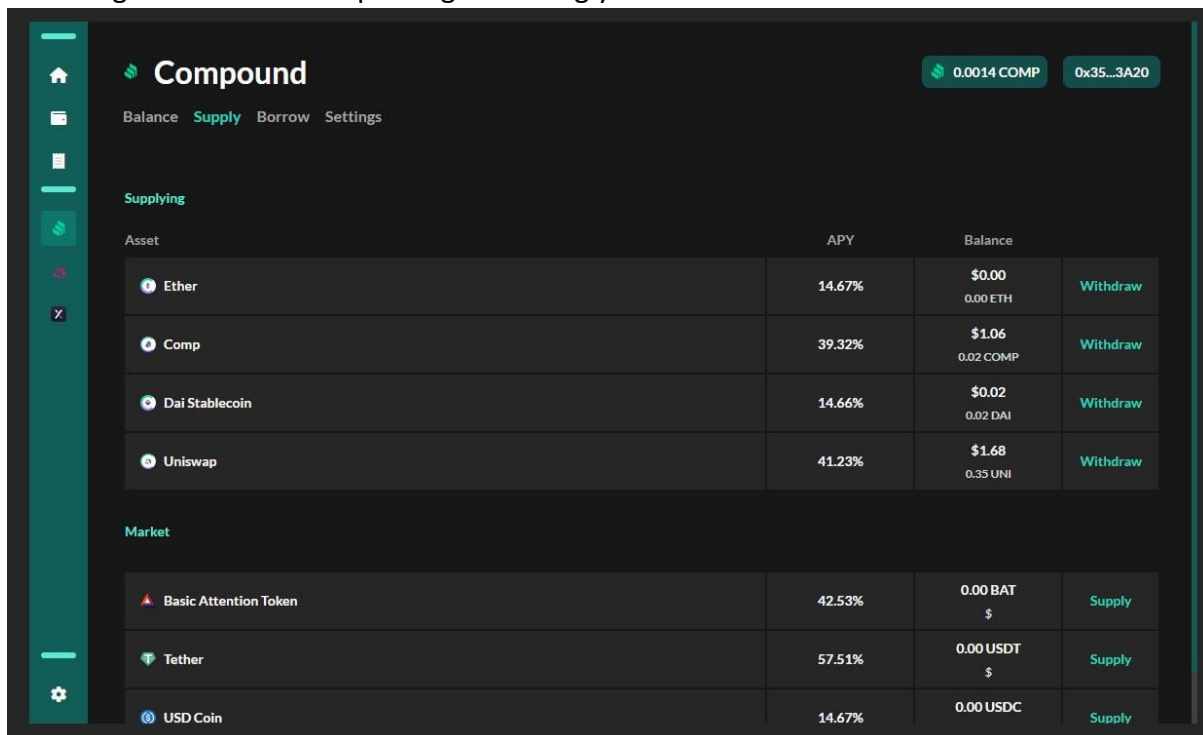


Figure 21: Alpha prototype, showcasing the dynamic menu.

## 5.2.4 Flowcharts and Interface

This section elaborates on user scenarios one might encounter using. Full flow of windows in the "Alpha" application and the included functionality.

### 1. Post-Installation

Users can either **create** new or **import existing** wallets

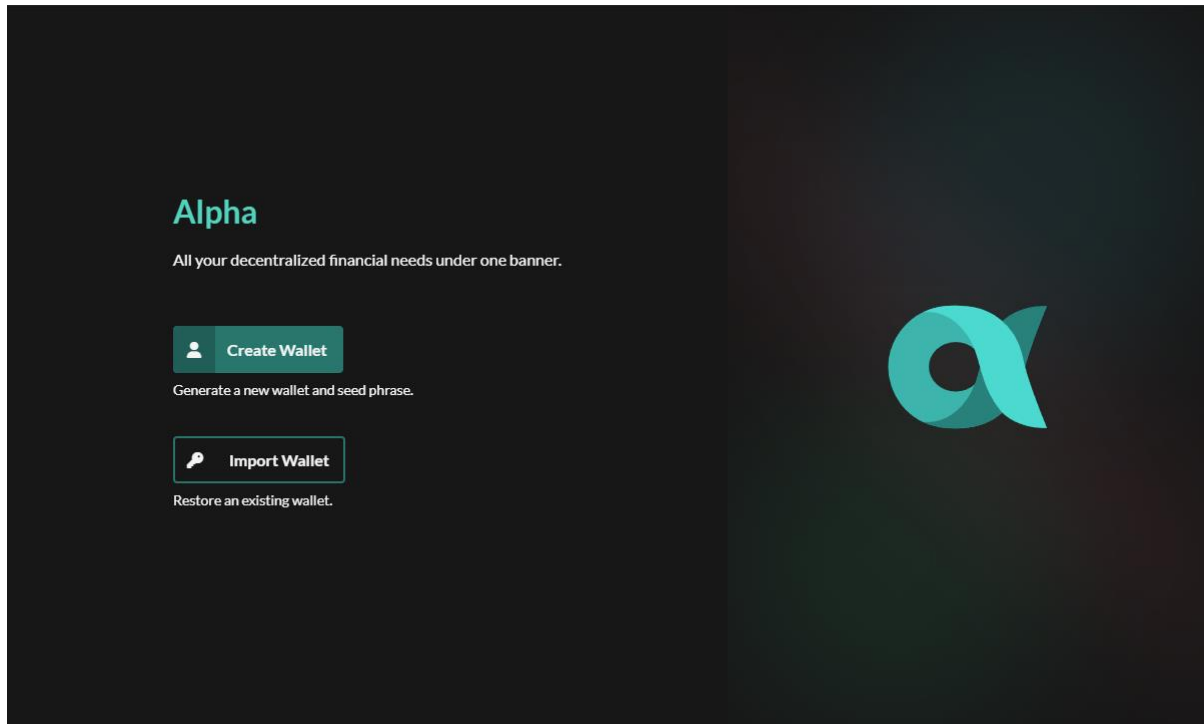
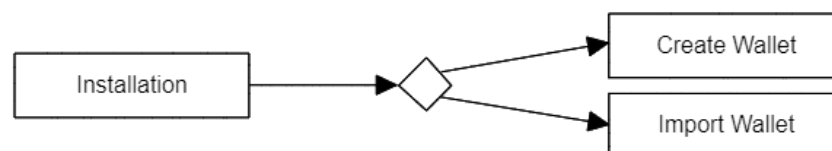


Figure 22: The main menu presented to a new user.



## 2. New Wallet Creation

Here shows creation of a new wallet, as well as some validation feedback.

We elected to stop users from making errors by preventing them from proceeding, a common UX design pattern.

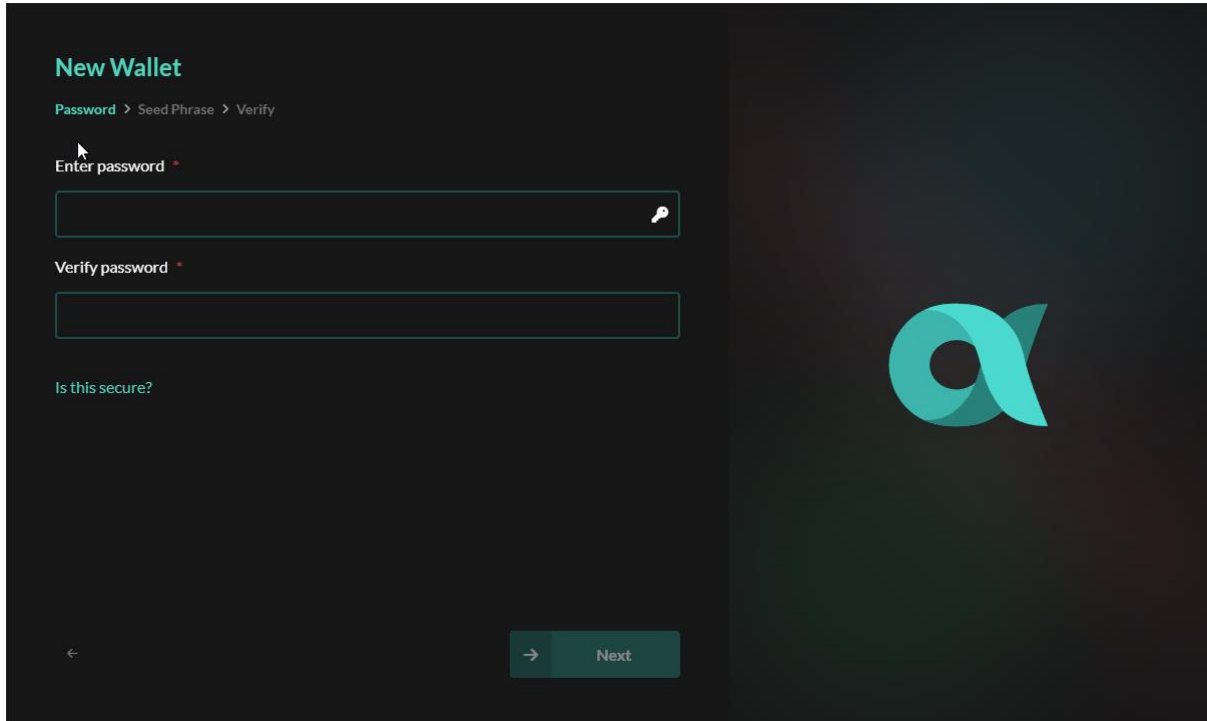


Figure 23: The user is prompted to choose a password of his liking.

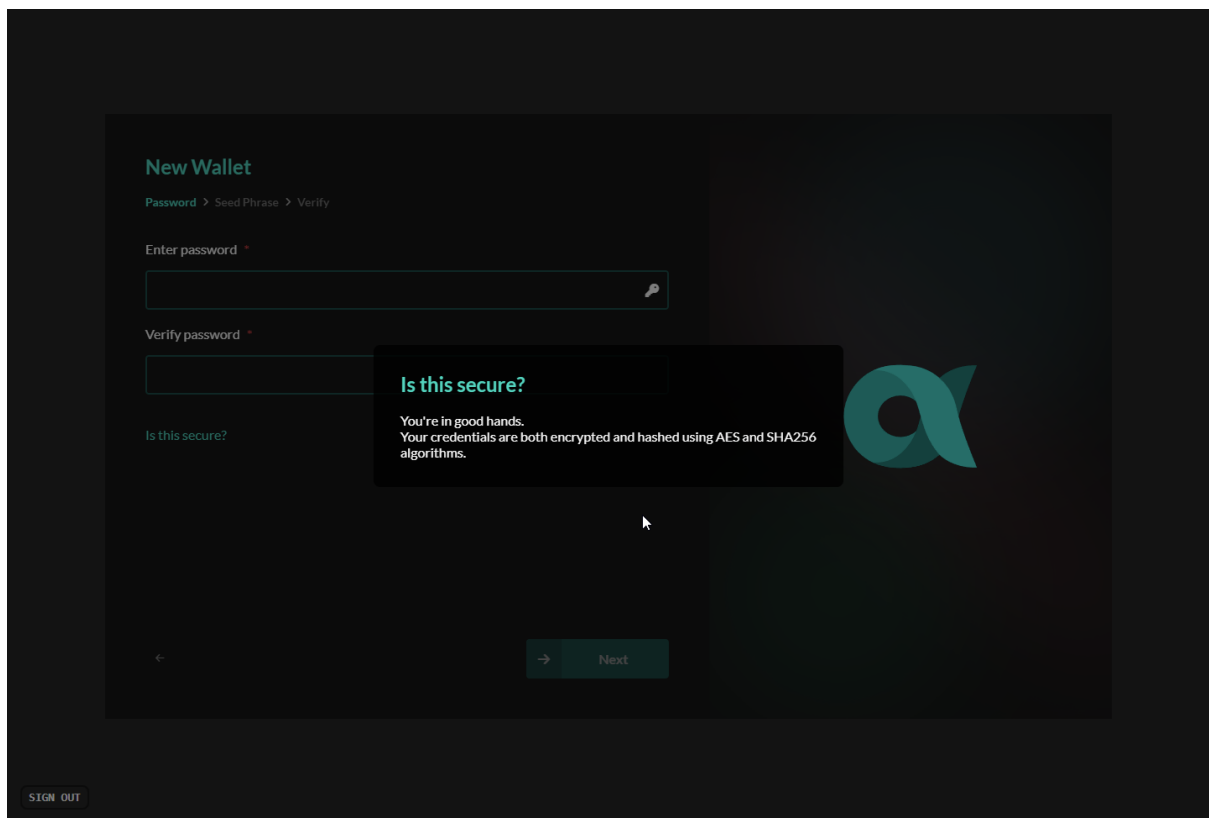


Figure 24: Encryption information for user

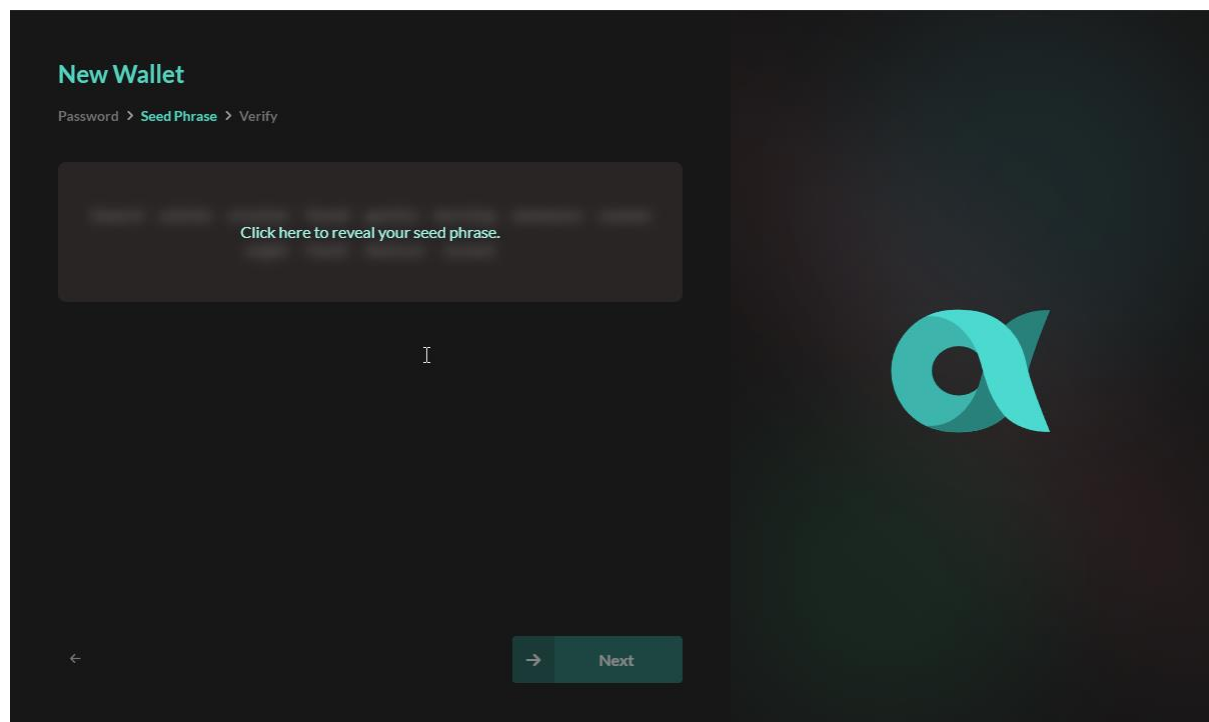


Figure 25: Words of seed phrase are blurry, when the user is ready to see them, he clicks and remembers them well

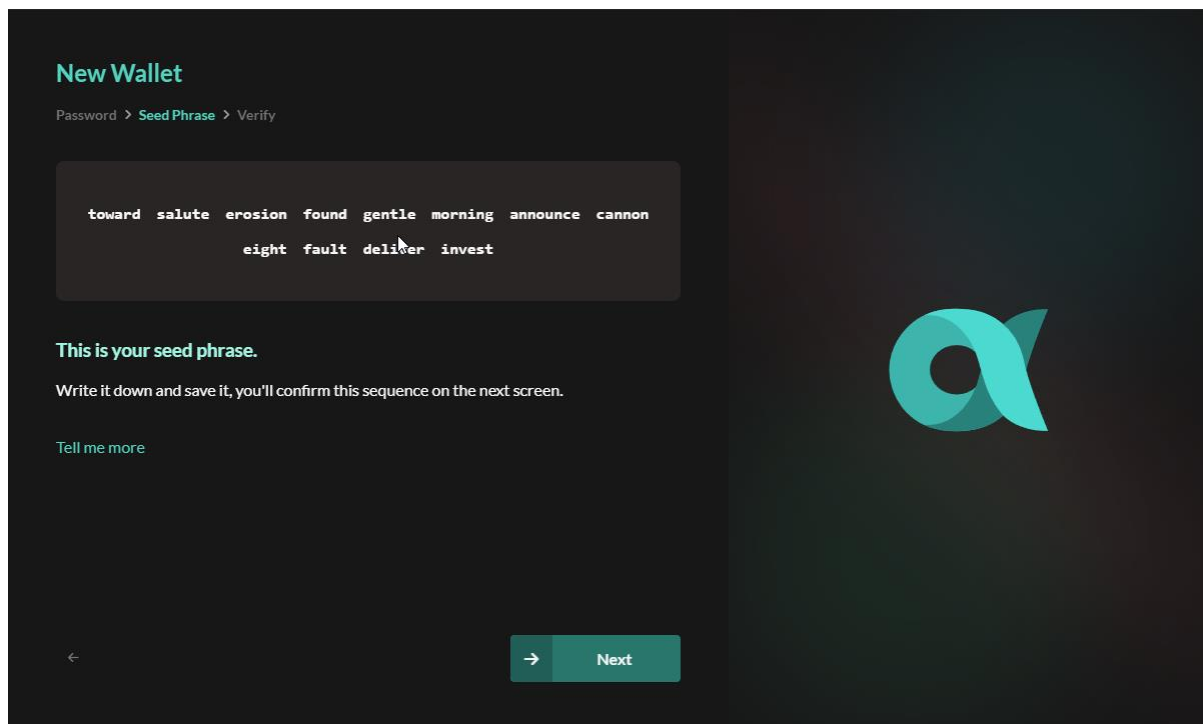


Figure 26: The user is presented with an automatically generated seed phrase.

