DECENTRALIZED CRYPTO EXCHANGE

A PROJECT REPORT

Submitted by

ABHILASH JOHN (TKM20MCA-2001)

to

The APJ Abdul Kalam Technological University

In partial fulfilment of the requirements for the award of the degree of

MASTER OF COMPUTER APPLICATIONS

Thangal Kunju Musaliar College of Engineering
Kerala

DEPARTMENT OF COMPUTER APPLICATIONS

JULY 2022

DECLARATION

I undersigned hereby declare that the project report on ”DECENTRALIZED CRYPTO EXCHANGE” submitted for partial fulfilment of the requirements for the award of the degree of Master of Computer Applications of the APJ Abdul Kalam Technological University, Kerala is a bonafide work done by me under the supervision of Prof. Natheera Beevi M. This submission represents my ideas in my own words and where ideas or words of others have been included, I have adequately and accurately cited and referenced the sources. I also declare that I have adhered to ethics of academic honesty and integrity and have not misrepresented or fabricated any data or idea or fact or source in the submission. I understand that any violation of the above will be a cause for disciplinary action by the institute and/or the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been obtained. This report has not previously formed the basis for the award of any degree, diploma or similar title of any other University.

Kollam

01/07/2022

Abhilash John

DEPARTMENT OF COMPUTER APPLICATIONS

TKM COLLEGE OF ENGINEERING

C E R T I F I C A T E

This is to certify that, the project report entitled **"DECENTRALIZED CRYPTO EXCHANGE"** is submitted by **ABHILASH JOHN (TKM20MCA-2001)** to the APJ Abdul Kalam Technological University in partial fulfilment of the requirements for the award of the degree of Master of Computer Applications is a bonafide record of the project work carried out by him under our guidance and supervision. This report in any form has not been submitted to any other University or Institute for any purpose.

Internal Supervisor | Head of the Department | External Examiner

ACKNOWLEDGEMENT

First and foremost I thank GOD almighty and my parents for the success of this project. I owe sincere gratitude and heart full thanks to everyone who shared their precious time and knowledge for the successful completion of my project.

I am extremely grateful to **Prof. Dr Fousia.M.Shamsudeen**, Head of the Department, for providing me with the best facilities.

I would like to thank my project guide **Prof. Natheera Beevi M**, Department of Computer Applications, who motivated me throughout the work of my project.

I profusely thank all other faculty members in the department and all other members of TKM College of Engineering, for their guidance and inspirations throughout my course of study.

I owe my thanks to my friends and all others who have directly or indirectly helped me in the successful completion of this project.

Abhilash John

ABSTRACT

A Decentralized Crypto Exchange is a place where people can go to trade cryptocurrencies without an intermediary. A DEX, lets people trade directly from their crypto wallets. The exchange doesn't take control of their private keys or act as an intermediary. Instead, it uses smart contracts(self-executing programs) to facilitate peer-to-peer exchanges. A decentralized exchange (DEX) lets crypto holders remain pseudonymous and keep control of their crypto wallet's private key. The idea behind a DEX is "disintermediation," which means removing middlemen to allow regular people to do business directly with each other. A DEX doesn't offer custody of users' crypto assets. Instead, users directly hold all their assets in their wallets at all times. The exchange is implemented with smart contracts on an Ethereum blockchain and deployed on an Ethereum test network. In addition to implementing transactions between individual users, it also allows transactions among multiple users.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Based on cryptographic proofs, Bitcoin was introduced in 2008 and now users can sell and buy it. Before Bitcoin, typical cashless trading systems relied on third parties and required them to clear transactions between two parties.

Unlike traditional trading systems, the entire Bitcoin network functions as a trusted third party to make transactions between two accounts. Nodes in the network verify the validity of processed transactions. This node generates a data record file that resembles a ledger that monitors account balances. Entries in that ledger are used, depending on their present states, to verify and update transactions.

Since Bitcoin uses a P2P (Peer to Peer) network to facilitate management rather than a single issuer, no one entity can entirely control it. Instead of being stored in a centralised database, its transaction data is written in a distributed hash chain known as the blockchain. A centralised exchange built on a trustworthy third party is widely used in recent works to help clients manage various cryptocurrency types. A centralised exchange serves as the venue for holding bitcoin assets and carrying out exchanges. The ease of managing funds and conducting transactions through a centralised exchange benefits users.

Users can trade conveniently on centralised exchanges, but there are certain security dangers as well. When users deposit their assets on a centralised exchange platform, the platform becomes the system's "Achilles' heel" and can be used maliciously to access user attributes and transaction data. Moreover, the centralized exchanges have a single point of failure at all times. Decentralized crypto exchanges are predicted to be the solution to this issue. Smart contracts built

on the Ethereum platform can be used to construct decentralised cryptocurrency exchanges and validate various cryptocurrency transactions sent by various users. The proposed decentralised cross-cryptocurrency exchange method is based on smart contracts and may conduct transactions between two different cryptocurrencies in single-user and multi-user situations using randomly chosen users as intermediaries. Through the use of proof of deposit and proof of labour, a review committee is created using a random selection of unreliable users.