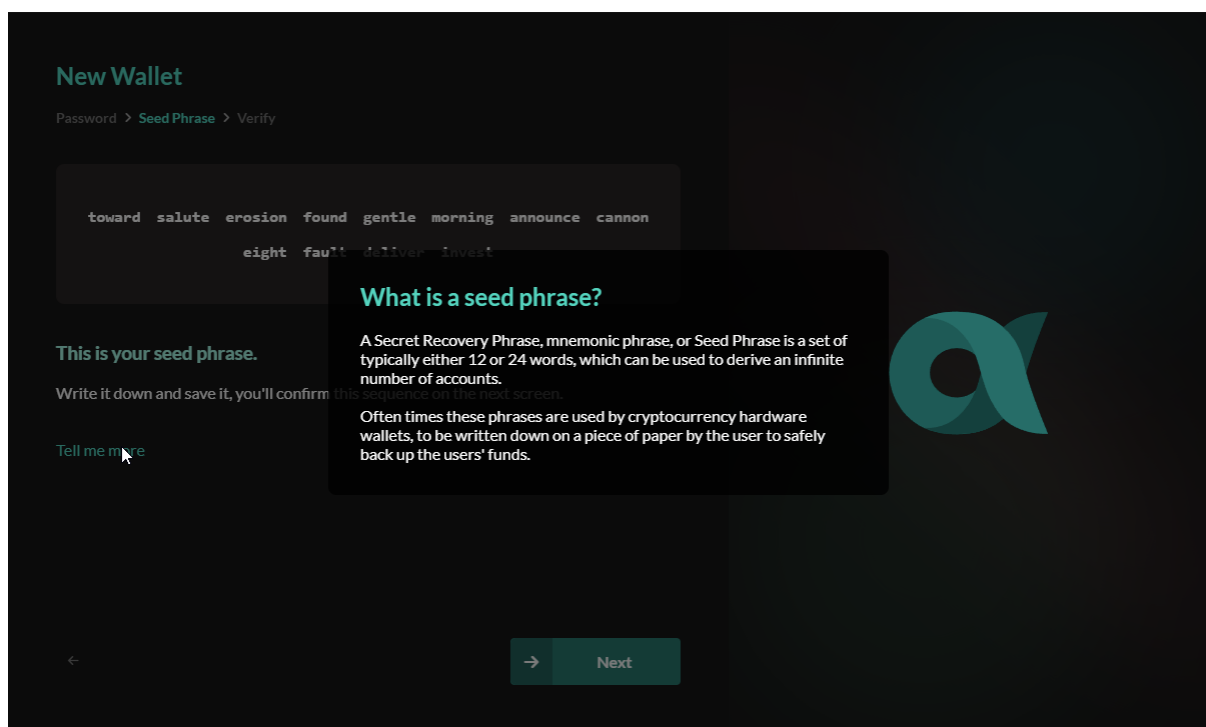


Figure 27: Described what is seed phrase and its importance

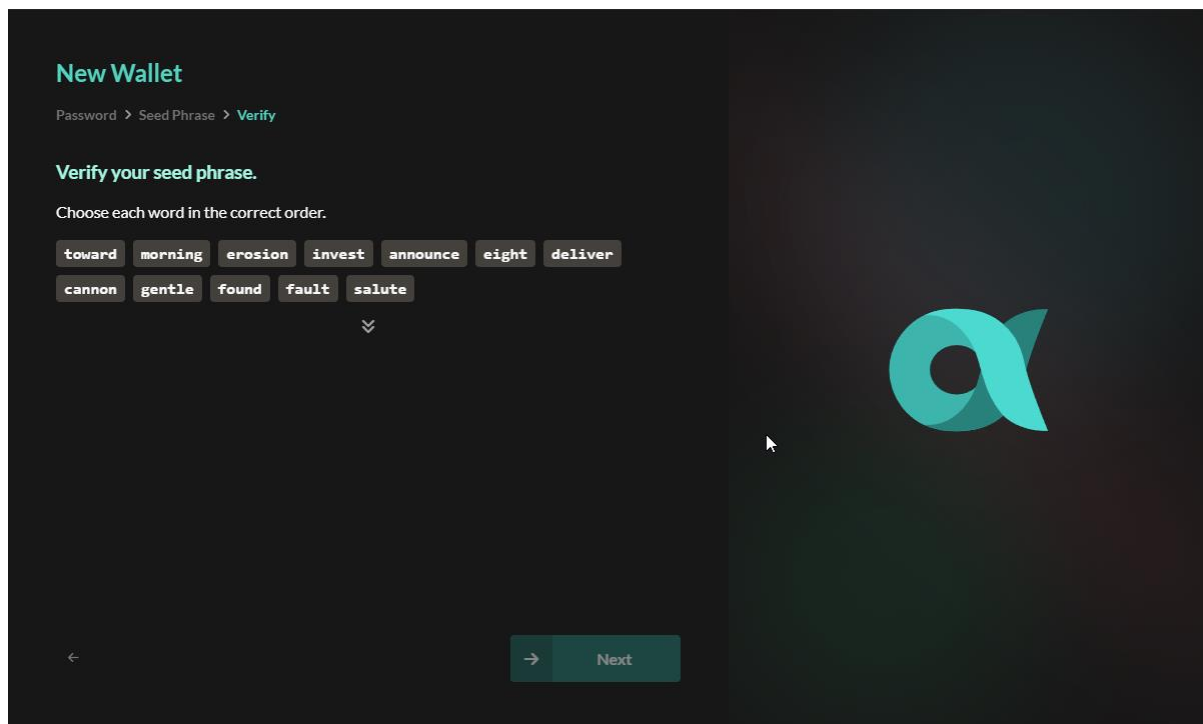


Figure 28: The user should select the words in the correct order to be sure that the user remembers the source phrase and order

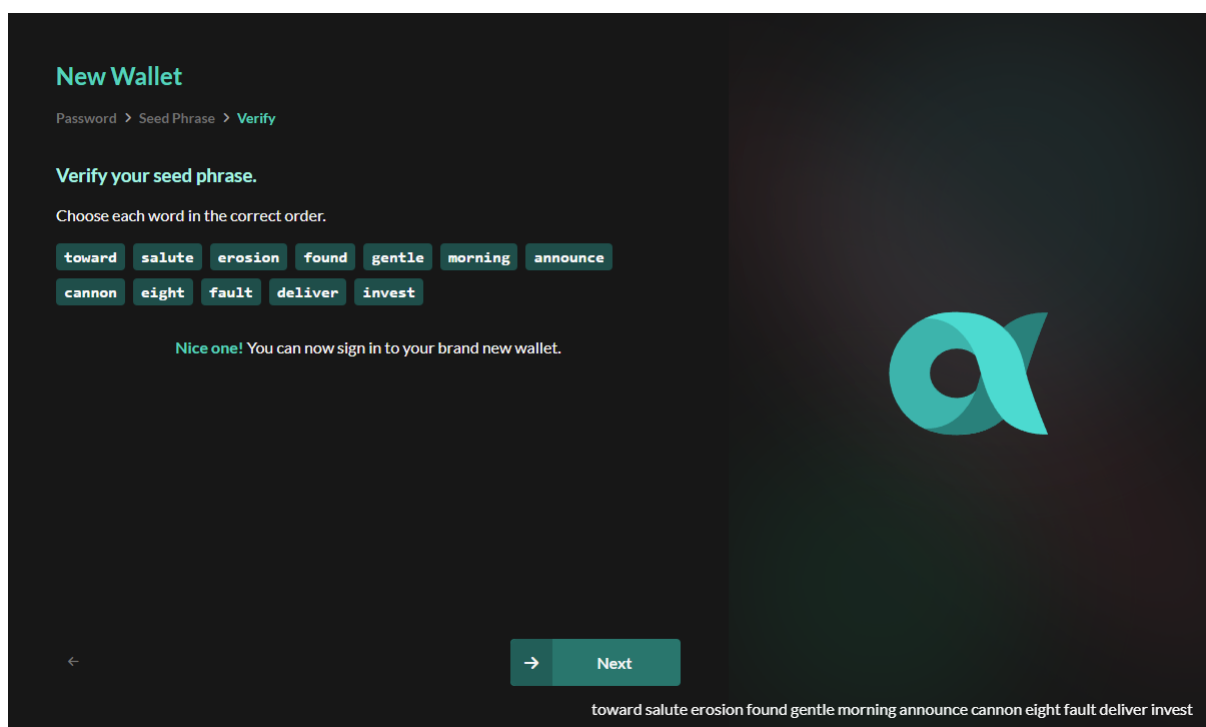


Figure 29: In case if the user is selected it correctly



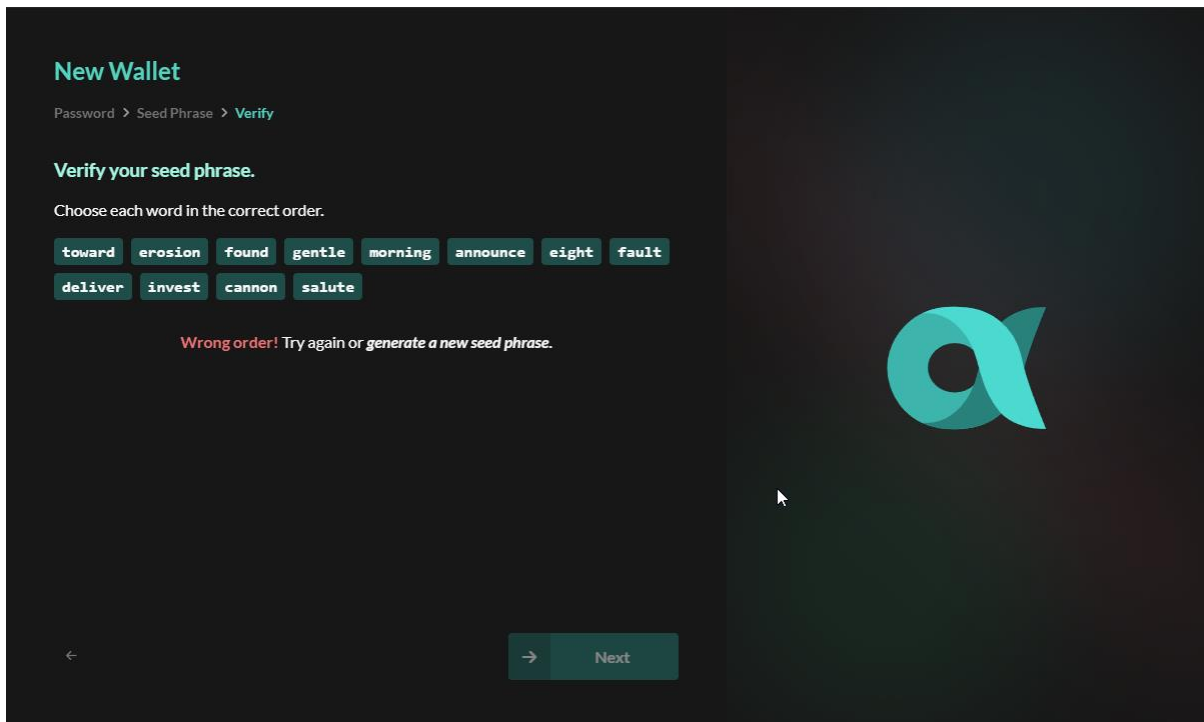


Figure 30: In case the user selects it incorrectly. The user has the option to return the words back and correct the order

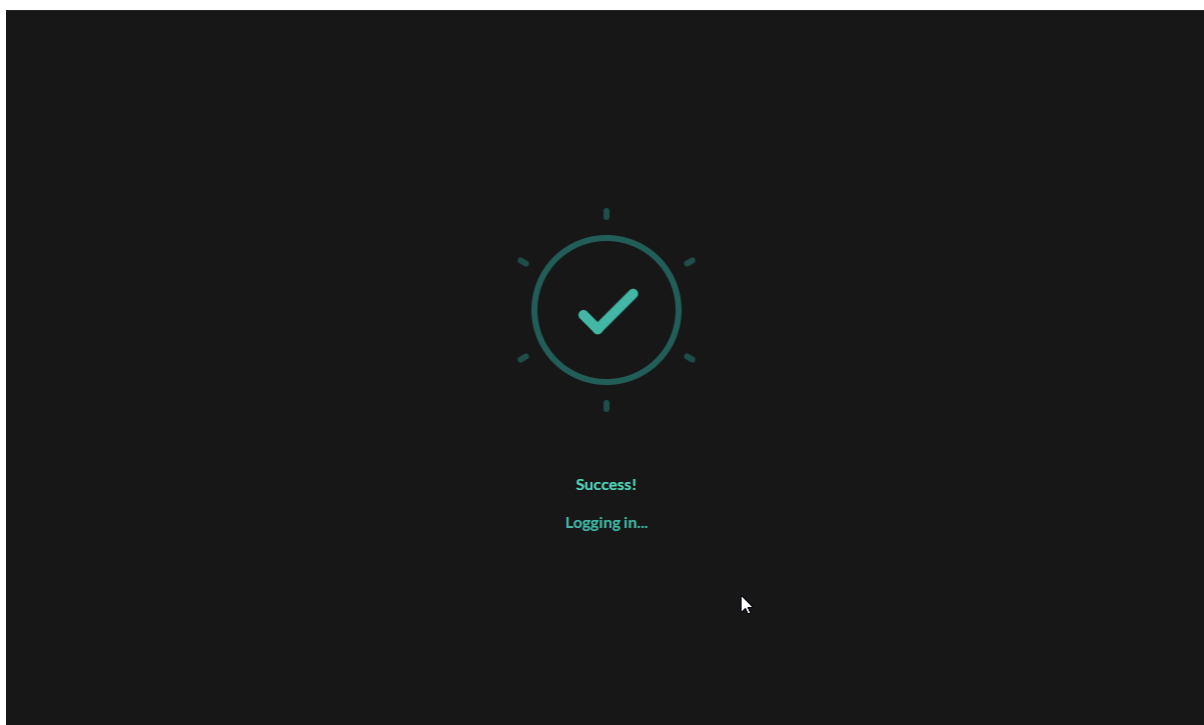


Figure 31: The feedback for users that everything goes successfully

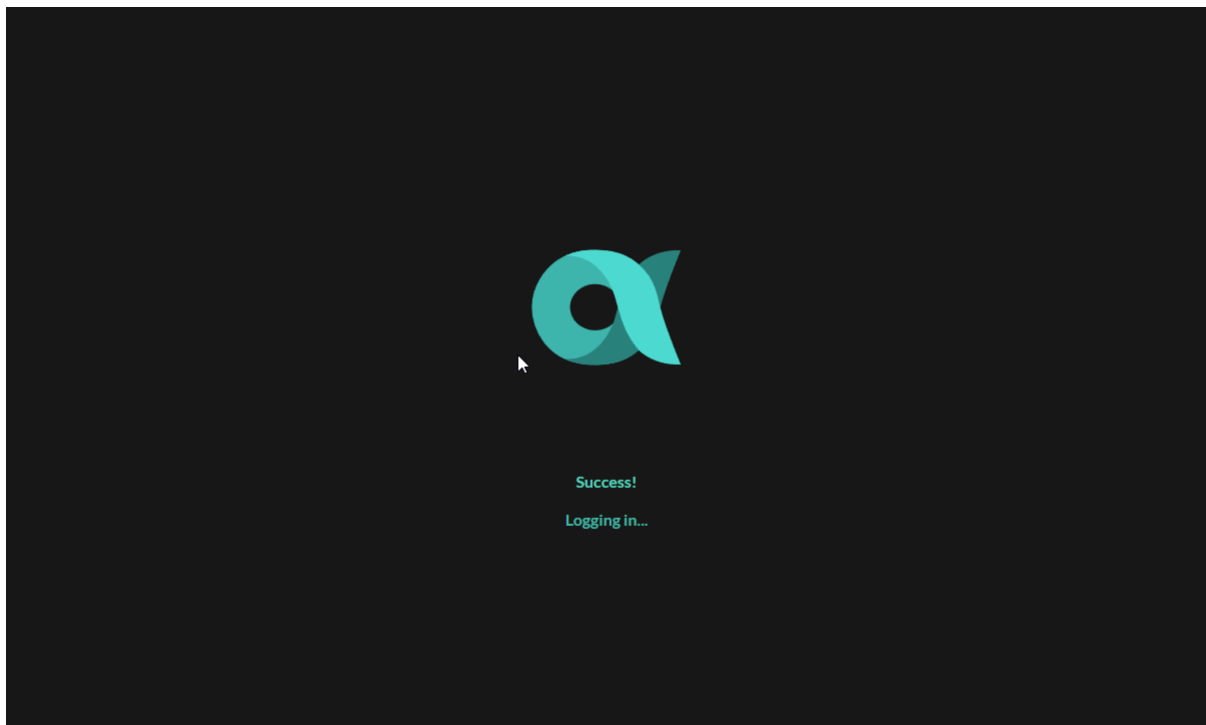
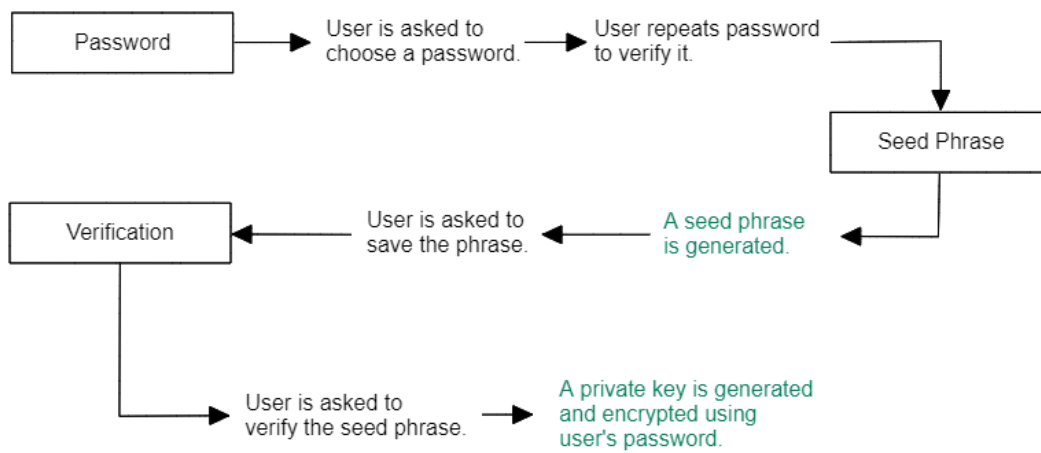
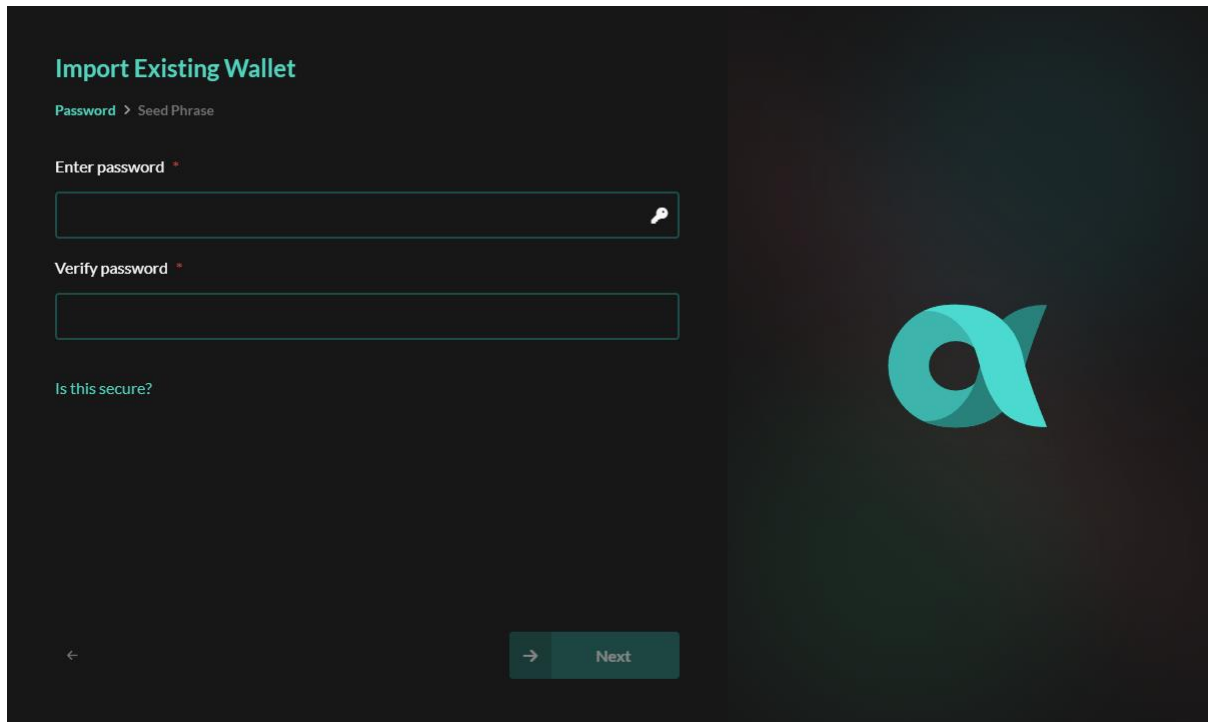