1.1 Problem Definition

While centralised crypto exchanges make it convenient for customers to manage their money and carry out transactions using bitcoin, they also pose significant security threats. Although centralised exchanges are easy to use and typically the first stop for novice investors and traders, they have several drawbacks that are slowly eroding user appeal.

Once users store their assets on a centralised exchange platform, the platform becomes the system's ”Achilles Heel” and opens the door to criminal exploitation of users' assets and transaction data. There will always be a single point of failure because it is a major institution. Since centralised exchanges enable users to avoid transaction costs by merely reallocating user funds using a centralised database without actually performing any on-chain transactions, it is practical to keep cryptocurrencies in a location where they may also be utilised for trading. Users have no alternative but to wait until the exchange starts operations before they can access their funds because centralised exchanges do not permit users to access their private keys. Because they violate one of the fundamental principles of cryptocurrencies—that the owner of the private key is also the owner of the asset—centralized exchanges face a lot of issues. The largest exchanges, including Binance, OKEx, and Huobi, seize user cash and utilise them to manipulate the market. The custodian puts their interests before serving the needs of the consumer. Custodial transfers are comparable to a fox watching over a henhouse.

1.2 Objective

The main goal of the project is:

- To create a decentralised cross-cryptocurrency exchange system based on smart contracts that enables single-user and multi-user transactions and provides a platform for swapping different cryptocurrencies.

- To put the exchange into practice and make it available on an Ethereum test network.

- To create an Ethereum-based smart contract system that can authenticate various cryptocurrency transactions sent by various users .

Chapter 2

LITERATURE SURVEY

This section discusses several studies on decentralised cryptocurrency exchanges that use smart contracts on an Ethereum blockchain network.

Blockchain is a distributed ledger that keeps track of various system updates. A change could be any piece of information, such as the most recent currency exchange rates or just deposited commercial contracts. To prevent the system from becoming centralised and to make it more difficult for an adversarial party to easily manipulate the blockchain data, blockchain is typically dispersed among numerous independent parties. The blockchain is most famously employed in the decentralised peer-to-peer cryptocurrency Bitcoin, which uses it to keep track of all financial transactions among its users. Transaction recording, according to researchers, is the blockchain's most common application. But the blockchain's potential applications go well beyond that; they can also be used to run decentralised applications [1], store documents [2], for financial purposes, or for purposes unrelated to finance. The core of how blockchains are used financially is through smart contracts, which are short programmes run decentralised by all of the network's nodes.

The Ethereum platform is the most well-known system that uses smart contracts[3]. Ethereum leverages the blockchain and the Ethereum Virtual Machine, a Turing-complete programming language (EVM). Transactions change the states of this machine. A transaction on Ethereum can be used to transfer Ether, the native coin of the platform, in addition to activating smart contract features. Users must buy command-execution units called gas for the EVM to execute any transaction.

Every smart contract needs enough gas to pay for all of its instructions; otherwise, it will not be executed. This means that the user's purchase must pay for the operation of the EVM.

The Ethereum network has a standard interface for tokens called ERC-20 [4]. ERC20 is a decentralised cryptocurrency built on Ethereum's blockchain and Binance Smart Chain's cutting-edge innovation technology. A Token Standard called ERC-20 (Ethereum Request for Comments 20) defines an API for tokens used in smart contracts. It was first suggested by Fabian Vogelsteller in November 2015. ERC-20 allows different smart-contract enabled tokens a way to be exchanged. Tokens are a representation of anything that is not unique in and of itself but can be transferred, such as an asset, right, ownership, access, or money. The standard permits the exchange of tokens that represent one of these elements for another element as well as the use of smart contracts. Smart contracts are coding conditions that carry out several facets of a transaction between parties.

A decentralised dark pool proposal for cryptocurrency pairs is called Republic [5]. between many blockchains. An engine built on a multi-party computation protocol matches and trades financial assets and instruments through the dark pool, which offers a concealed order book.

A peer-to-peer internet currency called Dogethereum [6] allows for the exchange of Dogecoins for an equivalent amount of Ethereum tokens as well as the reverse. But only a small number of tokens can be exchanged for Ethereum on Republic and Dogethereum. However, in the scheme, we leverage smart contracts on Ethereum to validate other cryptocurrencies through a committee chosen among middle nodes, allowing the scheme to support exchanges of any cryptocurrency.

Even though blockchain-based crypto assets have been traded since 2009, there is a functional gap between on-chain transactions and centralised exchanges that rely on trust. This gap is filled by the decentralised exchange Uniswap[7]. Trading blockchain tokens is made possible by Uniswap's constant product automated market maker, which eliminates the need for market makers, bids, or asks. As any level of volume can be exchanged at any moment predictably, this challenges the regulation of traditional financial market structure and increases market completeness.

Using randomly chosen users as intermediates, transactions between any two types of cryptocurrencies can be achieved in single-user and multi-user scenarios in the decentralised cross-cryptocurrency exchange method [8] proposed by Hangyu Tian and Kaiping Xue. In the latter, different users can start numerous transfers at once, and the formed contract can instantly gather these payments and finish all of these transactions at once.

Chapter 3

Methodology

Centralized exchanges make trading more convenient for users, but there are certain security dangers as well. Once users store their assets on a centralised exchange platform, the platform becomes the system's "Achilles Heel" and opens the door to malicious exploitation of users' assets and transaction data. Additionally, there will always be a single point of failure because it is a central institution. Adopting a distributed coin exchange scheme is the greatest solution to this issue and is also consistent with the decentralisation of cryptocurrencies.

3.1 Proposed System

Using randomly chosen users as intermediates, a decentralised cross-cryptocurrency exchange method based on smart contracts allows for the realisation of single-user and multi-user transactions between any two types of cryptocurrencies. On an Ethereum test network, the exchange will be created and put into operation. A decentralised cryptocurrency exchange system that can validate various sorts of cryptocurrency transactions sent by various users is implemented using smart contracts based on Ethereum. The user can swap out various cryptocurrencies in his or her wallet via the exchange. Additionally, users might transmit cryptocurrency to another wallet. Users have the option of sending cryptocurrency to their wallet address or another person's wallet address. Full transaction history will be logged in Etherscan, and users may explore a collection of prior transactions directly from the interface.

3.2 System Model

There are four main components in the system model, which are described in detail as follows:

- Payer:- A payer is a user who wants to send crypto to another user.
- Payee:- A payee is a user who requires a transfer-in.
- Intermediary:- A user who is an intermediary is one who Both an Ethereum account and an account for a different cryptocurrency are sufficiently funded to be utilised for transactions. Through intermediaries, smart contracts link the payment and the payee.
- Blockchain:- Blockchain networks used by payer, payee and system.

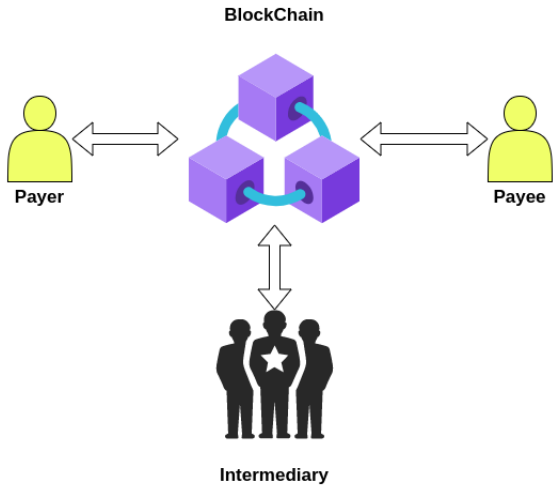

Figure 3.1: Components of System Model

3.3 Design Goals

The goal is to develop a decentralized secure transfer scheme between different types of cryptocurrencies. Specifically, design goals include the following:

- Decentralized and non-repudiation:- In the plan, there won't be a centralised third party. No participant in the agreement may dispute their actions. The blockchain will keep a permanent record of every action that a user takes on the smart contract.

- Portability:- Scheme need to facilitate transactions across several coin kinds in the cross-blockchain transactions system. In other words, the type of cryptocurrency has no bearing on the transaction process.

- Fairness:- The plan nevertheless ensures that the system is operating normally and won't jeopardise the interests of any loyal users, even if one or more of its steps fail.

- Stability:- Under reasonable security assumptions, the protocol must be able to survive typical assaults on blockchain networks.

- Atomicity:- If all parties adhere to the rules in the cross-blockchain transactions scheme, then all exchanges happen. Then, no party that follows the rules suffers a loss, and no coalition is motivated to depart from the rules.

- Safety and liveness:- For a consensus protocol, liveness denotes that the transaction will eventually commit or abort, and safety denotes that honest (non-Byzantine) nodes agree on the same value.

3.4 Smart contracts

Smart contracts are algorithms that run on blockchains when specific conditions are met. They are frequently used to automate the execution of an agreement so that all parties may be certain of the outcome immediately, without the need for a middleman or further delay. They can also automate a process so that it will only take action when certain conditions are met.

The terms of the contract between the buyer and seller are directly written into lines of code in a smart contract, making it a self-executing contract. A decentralised blockchain network contains the contracts and underlying code. The code controls how transactions are carried out, and they are both traceable and irrevocable. The concept of smart contracts was first proposed in 1994 by

American computer scientist Nick Szabo, who launched the virtual currency ”Bit Gold” in 1998, more than a decade before the launch of bitcoin.