Figure 32: Feedback to users that the app will **log in** in a few moments



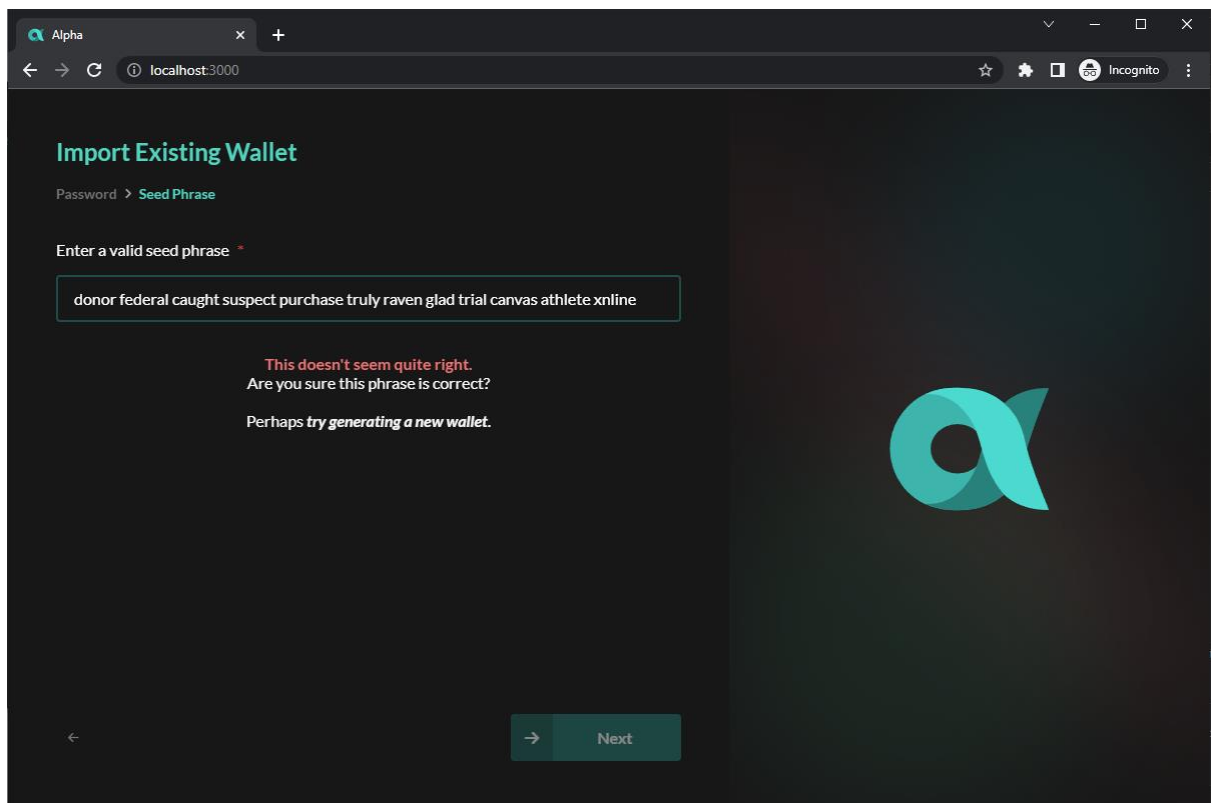
### 3. Wallet Import

Here shows the procedure of importing an existing wallet, as well as some validation feedback.



The screenshot shows the 'Import Existing Wallet' interface. At the top, the title 'Import Existing Wallet' is displayed in teal. Below it, a breadcrumb trail shows 'Password' > 'Seed Phrase'. There are two input fields: 'Enter password' with a red asterisk and a password icon, and 'Verify password' with a red asterisk. Below these is a checkbox labeled 'Is this secure?'. At the bottom, there is a back arrow and a 'Next' button with a right arrow. On the right side of the screen is a large teal Alpha logo.

Figure 33: User can import the wallet that already exist by entering password first



The screenshot shows the 'Import Existing Wallet' interface with an error message. The breadcrumb trail is 'Password' > 'Seed Phrase'. The input field is labeled 'Enter a valid seed phrase' with a red asterisk. The text 'donor federal caught suspect purchase truly raven glad trial canvas athlete.xnline' is entered. Below the input field, a red error message states: 'This doesn't seem quite right. Are you sure this phrase is correct? Perhaps try generating a new wallet.' At the bottom, there is a back arrow and a 'Next' button with a right arrow. On the right side of the screen is a large teal Alpha logo.

Figure 34: In case if the user is fill-in, incorrectly seed phrase

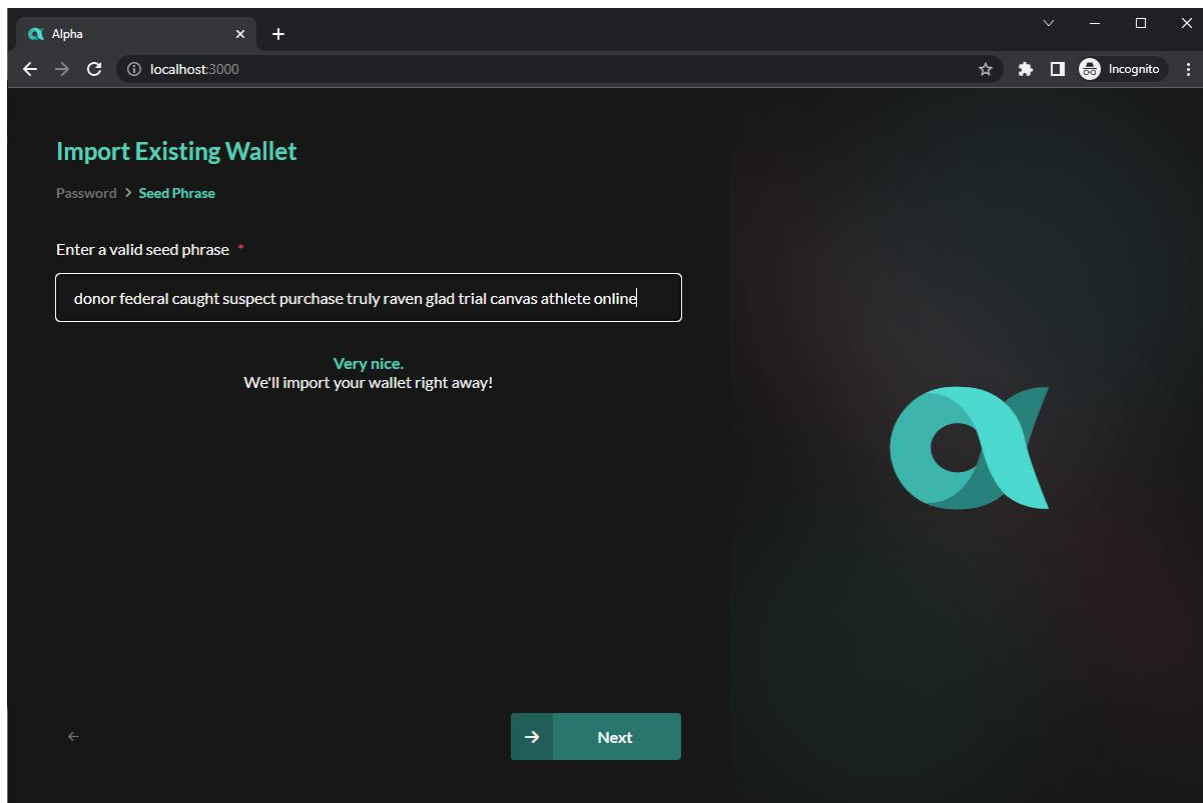
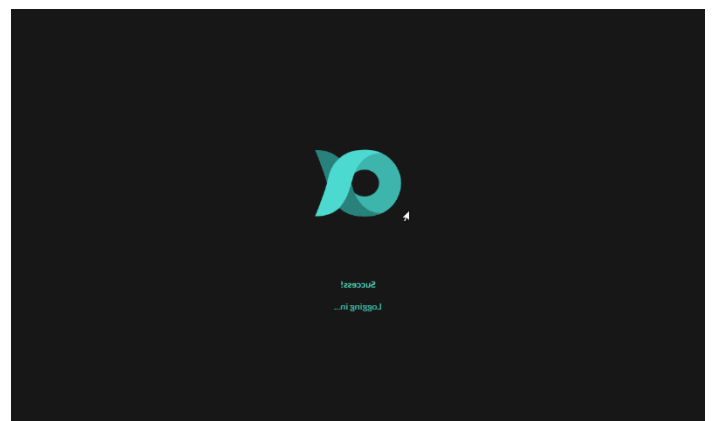
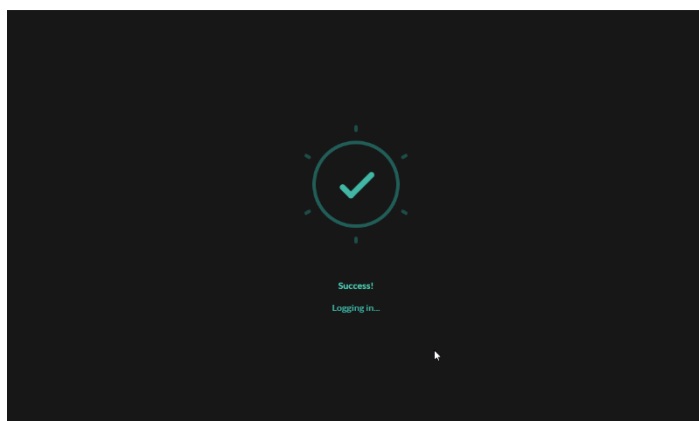
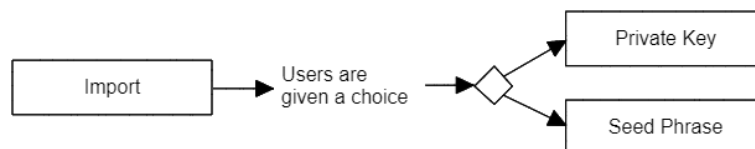


Figure 35: In case if the user is fill-in, correctly seed phrase



#### 4. Login to existing wallet

*This example shows the procedure of signing into an existing wallet, and validation feedback.*

*In case a user forgot his credentials he can choose to create a new wallet or re-import his existing wallet.*

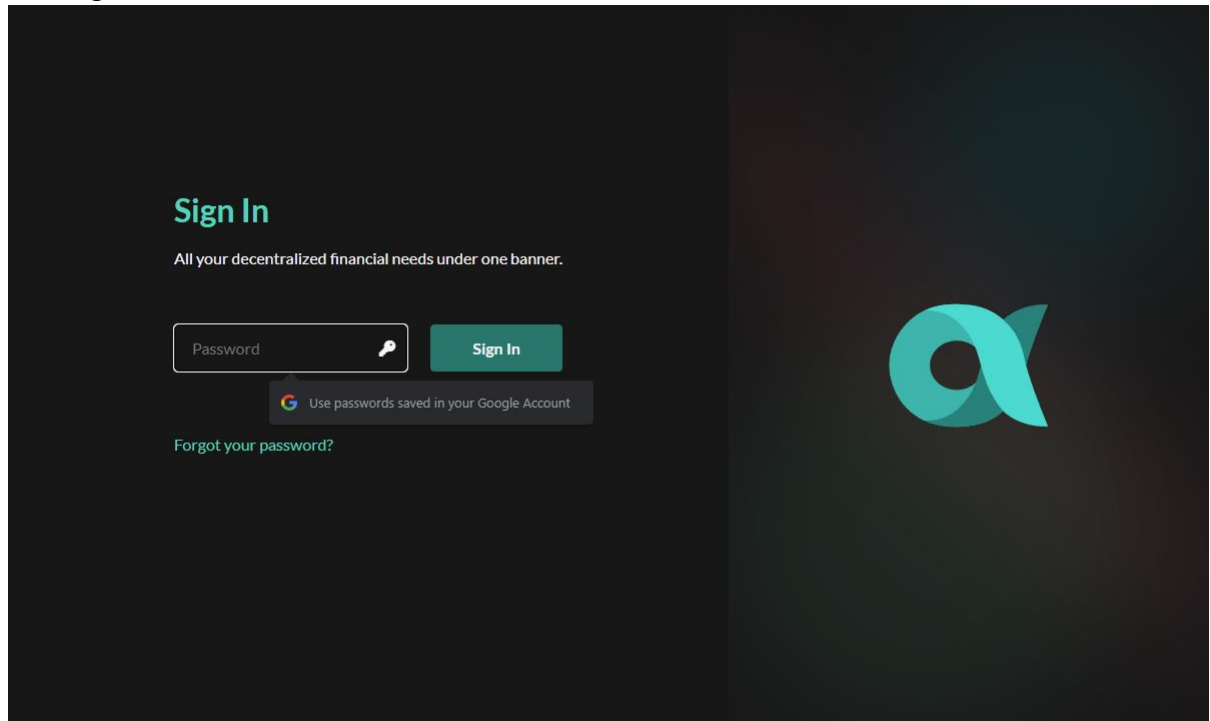


Figure 36: Sign in with user's password to existing wallet

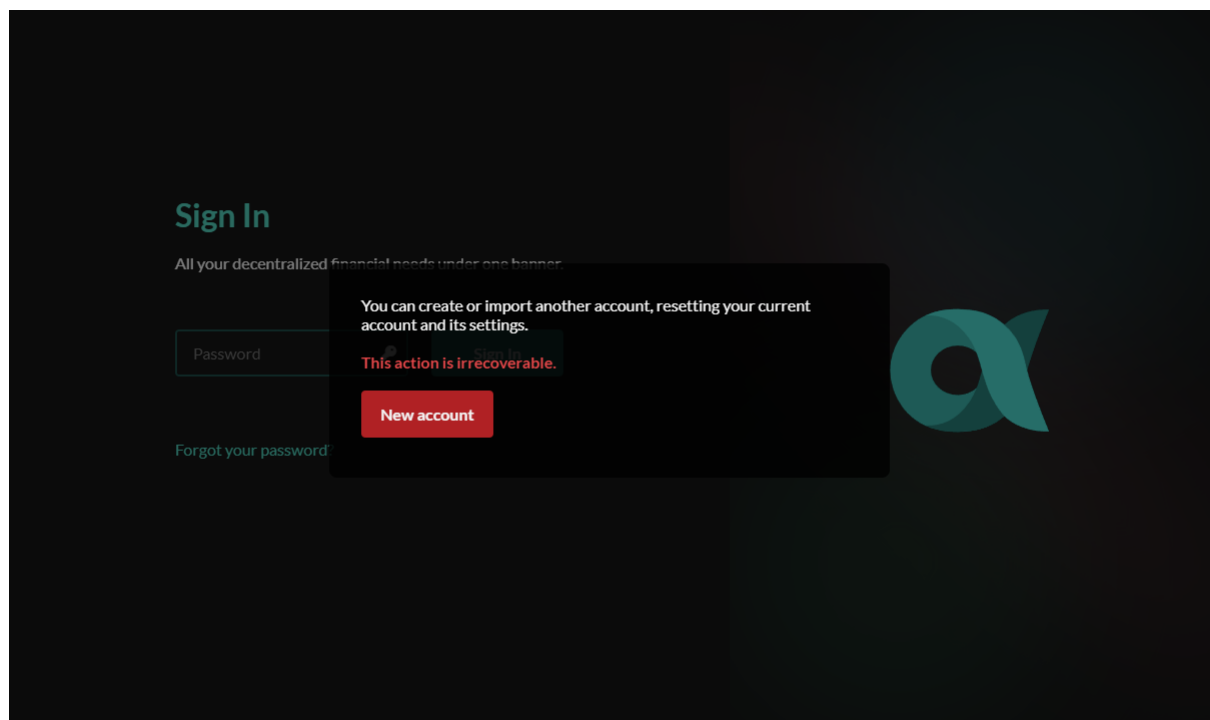


Figure 37: In case a user forgot the password he can choose to create a new wallet or re-import his existing wallet. In case the user come into he got feedback that inference succeed.

5. Home (Dashboard) Interaction into Alpha after successful sign-in.

Users will interact with the menu to access a user dashboard, different modules, services, and extension settings.

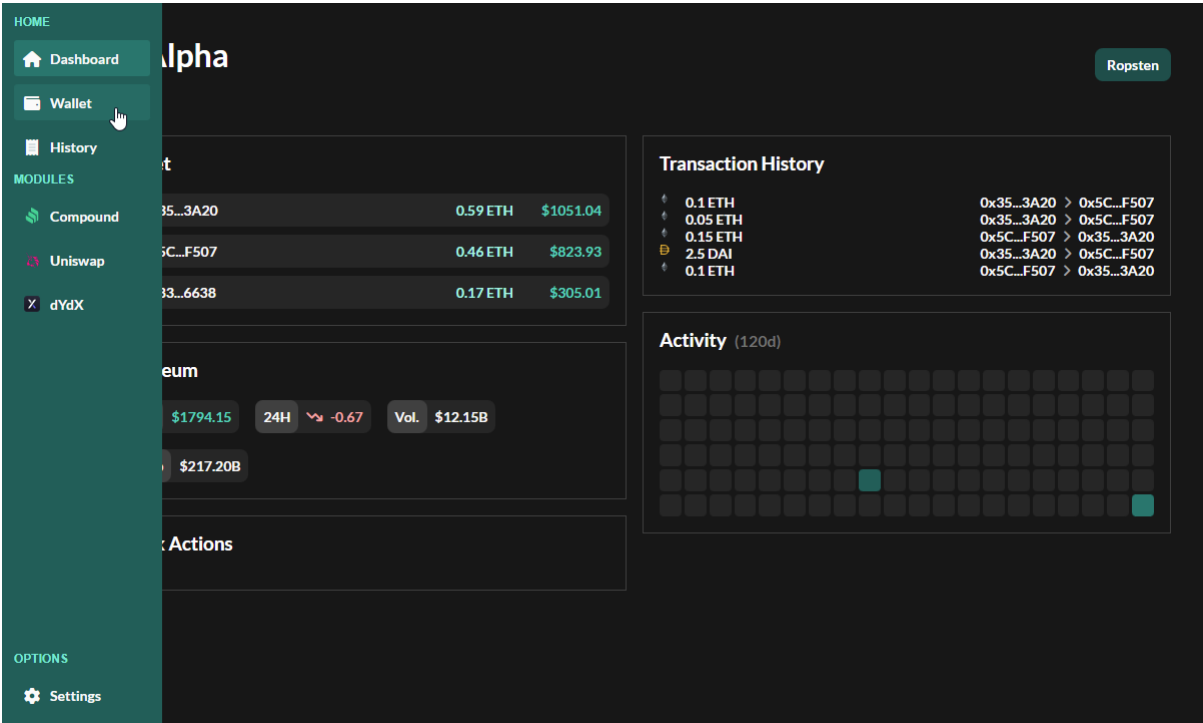


Figure 38: A user dashboard with slide window opened in left side

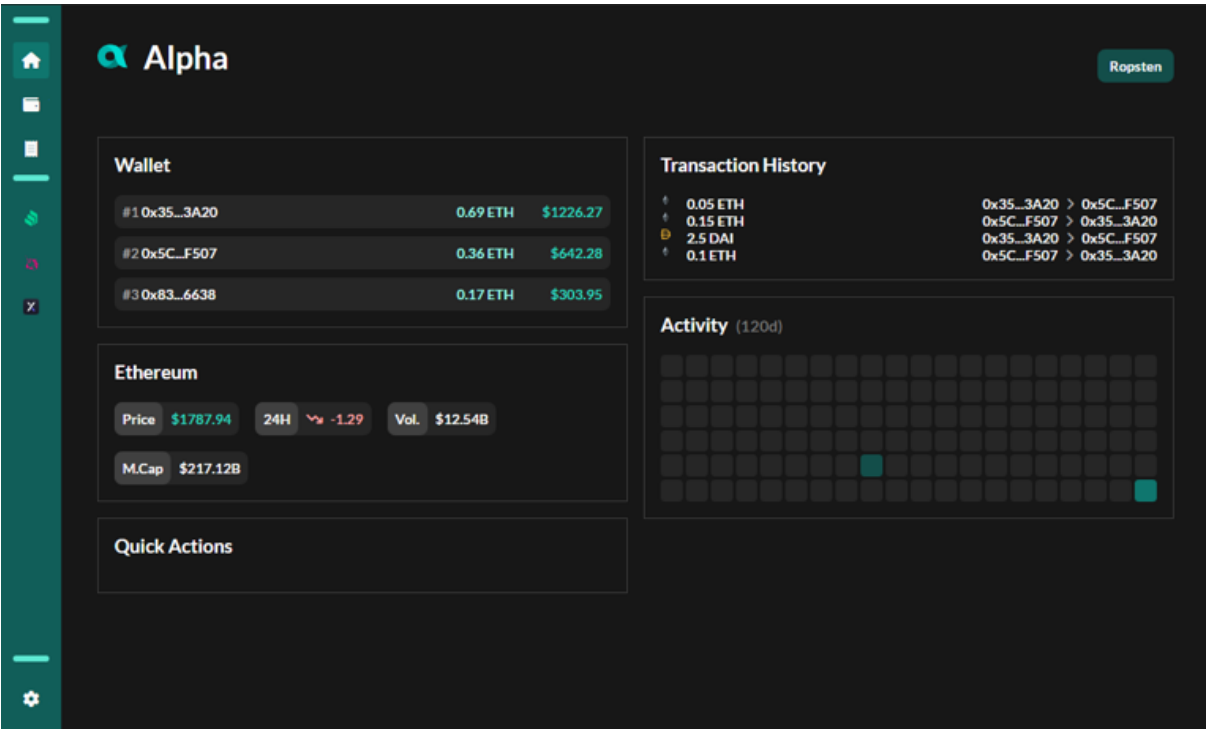
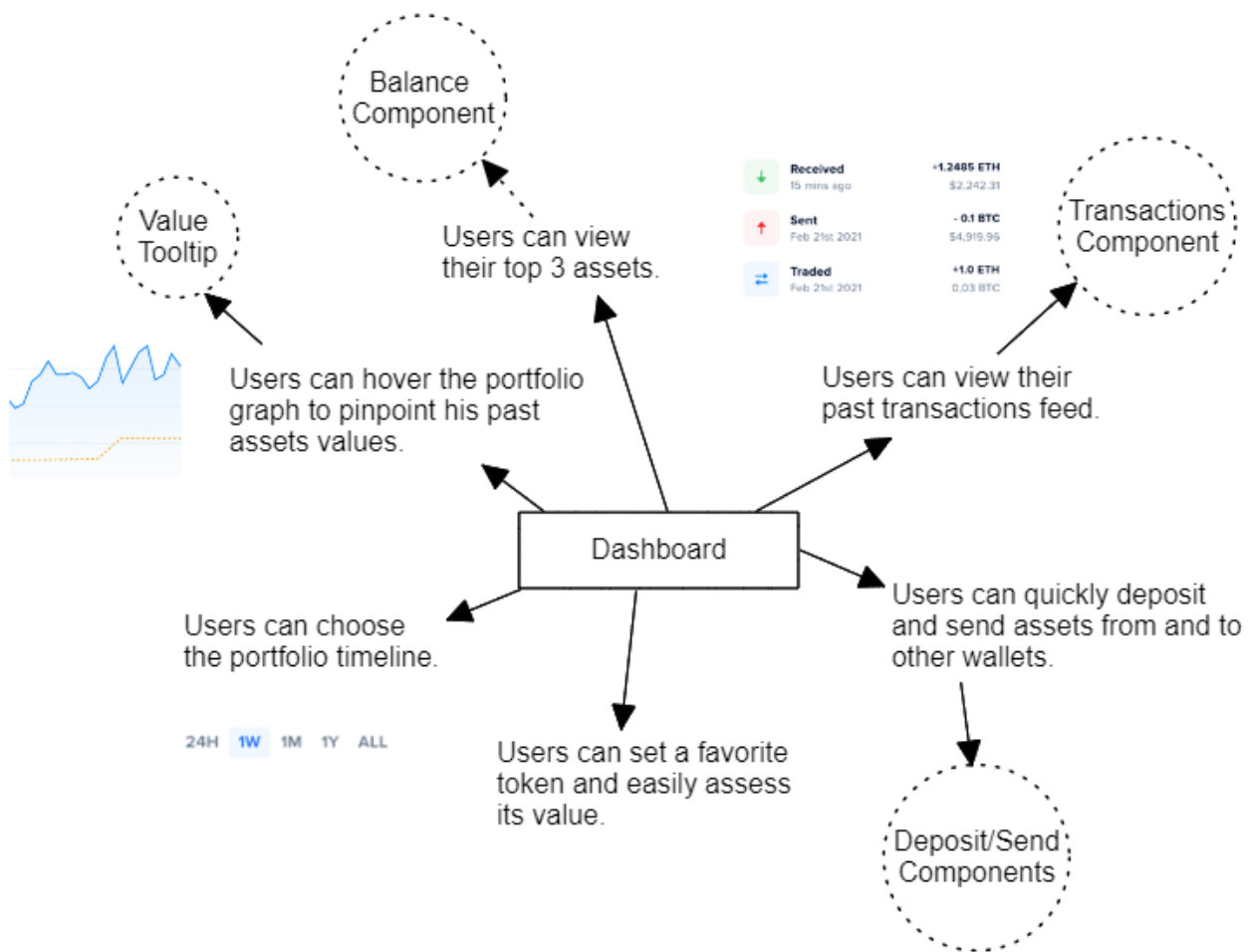


Figure 39: A user dashboard with closed slide window



## 6. Wallet

Users can view their balance and transfer assets to different addresses.

A user can change the amount of accounts he owns, a wallet can have up to 10 accounts. (Realistically a wallet can have an infinite amount of accounts.)

This example showcases transferring some amount of Ethereum from one account to another within the same wallet.

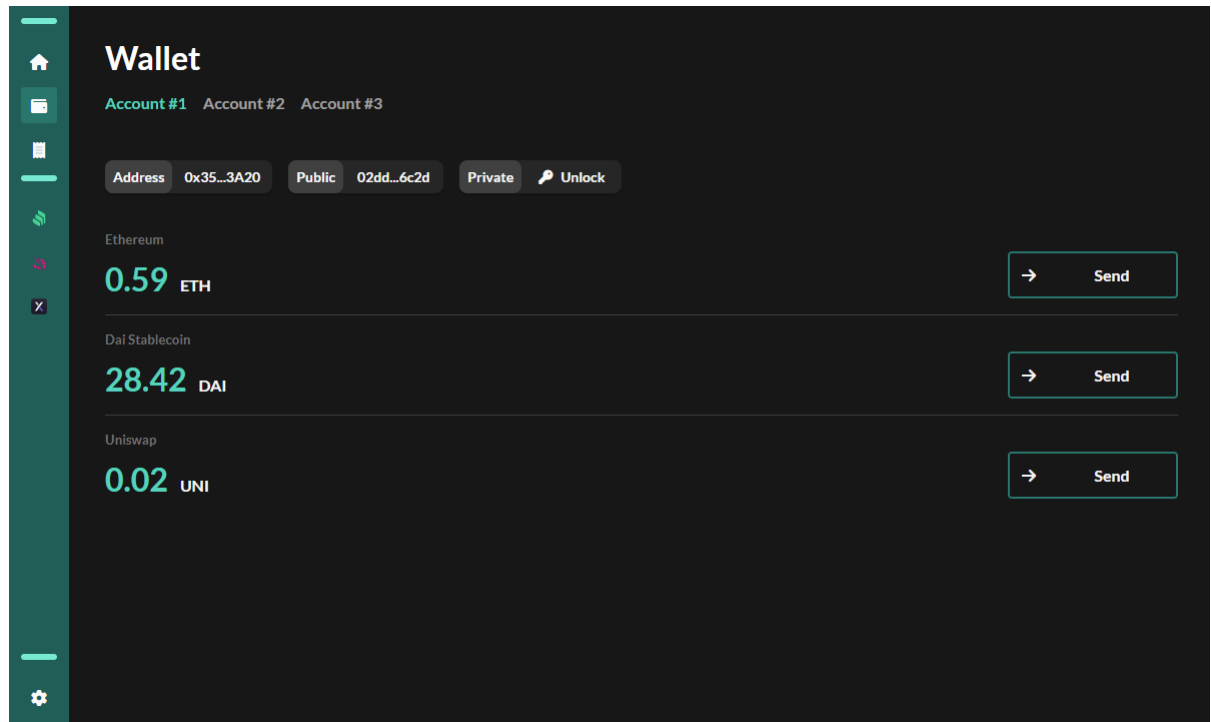


Figure 40: Account#1 user can copy the address by press on it for transaction

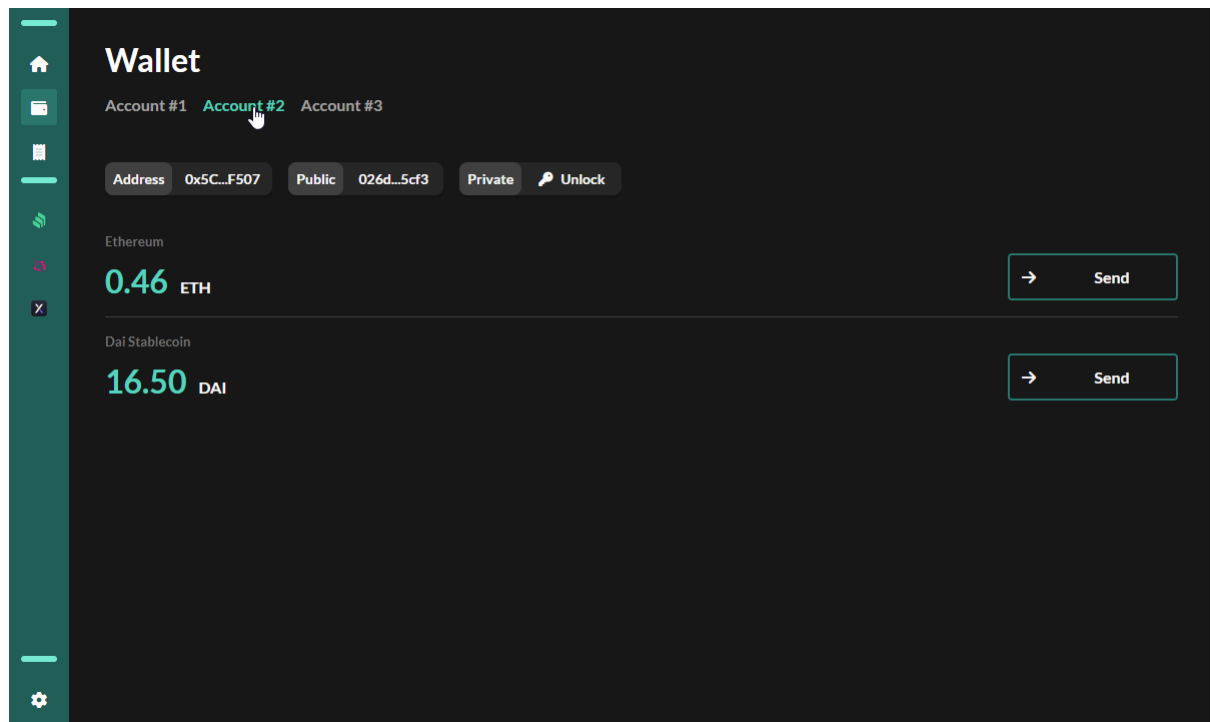


Figure 41: Account#2 user can see all tokens for each account



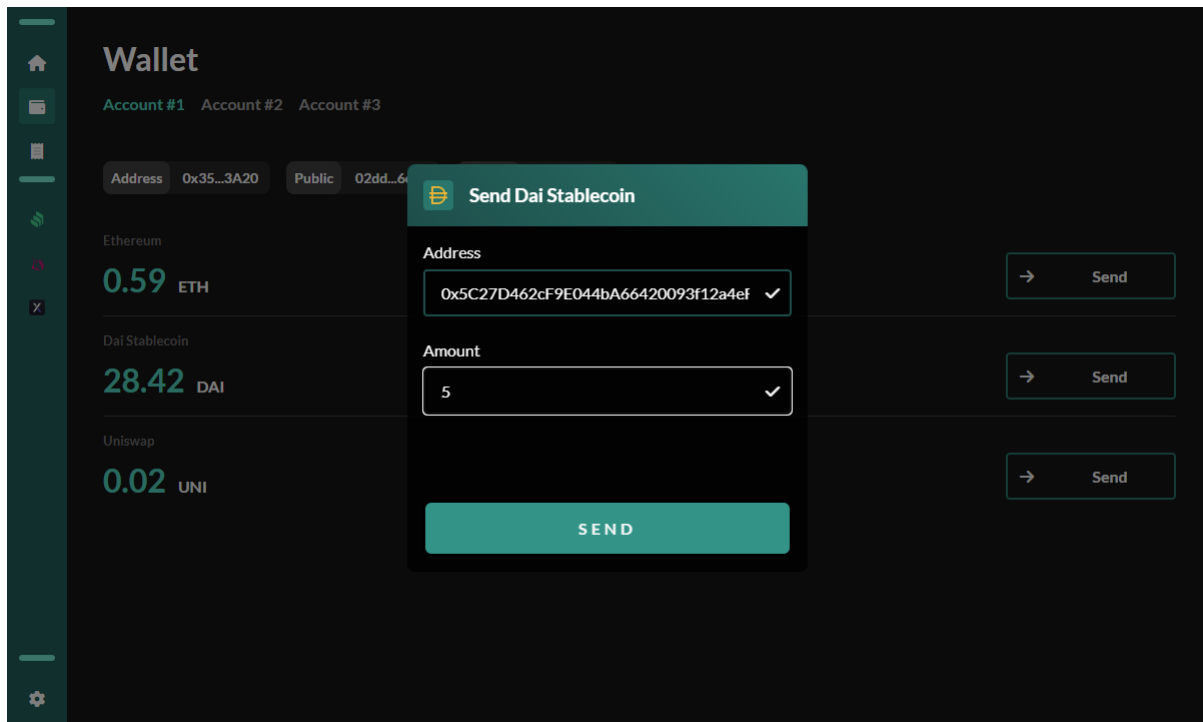


Figure 42: When the user wants to send a particular token, he will have to type in the destination address and amount of money to send