In actuality, Szabo has disproved assertions that he is the real Satoshi Nakamoto, the guy who secretly invented bitcoin. Smart contracts are verified, carried out, and enforced by computer software running on a blockchain network. After all, parties have agreed to the terms of the smart contract, the programme will begin to run. A third party is no longer required since the blockchain network confirms and upholds the contract. Every transaction in a contract is processed by the blockchain network that underpins smart contracts, doing away with the need for intermediaries to complete the transactions.

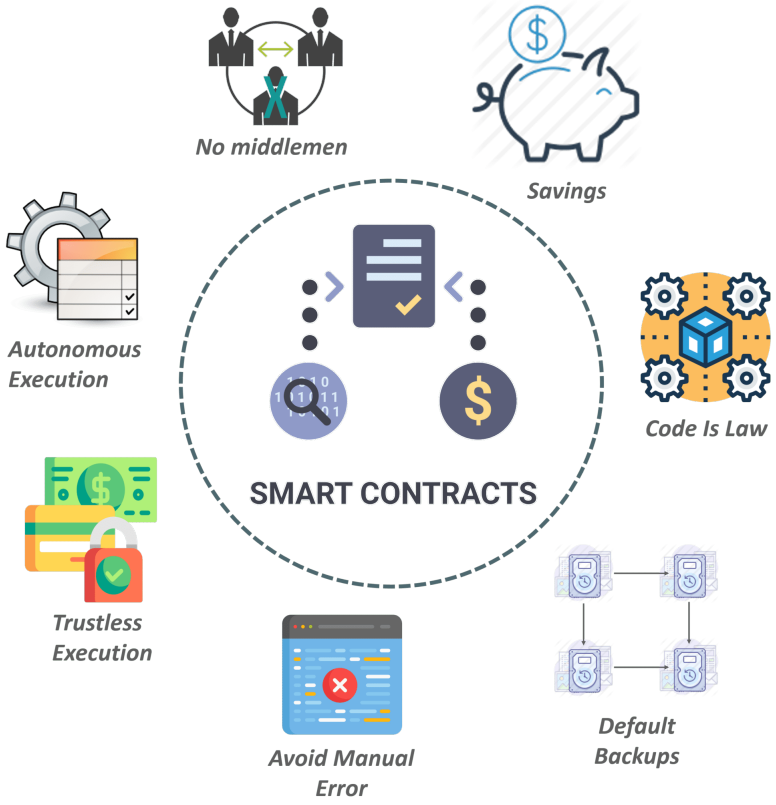

Figure 3.2: Features of Smart Contracts

A ”smart contract” is a programme that runs on the Ethereum blockchain. It is a collection of operations and state-related information that is kept on the Ethereum blockchain at a specific address. Smart contracts are one type of Ethereum account. They now have a balance and may

send transactions via the network. They are dispersed around the network and are programmed to perform a certain way; they are not, however, user-controlled. Then, by submitting transactions that perform a smart contract function, user accounts can connect with a smart contract. Smart contracts are similar to regular contracts in that they allow for the creation of rules and the automatic enforcement of those rules by the system. Interactions between smart contracts are always irreversible and cannot be reversed.

Smart contracts are applications that may be loaded and executed on a blockchain. The smart contract code may be transmitted by developers across the Ethereum network, and it is then verified by miners and put into the blockchain. Any smart contract code that has been stored on the blockchain can be called by a user after they have satisfied certain conditions. In this project, I utilise smart contracts built on Ethereum to establish a decentralised cross-cryptocurrency exchange system that can validate different cryptocurrency transactions made by different users.

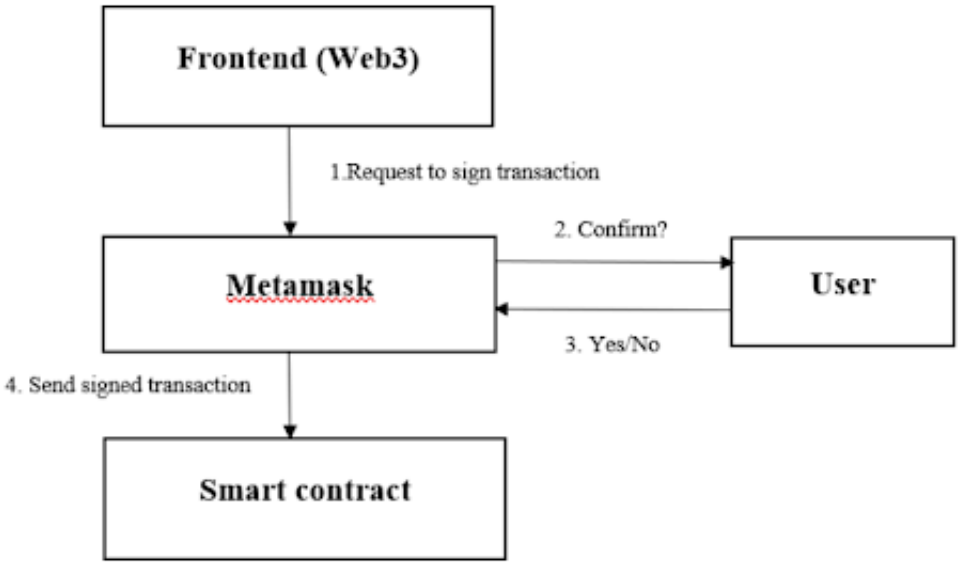

Figure 3.3: Frontend Interacting with Smart Contract

3.5 Modules and their Working

The entire project is divided into two main modules the client and the Ethereum Network:

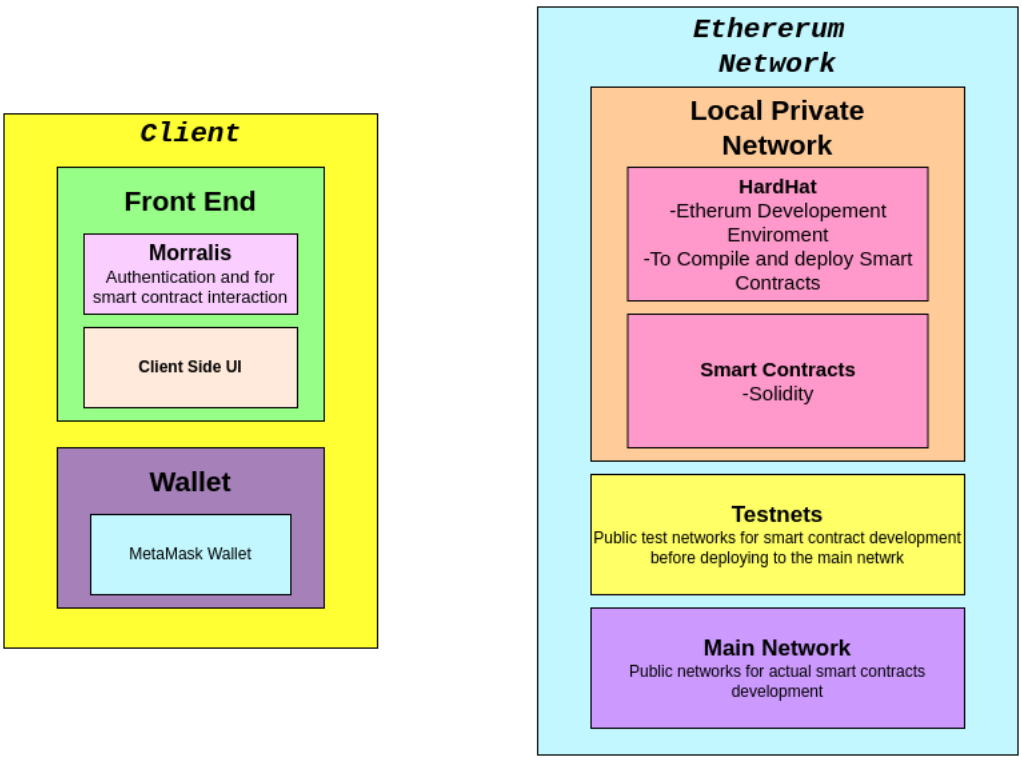

Figure 3.4: Modules of Decentralized Crypto Exchange

3.5.1 Client

The MetaMask and the exchange's user interface make up the client module. The frontend interface is created using the NextJS framework. The wallet for interfacing with smart contracts is MetaMask. The associated wallet itself is used to subtract the gas costs. You must click the platform's 'Connect to Wallet' button to utilise MetaMask to communicate with the exchange. When you click this, a screen asking if you want to allow the exchange to connect to your wallet should appear. Exchange connects to MetaMask automatically, streamlining the connecting process.

If payment is necessary within the exchange, a pop-up window requesting confirmation of the transaction from the MetaMask account will display. Moralis is used to interface with smart contracts as well as to authenticate every user that logs in. The full-stack approach for developing the exchange is provided by Moralis. The balances of your users are automatically synced with the exchange. The ultimate Web3 development platform from Moralis gives you access to MetaMask's full potential while acting as a portal into the Web3 universe.