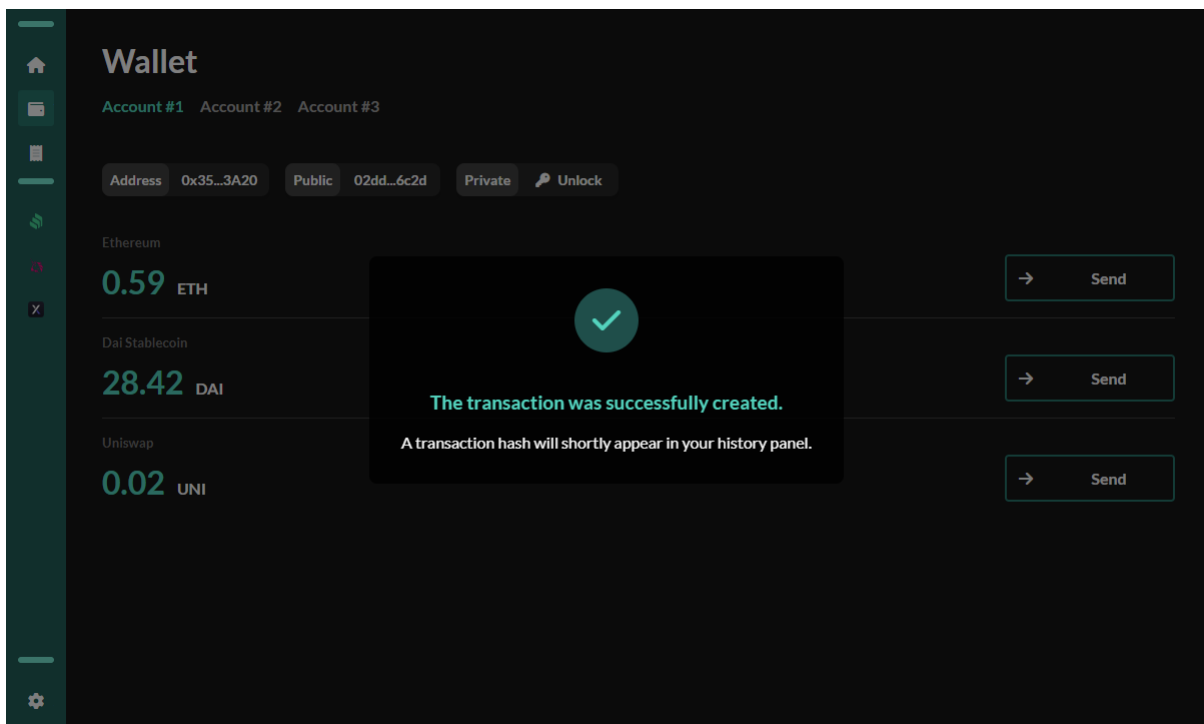
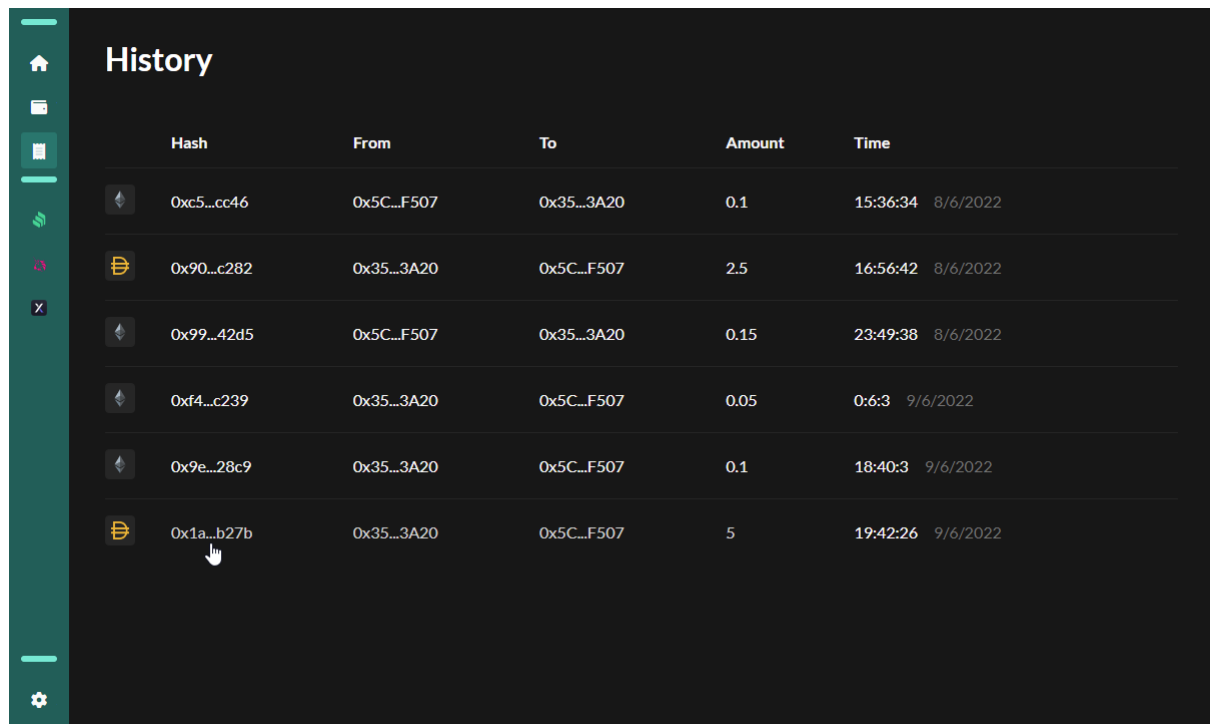


Figure 43: Feedback on the success of the transaction. In addition, we can see the sending record in history

## 7. Transaction History

Users can view previous transactions, whether a user sent assets directly to another address or performed an action using one of the DeFi services.









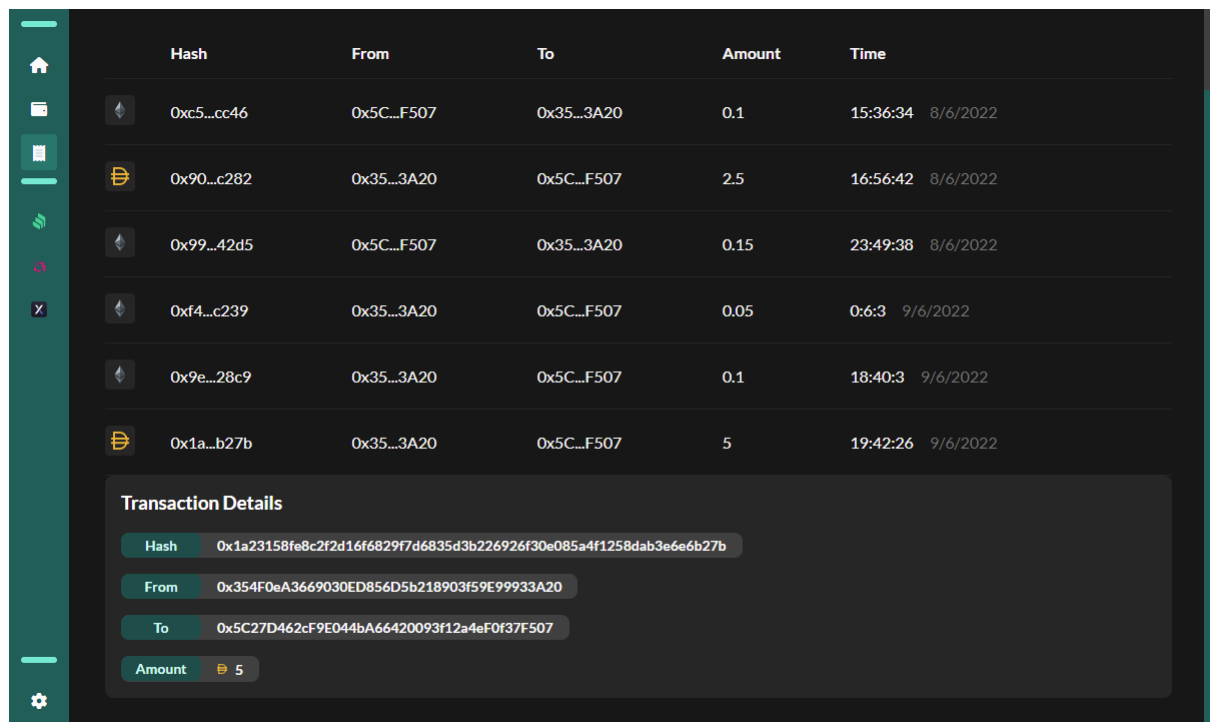






	Hash	From	To	Amount	Time
	0xc5...cc46	0x5C...F507	0x35...3A20	0.1	15:36:34 8/6/2022
	0x90...c282	0x35...3A20	0x5C...F507	2.5	16:56:42 8/6/2022
	0x99...42d5	0x5C...F507	0x35...3A20	0.15	23:49:38 8/6/2022
	0xf4...c239	0x35...3A20	0x5C...F507	0.05	0:6:3 9/6/2022
	0x9e...28c9	0x35...3A20	0x5C...F507	0.1	18:40:3 9/6/2022
	0x1a...b27b	0x35...3A20	0x5C...F507	5	19:42:26 9/6/2022

Figure 44: History of transactions, the user can open and see the details



	Hash	From	To	Amount	Time
	0xc5...cc46	0x5C...F507	0x35...3A20	0.1	15:36:34 8/6/2022
	0x90...c282	0x35...3A20	0x5C...F507	2.5	16:56:42 8/6/2022
	0x99...42d5	0x5C...F507	0x35...3A20	0.15	23:49:38 8/6/2022
	0xf4...c239	0x35...3A20	0x5C...F507	0.05	0:6:3 9/6/2022
	0x9e...28c9	0x35...3A20	0x5C...F507	0.1	18:40:3 9/6/2022
	0x1a...b27b	0x35...3A20	0x5C...F507	5	19:42:26 9/6/2022

Transaction Details

Hash 0x1a23158fe8c2f2d16f6829f7d6835d3b226926f30e085a4f1258dab3e6e6b27b

From 0x354F0eA3669030ED856D5b218903f59E99933A20

To 0x5C27D462cf9E044bA66420093f12a4eF0f37F507


Amount  5

Figure 45: Details of one of the transaction

## 8. Infura

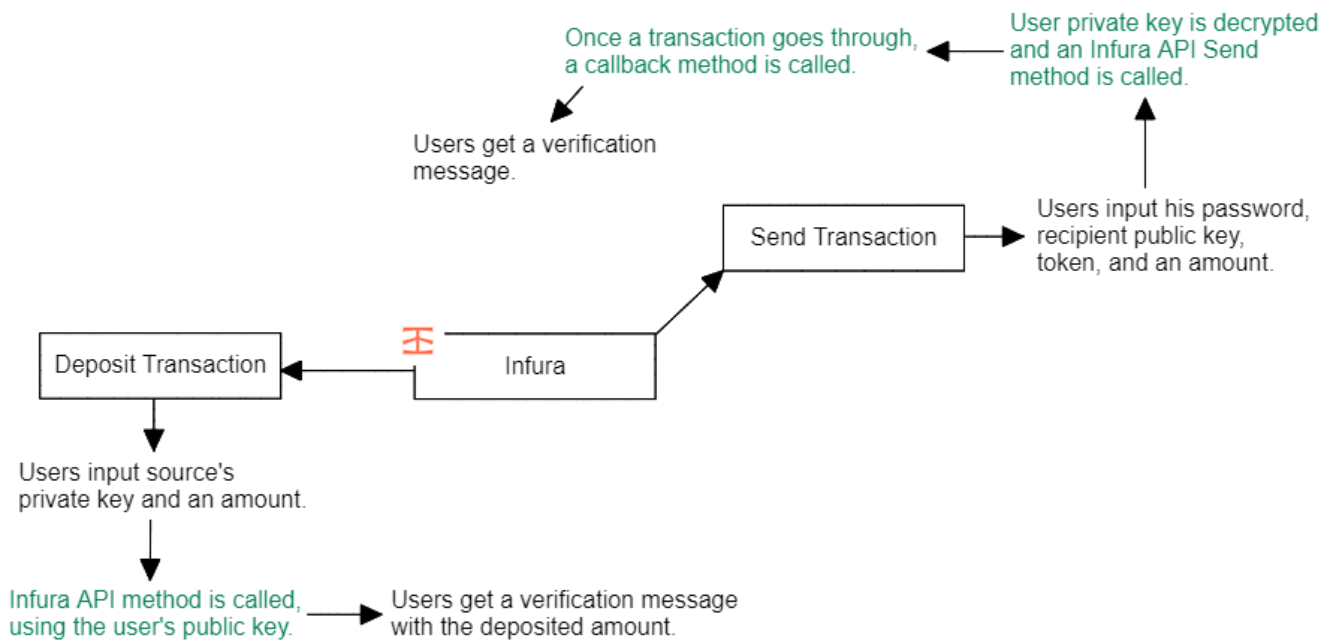


Figure 46: Different interactions based on the Infura API.

## 9. Compound

Compound is a decentralized, blockchain-based protocol that allows you to lend and borrow crypt.

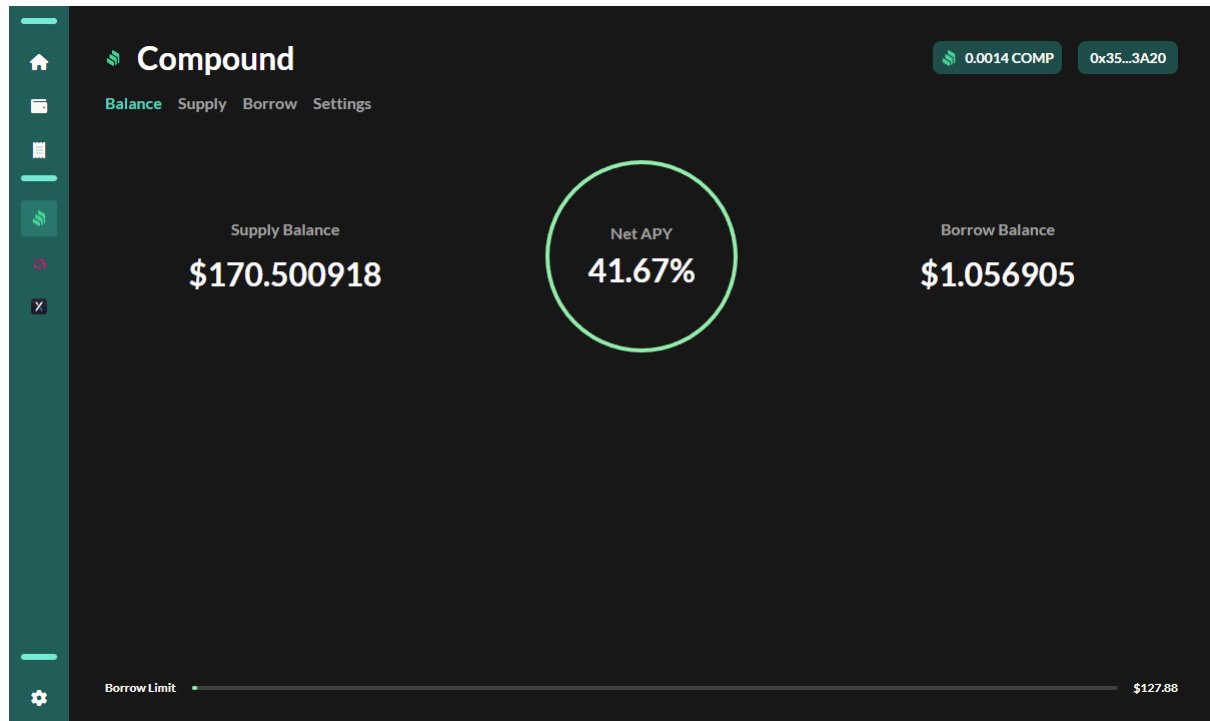


Figure 47: Service Compound ALPHA'S interface, user can choose and check the balance