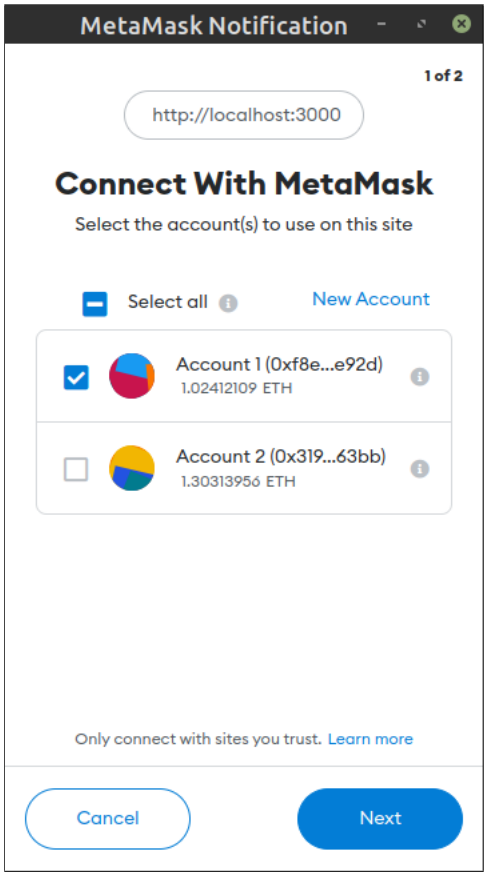

Figure 3.5: Exchange Connecting with metamask

3.5.2 Ethereum Network

Solidity-based smart contracts are first deployed via hardhat to the rinkeby Ethereum test network. A development environment called Hardhat aids in the compilation, deployment, testing, and debugging of Ethereum applications. It features some of the most organised, thorough documentation. When utilising Hardhat, the key component is the Hardhat Runner. It is a versatile and extendable task manager that aids in organising and automating the ongoing work necessary for creating smart contracts and dApps. By interacting with the smart contract using MetaMask and Moralis, the client module may validate various bitcoin transactions submitted by various individuals. The client was required to pay a certain amount of Ether as gas fees for each interaction with the smart contract.

By communicating with the smart contracts that have been set up on the test network, the ex-

Transaction	Gas	USD
Deploy	3690283	1.43
Register	181591	0.07
Update	80995	0.03
Verify	191737	0.07
Prepare	398525	0.15
Deposit	36452	0.01
Validation	163780	0.06

Table 3.1: Standard Gas Fees of each Operation with smart contracts

change gives the user access to the swap and send crypto features. Using the Metamask wallet, the exchange communicates with the smart contracts when exchanging one cryptocurrency for another. Coin A will be exchanged for Ether using smart contracts, and then this Ether will be exchanged for Coin B. The exchange will work with the smart contract to transmit a certain quantity of Ethereum to another account when sending cryptocurrency. The user must pay a specific amount of Ether as gas costs for each transaction that takes place in the exchange, and Etherscan will track every exchange transaction.

3.6 Deploying Smart Contracts

Truffle and Hardhat, two Ethereum development environments, make it simpler to deal with smart contracts and Ethereum nodes. They offer a collection of tools that make it simple to create, test, and deploy smart contracts. A development environment called Hardhat aids in the compilation, deployment, testing, and debugging of Ethereum applications. It features some of the most organised, thorough documentation. For debugging reasons, Hardhat also has console.log() capabilities, comparable to javascript.

Steps for deploying smart contracts using Hardhat

- Step 1. Set up the environment and install all the dependencies for Hardhat
- Step 2. Initiate a hardhat project using npx hardhat command

- Step 3. Configure network and private key
- Step 4. Code for Deployment
- Step5. Deploy using npx hardhat run scripts/deploy.js –network ¡network-name¿

```
coldfire@sahil:~/QuikNode/hardhat_demo$ npx hardhat run scripts/deploy.js --network ropsten
Deploying contracts with the account: 0x6E0d01A76C3Cf4288372a29124A26D4353EE51BE
Account balance: 6940453960155364968
Contract deployed at: 0x3B65a9AbD0c561EaB30F556428e966b4449DA0bE
```

Figure 3.6: Deploying smart contracts using hardhat

3.7 Tools and Software Requirements

3.7.1 Next.js

The Next.js framework is React-based and has server-side rendering capabilities. Both it and SEO are pretty quick. You can simply build and test sophisticated React-based applications with the help of Next.js.Next.js offers the best developer experience since it includes all the production-level features needed, such as TypeScript support, smart bundling, and route pre-fetching, and hybrid static server rendering. With the help of the adaptable Next.js React framework, you may construct quick online apps.

By framework, it's mean Next.js, which manages the React-specific tools and setup requirements as well as adds extra organisation, functionality, and optimizations to your application. Using Next.js features to solve common application requirements like routing, data fetching, and integrations after using React to build your user interface can improve the experience for both developers and end users. React may be used without Next.js, however, Next.js leverages React to deploy apps, extending React's functionality and expediting the development process. To create a project, run:

Create React App (CRA), a programme used to set up React applications that contain tools like eslint, is a framework specifically designed for React. A React-based framework called Next.js

```
npx create-next-app@latest
# or
yarn create next-app
# or
pnpm create next-app
```

Figure 3.7: Creating a Next.js Project

creates server-side rendered applications. The architecture of the app is determined by Next.js, although React is still the project's underlying technology. The use of a folder system in Next.js also enables hassle-free complicated structures, making the process more user-friendly and straightforward. On the other hand, CRA automatically creates single-page applications if routing is not an issue. The primary distinction between CRA and NextJS is that whereas CRA runs in the client's browser, NextJS runs on the server, necessitating considerable code modifications.

3.7.2 Moralis

The full-stack methodology for creating high-performance apps is offered by Moralis. fully functional with your preferred web3 tools and services. automatically updating the database with users' balances, giving you the ability to create indexes, set up on-chain notifications, and so much more. An intuitive SDK is used to access all functionality. By default, every functionality that Moralis offers is cross-chain.

Moralis Setup:

- Step 1. Log In to your Moralis Account
- Step 2. Create a New Server with a suitable network.
- Step 3. Access Server Details to get Server URL and Application ID.
- Step 4. Initialize Moralis
  Fill up the ".env" file with the Moralis server information now to get the necessary backend capabilities provided by Moralis (application ID and server URL).

Figure 3.8: Initializing moralis in the app

The quickest method to create and launch dApps on Ethereum, BSC, Polygon, Solana, and Elrond is via Moralis. By default, all Moralis dApps are cross-chain. By using Moralis as a foundation, your dApp will be future-proof. Your dApp will immediately function on any chain, even if additional blockchains are created. Any dApp made with Moralis is built on a Moralis Server. Together with the Moralis SDK, it makes it simple to create dApps that include user authentication and blockchain data like user token balances, NFTs, transactions, and events.

3.7.3 Etherscan

The most popular Ethereum Blockchain block explorer is called Etherscan. Users may quickly look up, confirm, and authenticate transactions that have occurred on the Ethereum Blockchain using a block explorer, which is a search engine. It is independently run and being developed by a group of people that are enthusiastic and passionate about the types of decentralised information and infrastructure applications that Ethereum enables.

The Ethereum network's transactions are not under the control of Etherscan, which also doesn't hold your private keys or provide wallet services. Etherscan.io scans the public ledger that exists on the Ethereum Blockchain (which functions like a decentralised database) and then makes this data accessible on this website. By indexing and making all transactions on the Ethereum Blockchain searchable most openly and transparently possible, Etherscan aims to promote blockchain transparency.

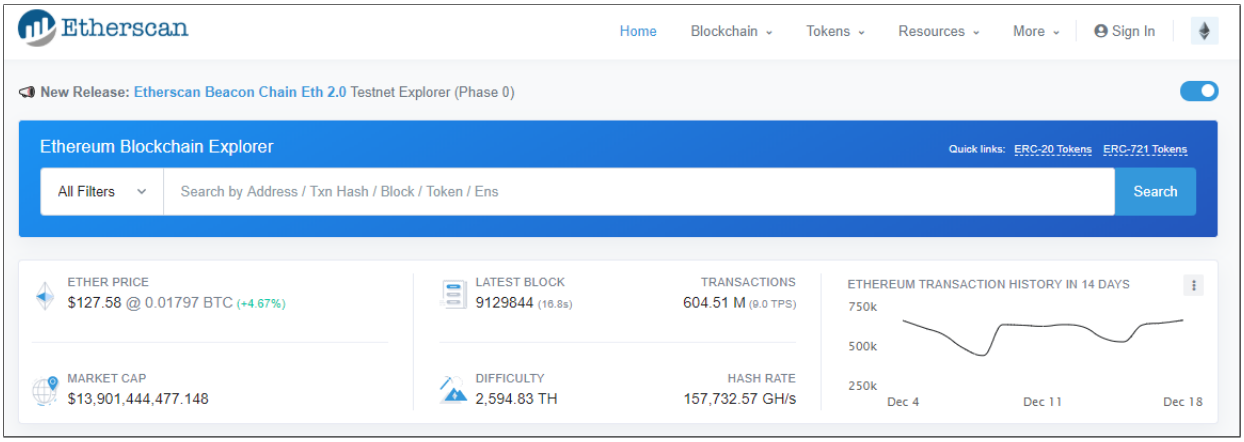

Figure 3.9: Etherscan Interface

3.7.4 Rinkeby

Before being implemented on the Ethereum main net, blockchain testing is mostly done on the Rinkeby testnet. It is an Ethereum main net split. The Geth, Besu, Nethermind, and OpenEthereum node clients are presently supported by the Rinkeby testnet. The money has no value because it is a testnet. Only requests may be made for ETH on the Rinkeby test net. Other nodes cannot receive mining rewards since authorised nodes may only add new blocks. Use the URL rinkebyfaucet.com to obtain testnet Ether on testnet Rinkeby. Only the authorised faucet offers testnet ether. On the network at this time, there are around 11,000,000 blocks, and as of 2021, Rinkeby has over 50 million transactions.

The Rinkeby testnet has 46 active nodes, and each block takes roughly 15 seconds to complete. A block can include a maximum of around 41,000 transactions. Because Rinkeby is more centralised than PoW testnets like Ropsten, which are open to spamming, many developers prefer it. Comparing PoW to PoA, security is improved overall. Additionally, developers like Rinkeby's testnet over others because of its quicker block time. The block period for Ropsten is around 30 seconds, however, Rinkeby reduces it by half. The chain data size for Rinkeby is only about 6GB. That means if you wanted to run an Ethereum node for Rinkeby, it wouldn't require a large amount of data size compared to other testnets.