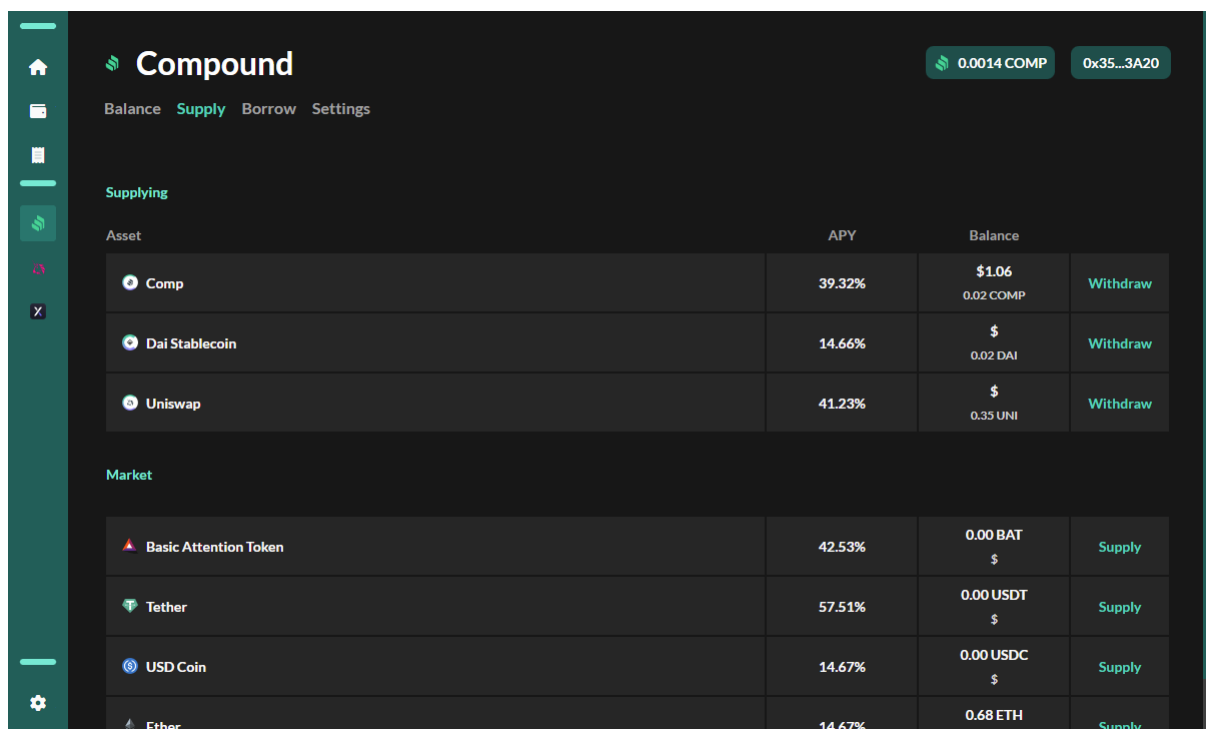


Figure 48: Service Compound, user can choose for supplying

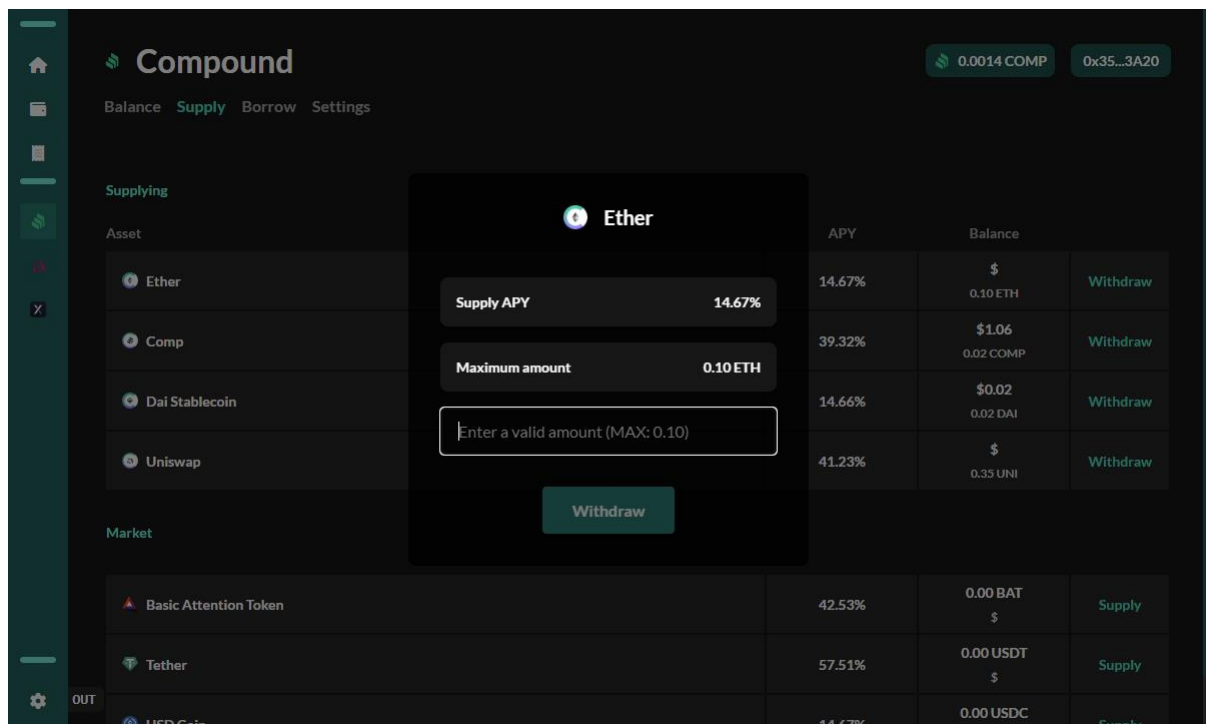


Figure 49: Compound supplying Ether in the additional window for entering the amount

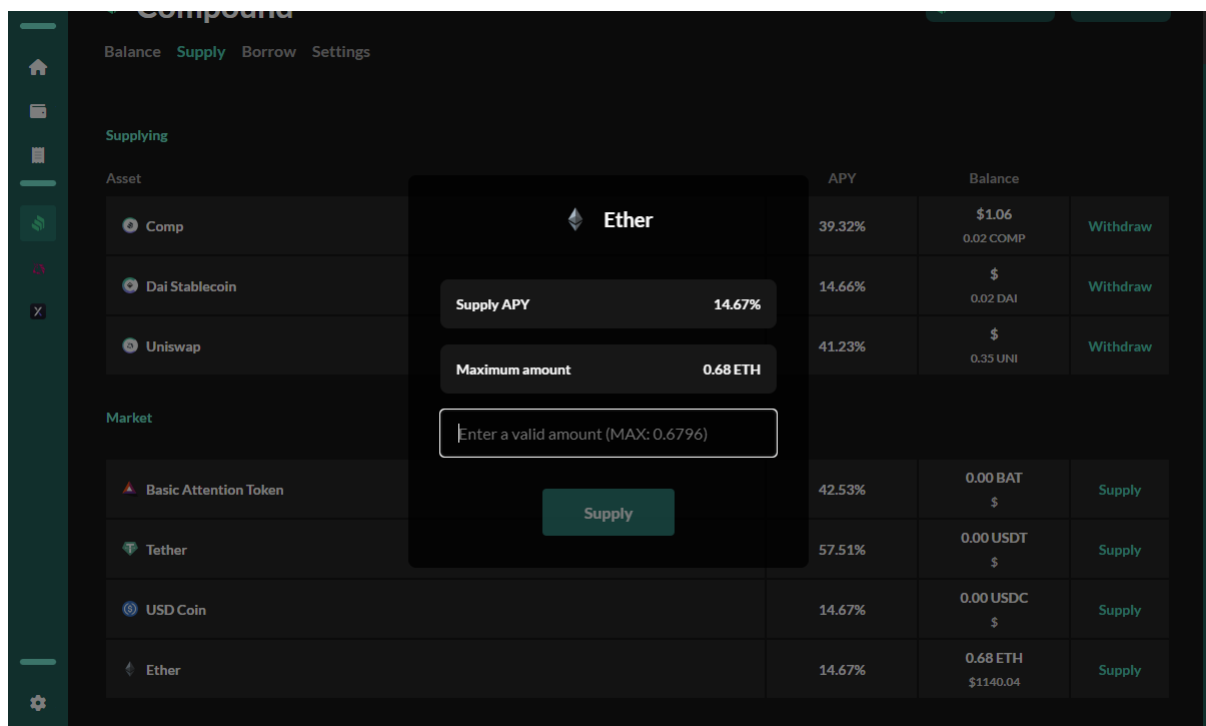


Figure 50: Compound supplying Ethereum in the additional window for entering the amount

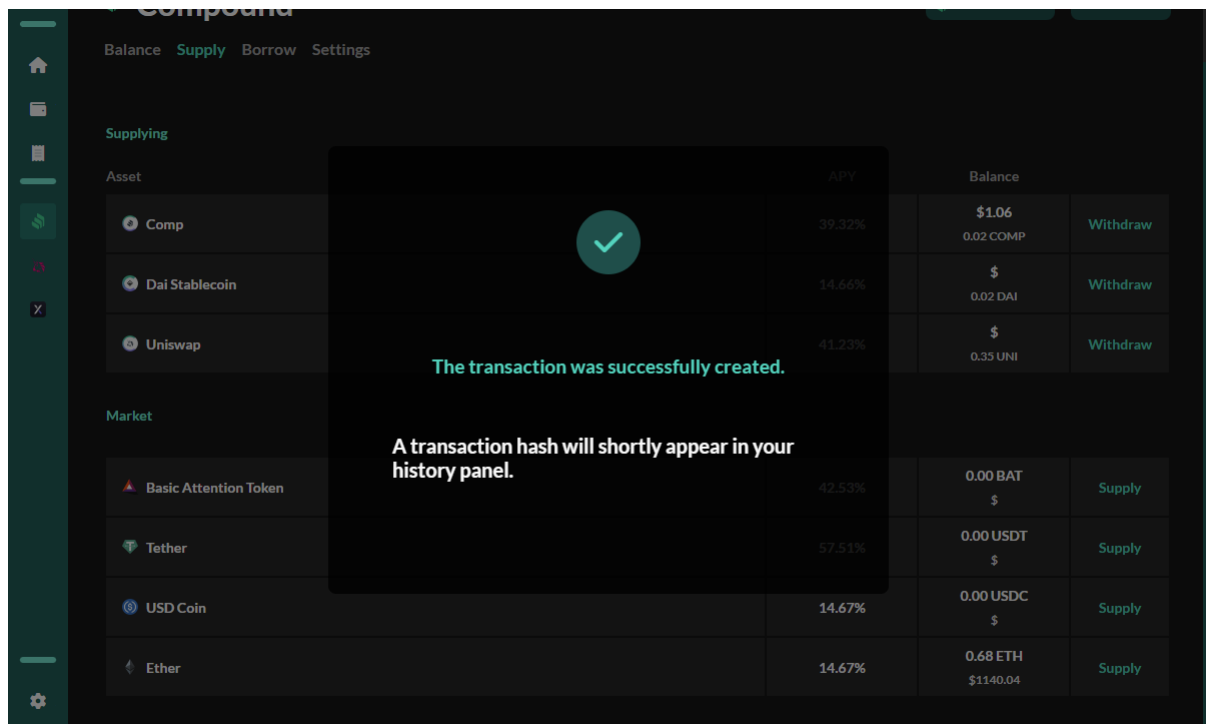


Figure 51: Feedback window of transaction complete successfully

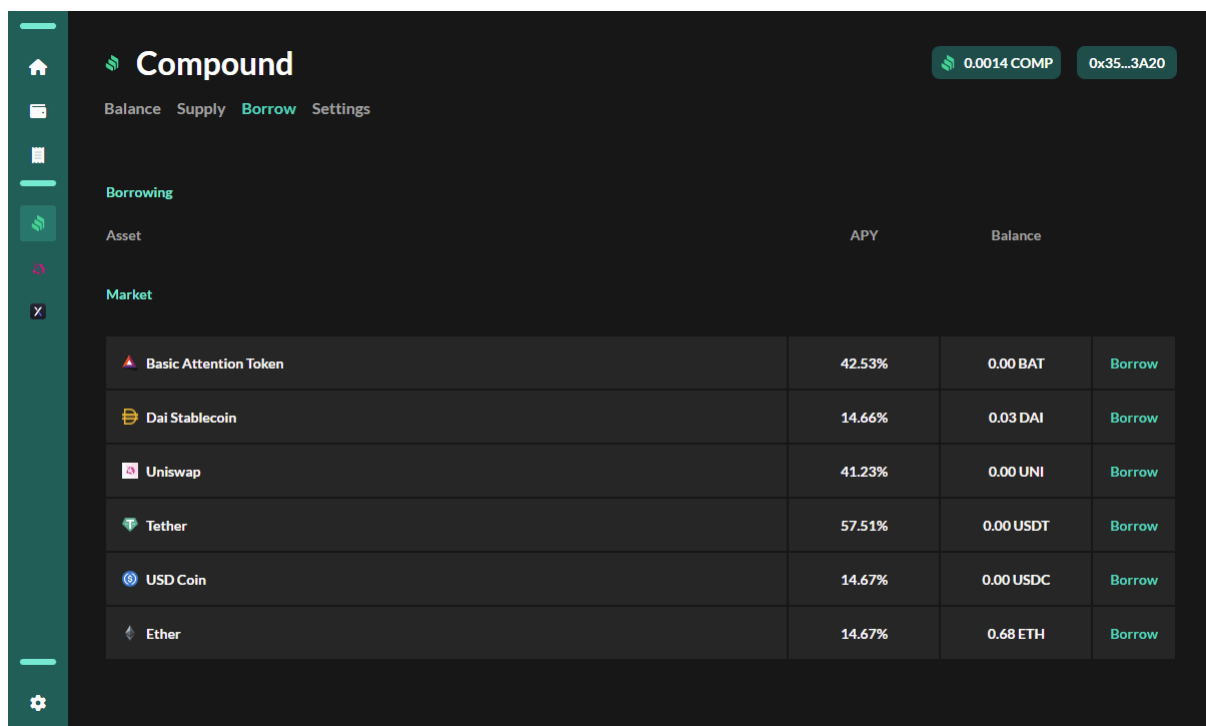


Figure 52: Compound Borrow option

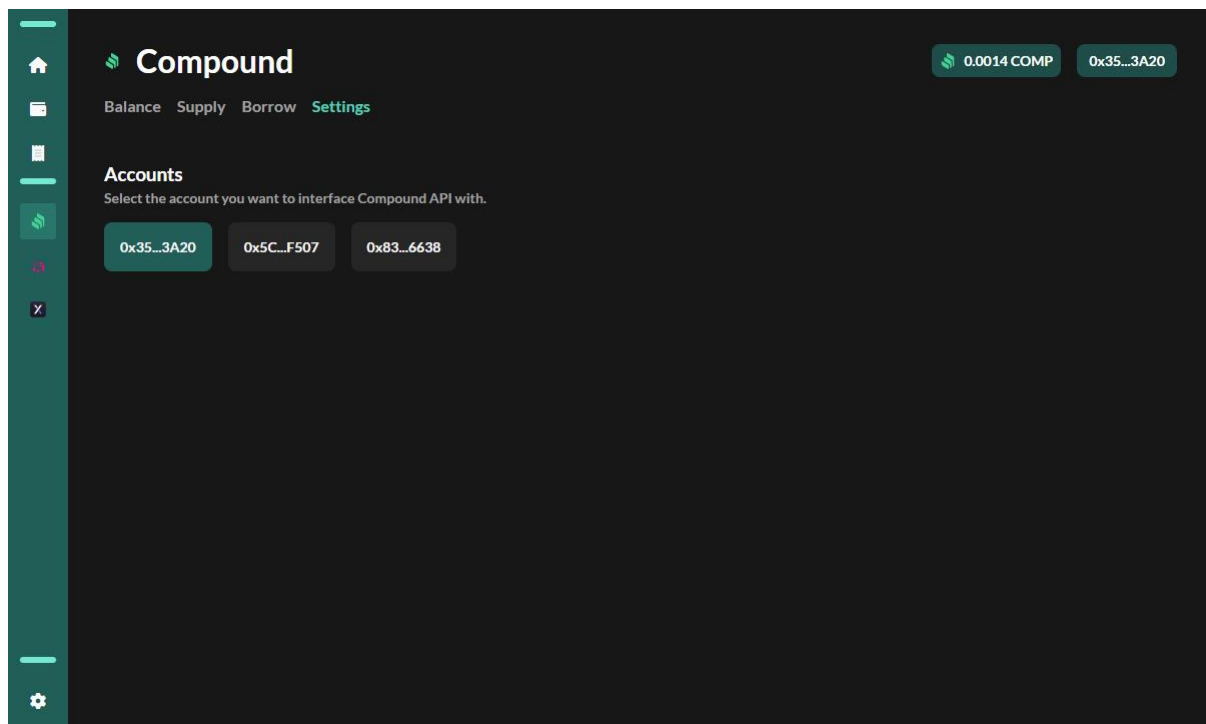


Figure 53: Compound settings, user can choose the relevant address

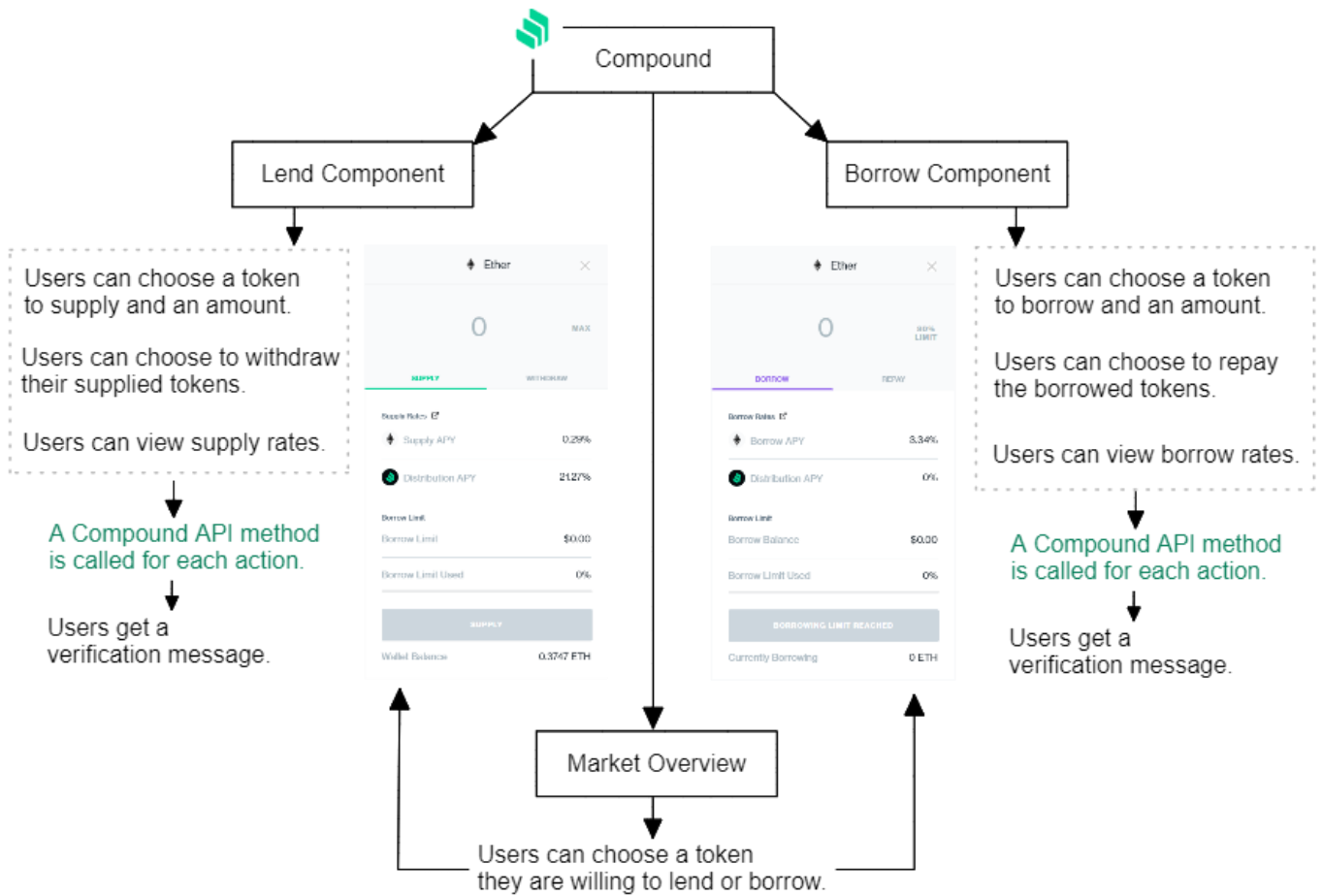


Figure 54: Shows all available actions Compound provides.



## 10. Uniswap

*It works for public welfare. It is a tool for members to exchange tokens easily without paying any platform charges or dealing with negotiators.*

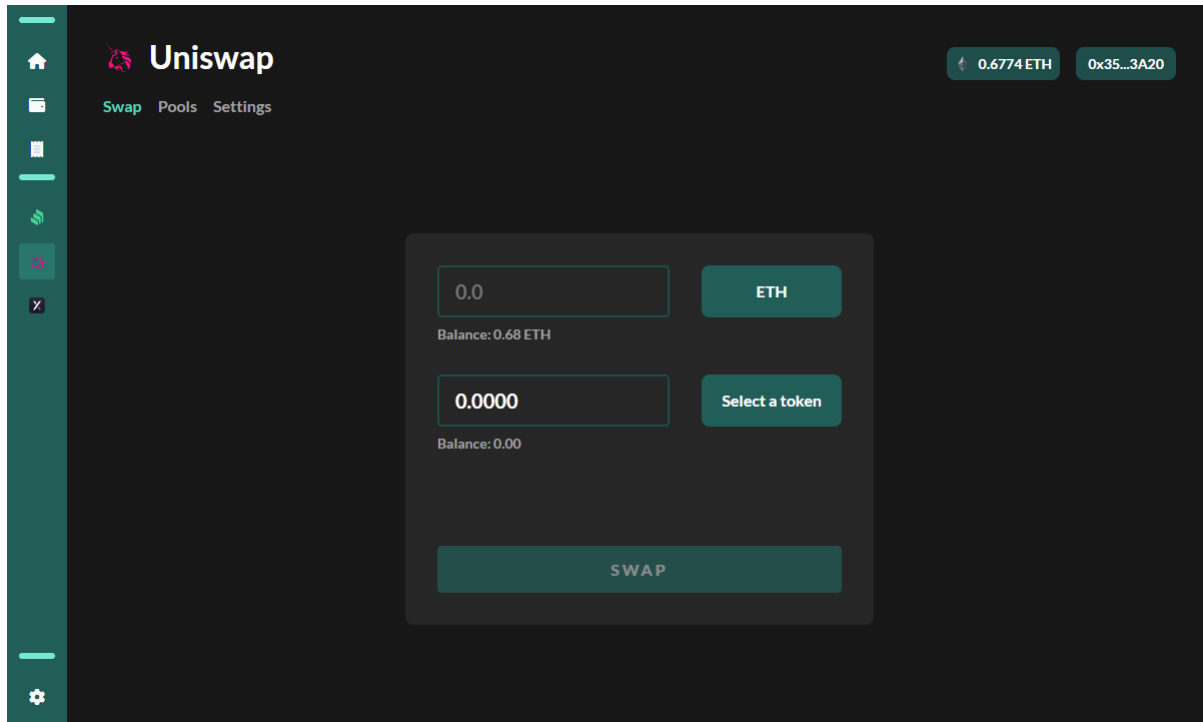


Figure 55: Uniswap ALPHA'S interface, user can select token for choose the relevant to exchange

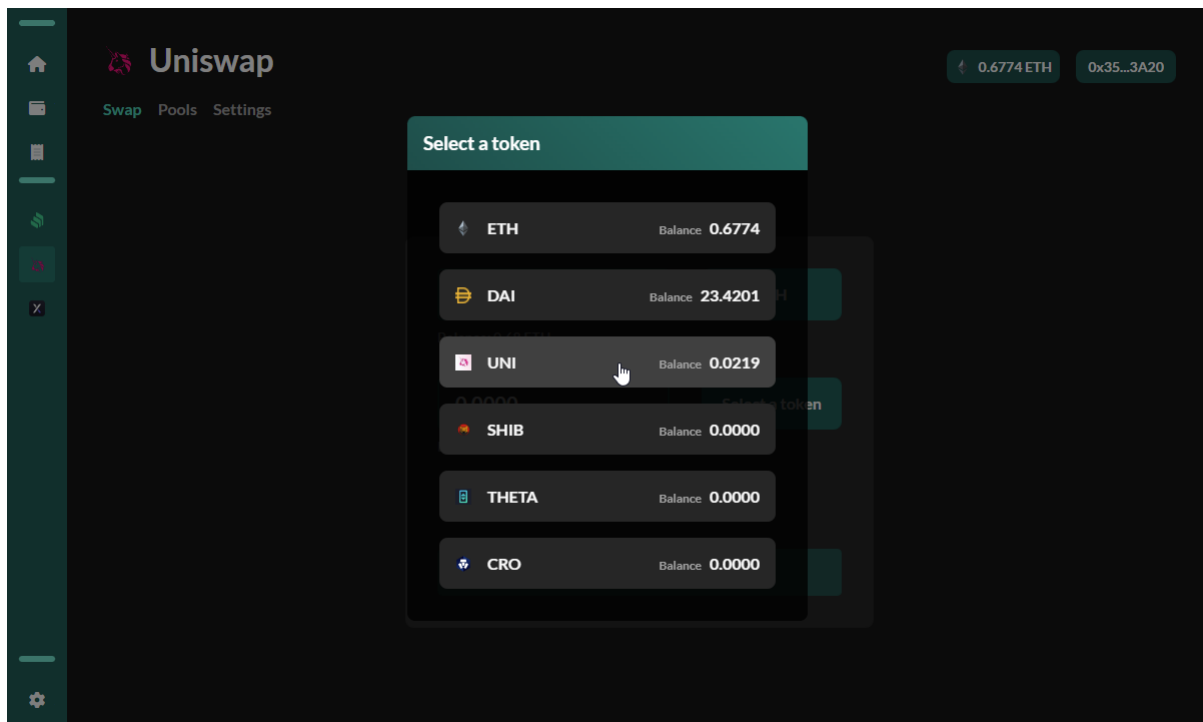


Figure 56: Uniswap selection window, user can see all the tokens that possible to convert to, and choose one of them

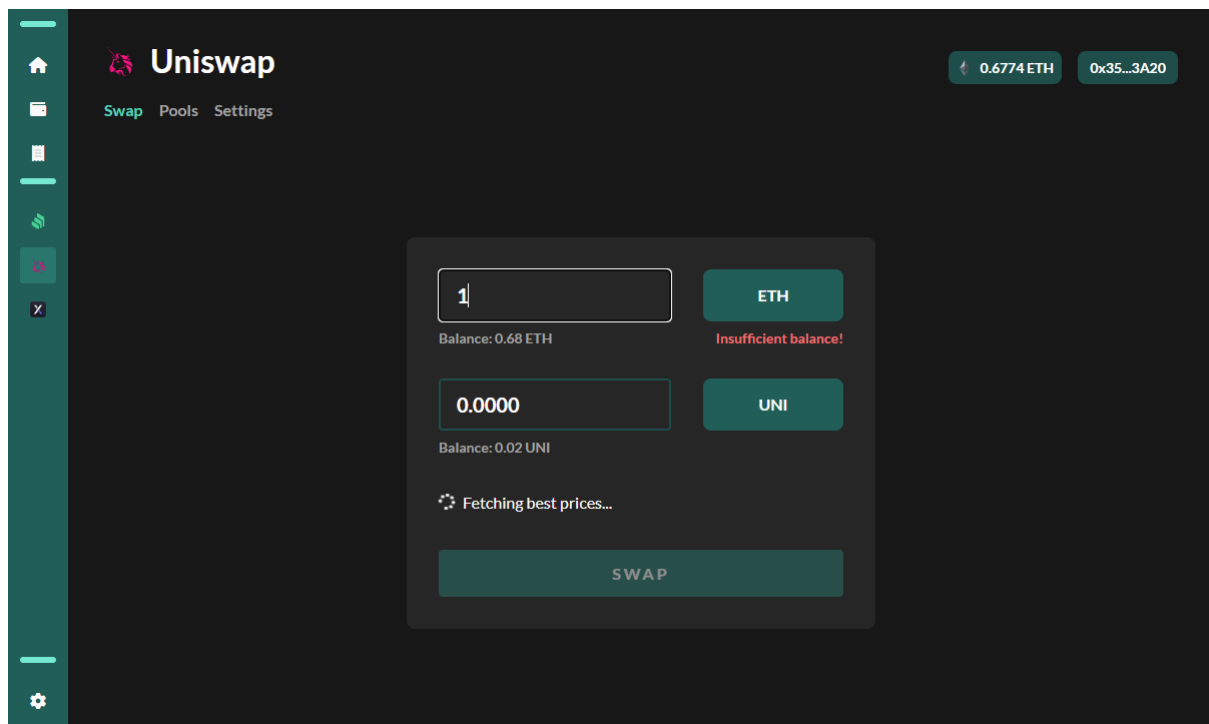


Figure 57: If the user selects the larger amount from his balance, he will be notified that he should select less

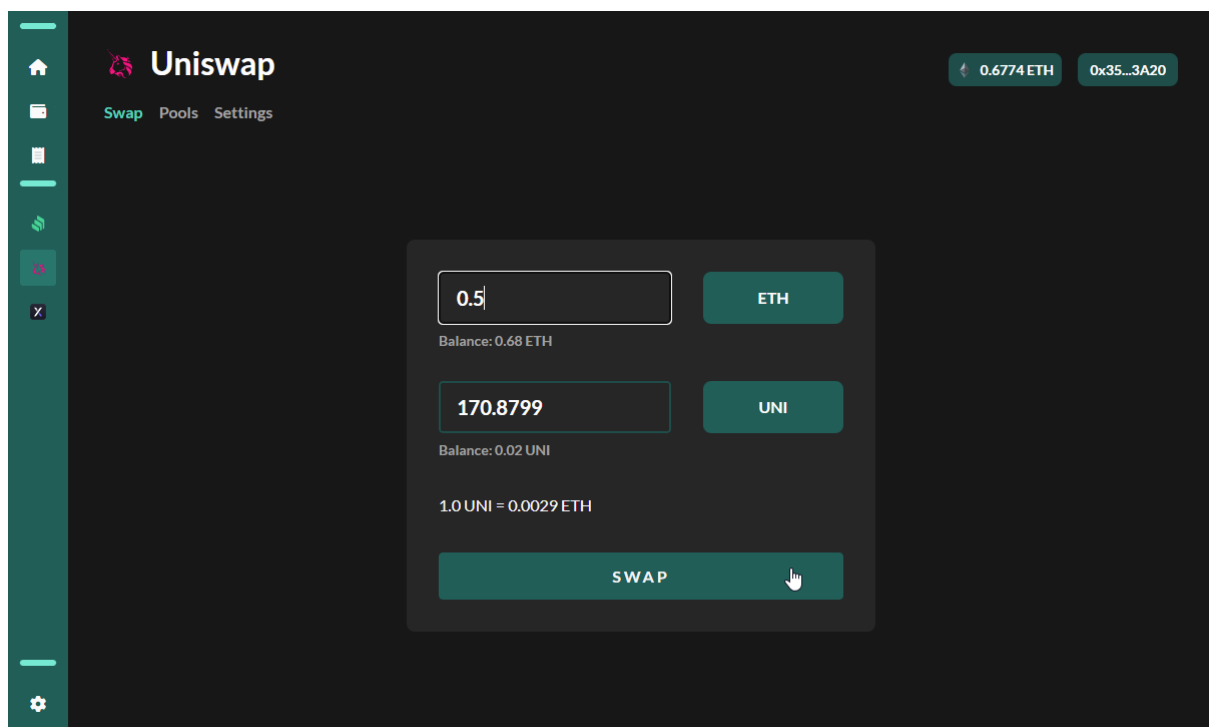


Figure 58: The correct amount for conversation

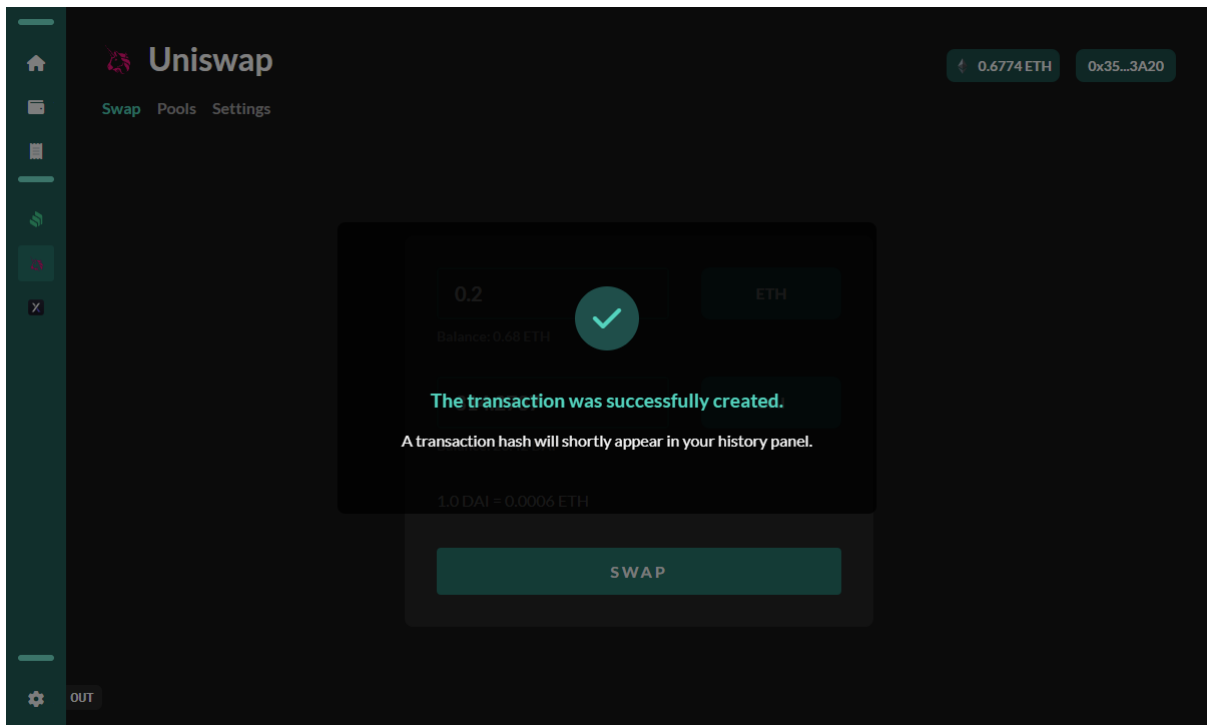


Figure 59: The transaction was finish successfully

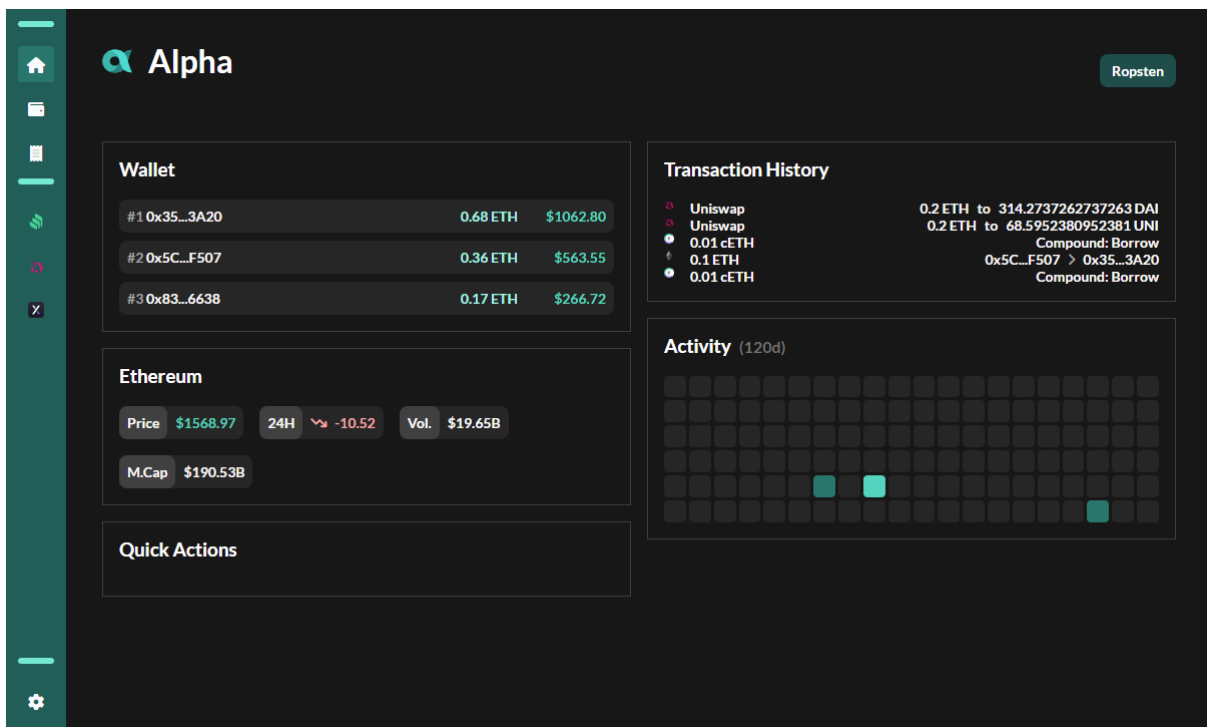


Figure 60: In main menu the user will be able to see the transaction record as history

Figure 61: In history the user will be able to see the transaction details

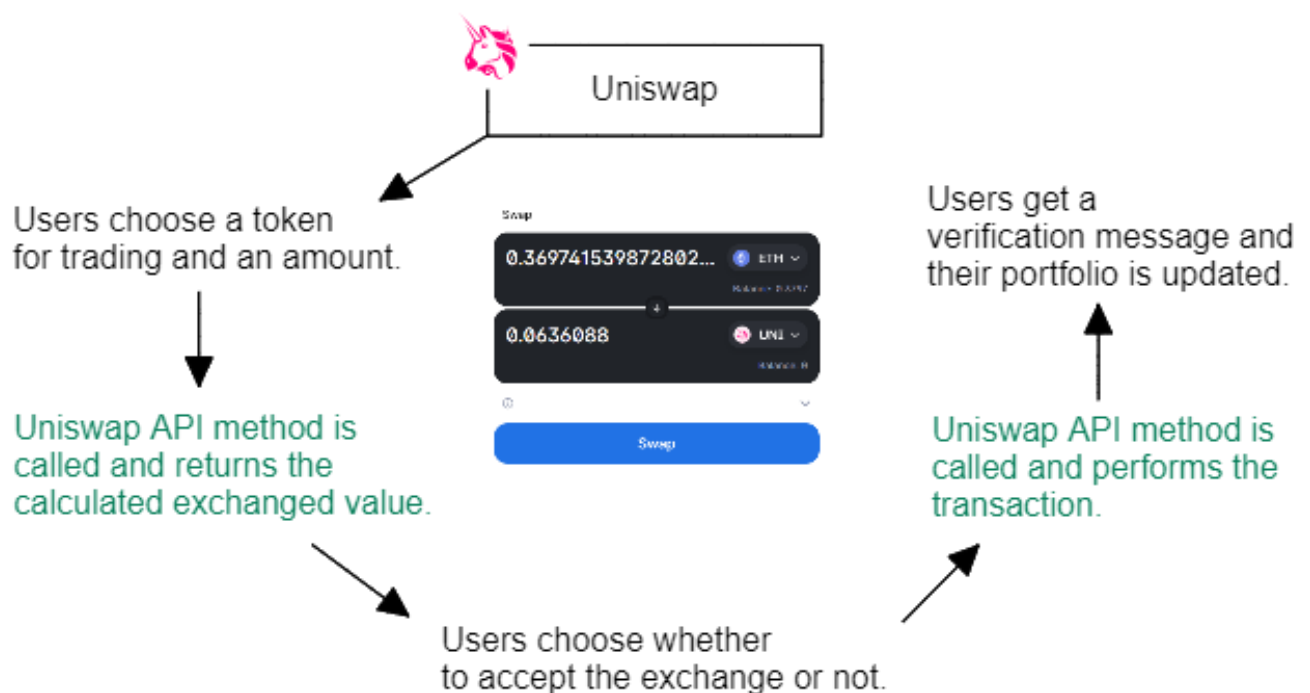


Figure 62: A simplistic sequence of actions within the Uniswap component.

## 11. Settings

Users can access and customize various settings.

For instance, managing different API keys, number of accounts per wallet, visual themes, and the integration of user-made modules, all taking effect immediately.

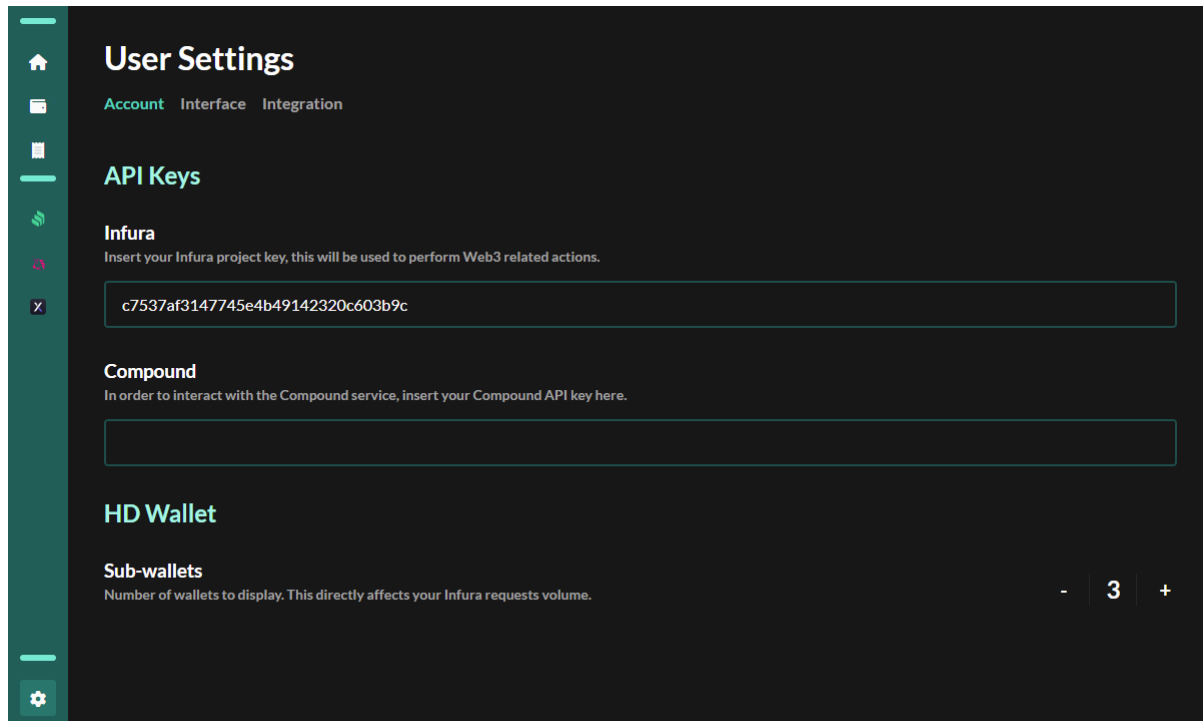


Figure 63: The settings account component, user can increase/decrease the number of accounts in wallet

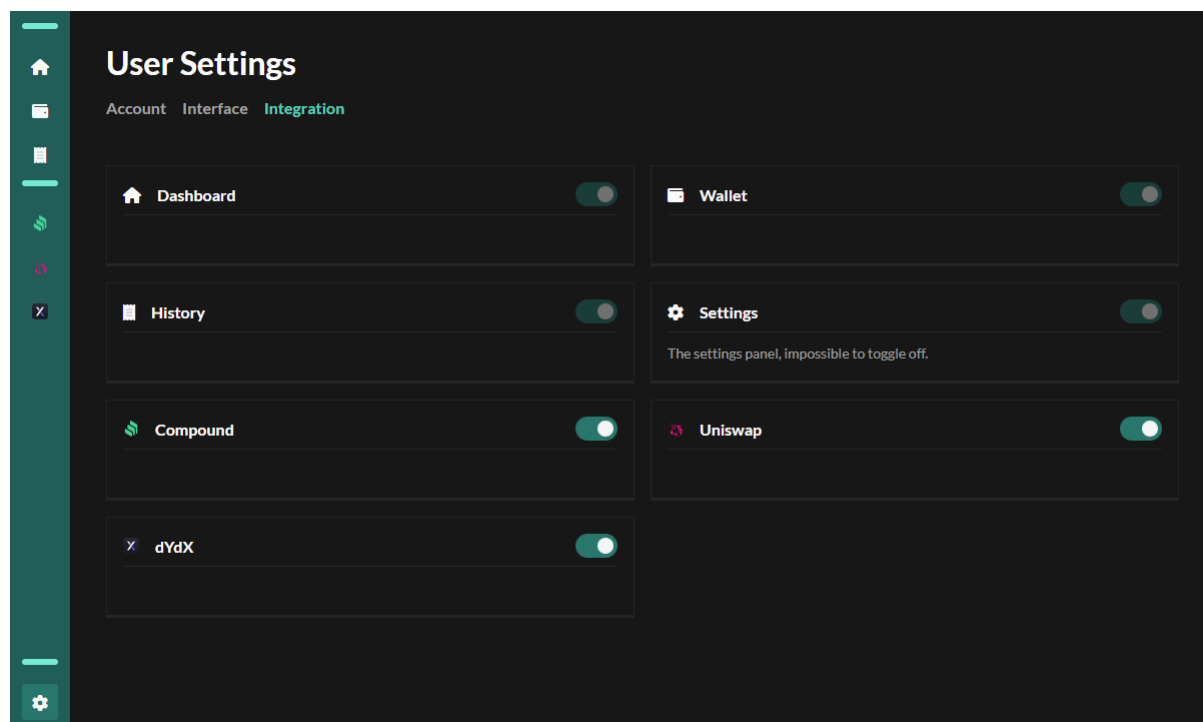
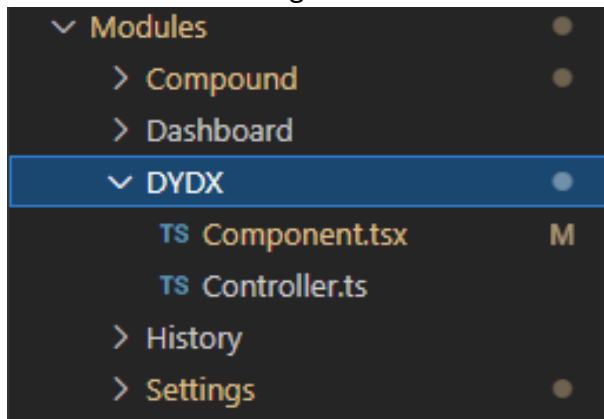


Figure 64: The settings integration component, user can ON/OFF the service

## 12. DYDX (optional)

*In our project we can relatively easily add different services that will be used from a wallet, the user needs to perform some actions in our code and know the API of the specific service he is interested in, here we will describe the actions required in our code:*

1. A developer adds a new folder for the module, with a Component file and an optional Controller for extra logic.



2. The new module extends a predefined class of Abstract Module, providing the developer with many built-in features.

```
1 |
2 | import { AbstractModule } from "../AbstractModule";
3 |
4 | class DXDY extends AbstractModule {
5 |   constructor() {
6 |     super();
7 |
8 |     this.Name = 'dYdX';
9 |     this.Category = 'Modules';
10 |    this.SVG = '/logos/dydx.svg';
11 |
12 |    this.Component = <></>
13 |
14 |    this.PostConstructor();
15 |   }
16 | }
17 |
18 | export default DXDY;
19 |
```

3. All that is left is to instantiate the module and add it to the list of all modules. And that's it, you're done.

```
8 export const InitModules = () => {
9
10     Add(new Dashboard());
11     Add(new Wallet());
12     Add(new History());
13     Add(new Settings());
14     Add(new Compound());
15     Add(new Uniswap());
16     Add(new DYDX());
17
18 }
```

4. The new module is automatically integrated and seamlessly appears in the extension.

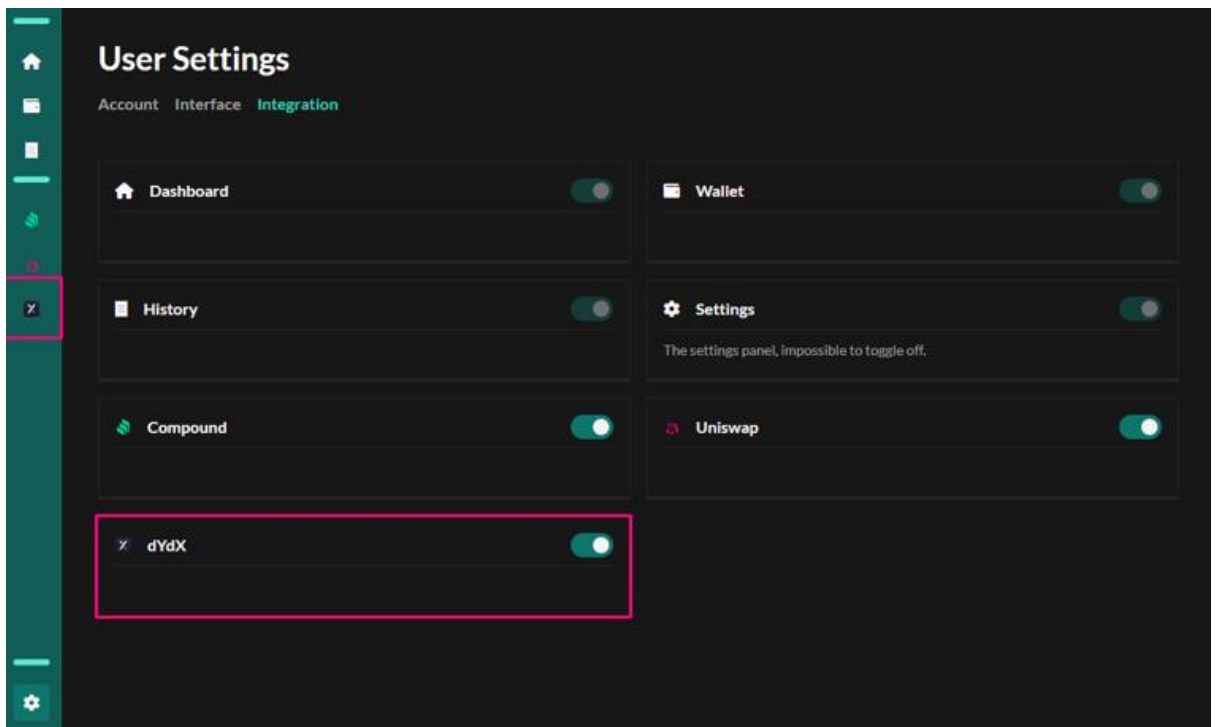


Figure 65: At the result user can see the service added successfully

### 5.2.5 Deployment

Acquiring the extension can be done via either cloning the git repository or using the Chrome Web Store.

For deployment developers should:

1. Build the repository using Node Package Manager by running  
`npm run build`
2. Enable Developer Mode in the Chrome Extensions settings panel
3. Load the unpacked directory directly



## 6 Evaluation and Verification

### 6.1 Testing

Testing will be conducted on two separate levels:

1. **Extension level**
2. **Blockchain level**

For frontend (extension level) testing, we plan on writing unit tests, using multiple JavaScript-based testing frameworks, such as JestJS or Cypress.



```
• toBe multi-line strings

expect(received).toBe(expected) // Object.is equality

- Expected   - 1
+ Received   + 1

Roses are red. Violets are blue.
- Testing with Jest is good for you.
+ Testing your luck is bad for you.
```

Figure 66: Example of the JestJS CLI tool.

We would also like to test our extension on real human testing group and ask for feedback about the intuitiveness of our system.

For backend (Blockchain level) testing, all of our API logic will run locally using the Ganache Testing Suite as well as a real decentralized testing network such as Ropsten TESTNET.

Ganache

ACCOUNTS

BLOCKS

TRANSACTIONS

CONTRACTS

EVENTS

LOGS

SEARCH FOR BLOCK NUMBERS OR TX HASHES

CURRENT BLOCK  
6

GAS PRICE  
20000000000

GAS LIMIT  
6721975

HARDFORK  
MURGLACIER

NETWORK ID  
5777

RPC SERVER  
HTTP://127.0.0.1:7545

MINING STATUS  
AUTOMINING

WORKSPACE  
TRUFFLE-SHUFFLE

SWITCH

MNEMONIC

candy maple cake sugar pudding cream honey rich smooth crumble sweet treat

HD PATH

m/44'/60'/0'/0'/account\_index

ADDRESS

0x627306090abaB3A6e1400e9345bC60c78a8BEf57

BALANCE

99.46 ETH

TX COUNT

32

INDEX

0

ADDRESS

0xf17f52151EbEF6C7334FAD080c5704D77216b732

BALANCE

100.00 ETH

TX COUNT

0

INDEX

1

ADDRESS

0xC5fdf4076b8F3A5357c5E395ab970B5B54098Fef

BALANCE

100.00 ETH

TX COUNT

0

INDEX

2

ADDRESS

0x821aEa9a577a9b44299B9c15c88cf3087F3b5544

BALANCE

100.00 ETH

TX COUNT

0

INDEX

3

ADDRESS

BALANCE

TX COUNT

INDEX

Figure 67: Example of Ganache GUI.

### Extension Level Testing:

We used Jest, a JavaScript based unit testing framework.



Test Case Examples	■ Expected	■ Failed
Importing an existing wallet	Valid seed phrase, user can proceed	An error is displayed, can't proceed
Making a transaction	Valid address, user can send assets	User receives an error message
Supplying assets to Compound	Amount equal or lesser than wallet's	An error is displayed, can't supply assets
Swapping assets with Uniswap	Valid asset amounts in wallet	An error is shown, can't exchange

### Blockchain Level Testing

For backend testing, we initially used Ganache, a local Blockchain testing suite, we then tested our results using Ropsten, a real testing network and the original DeFi services.



Test Case Examples	■ Expected	■ Failed
Asset transaction (local)	Valid signed transaction hash	An error is displayed in the console
Asset transaction (live)	Successful transaction between wallets	An error message on Etherscan.io (real Blockchain tracker)
Uniswap exchange rates	Same exchange rates for all assets	Mismatch between the original service and the application
Compound yield rates	Same APY when supplying assets to the protocol	Mismatch between the original service and the application

## 6.2 Evaluation

Users may be tasked with filling periodic surveys about various parts of the extension and provide any additional textual feedback regarding their user experience or any missing or faulty features within the Alpha extension.

## 7 References

- [1] The Bitcoin Standard, The Decentralized Alternative to Central Banking, 2018, S. Ammous.
- [2] How to DeFi, 2020, D. Lau, D. Lau, T. S. Jin, K. Kho, E. Azmi, T. Lee and B. Ong.
- [3] "Alexandria," CoinMarketCap OpCo, LLC, [Online]. Available: <https://coinmarketcap.com/alexandria/glossary>.
- [4] <https://freemanlaw.com/>
- [5] <https://blockgeeks.com/>
- [6] <https://ethereum.org/>
- [7] <https://www.cmcmarkets.com/en/learn-cryptocurrencies>
- [8] <https://www.investopedia.com/terms/s/smart-contracts.asp>
- [9] <https://compound.finance/>
- [10] <https://uniswap.org/>
- [11] <https://dydx.exchange/>
- [12] <https://medium.com/the-green-light/https-medium-com-the-green-light-moving-from-web2-to-web3-cf3cd4ac1a62>
- [13] <https://river.com/learn/terms/>
- [14] <https://infura.io/dashboard>
- [15] <https://ethereum.org/en/developers/docs/intro-to-ethereum/>