3.7.5 Tailwind CSS

Tailwind CSS is essentially a utility-first CSS framework for fast producing distinctive user experiences. It is a low-level CSS framework that is very versatile and gives you all the building blocks necessary to develop unique designs without making you fight against annoying, opinionated styles. The nicest thing about tailwind is that it doesn't impose design standards or specify how your website should look; rather, you just mix little elements to produce a unique user experience.

Simply said, Tailwind displays the results after processing a "raw" CSS file through a configuration file. Even though there are several CSS frameworks, people often choose the one that is easiest to understand and apply to a project. Tailwind's built-in features and styles are available for users to choose from, and they may be used to reduce the amount of CSS code necessary to create visually attractive and distinctive user interfaces. It will help you finish the difficult project. It is straightforward to include pre-existing classes into the HTML code thanks to Tailwind CSS's little tools, which were created with a specified set of settings.

Advantages of Tailwind CSS:

- No more silly names for CSS classes and Ids.
- Minimum number of lines in a CSS file.
- We can customize the designs to make the components.
- Makes the website responsive.
- For the first time, a CSS framework is available that includes classes like flex, pt-4, text-center, and rotate-90 that can be combined to create any design directly in your markup.

3.7.6 Hardhat

The Ethereum software development environment is called Hardhat. It is made up of many parts that may be used to edit, compile, debug, and deploy your dApps and smart contracts, together forming a whole development environment. When utilising Hardhat, the key element you deal with is Hardhat Runner. It is a versatile and extendable task runner that assists you in organising

and automating the ongoing processes necessary for creating smart contracts and dApps. The ideas of tasks and plugins are central to the design of Hardhat Runner. Every time you launch Hardhat from the command line, a task is executed. For instance, the built-in compile job is executed by npx hardhat compile. Because tasks may contact one another, sophisticated workflows can be defined. Existing tasks may be replaced by users and plugins, allowing users to modify and expand processes. In your project, hardhats are utilised through a local installation. You may reproduce your environment in this manner and prevent further version problems.

3.7.7 JavaScript

The lightweight object-oriented programming language JavaScript (js) is used by many websites to script webpages. It is an interpreted, full programming language that allows for dynamic interactivity on websites when combined with HTML content. It was initially made available in 1995 so Netscape Navigator users could add programmes to webpages. Since then, it has been accepted by every other graphical web browser.

JavaScript enables users to build interactive modern web apps without having to keep refreshing the page. Js is used by the typical website to provide varying levels of simplicity and engagement. However, JavaScript is unrelated to the Java programming language. The name was suggested and made available during a time when Java was rising in popularity.

Along with web browsers, databases like CouchDB and MongoDB use JavaScript as their query and scripting language. Instead of being compiled, JavaScript is a translated language. The JavaScript Translator, which is integrated inside the browser, must translate the JavaScript code for the web browser.

3.7.8 MetaMask

The MetaMask software cryptocurrency wallet is used to interact with the Ethereum network. To connect with decentralised applications, users can access their Ethereum wallet using a mobile app or browser extension. MetaMask was created by ConsenSys Software Inc., a blockchain software business that specialises on Ethereum-based infrastructure and tools. With MetaMask, users may safely connect to decentralised apps using a compatible web browser or the built-in browser

of the mobile app, send and receive Ethereum-based money and tokens, broadcast transactions, save and manage account keys, and send and receive account keys. Developers link Metamask to their decentralised apps by specifying interactions between Metamask and Smart Contracts using a JavaScript plugin like Web3js or Ether.

To get the cheapest exchange rate, the Metamask application aggregates many decentralised exchanges (DEXs) to provide an integrated solution for trading Ethereum tokens. The service cost for this feature, known as MetaMask Swaps, is equal to 0.875 percent of the transaction value. One of the most well-liked hot cryptocurrency wallets available right now is MetaMask. Additionally, MetaMask is an open-source wallet (software that anybody can use). It operates and supports all ERC-20 tokens that are compatible with the Ethereum Virtual Machine (EVM) and function on the relevant networks.

If you're using a phone, MetaMask functions as an application, and if you're using a web browser, it functions as an extension. The currently supported browsers of MetaMask are Chrome, Firefox, and Brave. You may communicate with DApps in decentralised finance (Defi), like PoolTogether, using MetaMask in addition to storing your cryptocurrency. MetaMask must inject a Web3 JavaScript object onto each page to allow dapps (decentralised applications) to access the blockchain. By doing this, only enables the website to access the network without altering it.

Connecting to several DApps is one of MetaMask's major advantages. You must visit the website of the desired DApp in order to achieve this. Once there, look for a button that lets you link MetaMask to that DApp, such as "connect wallet." The only thing left to do is confirm the connection in the next pop-up box.

Simply connecting your wallet address to the programme is what this is. On the DApp, money could be needed whenever you want a service to be performed. They do not, however, demand money from you immediately; instead, they first ask you whether you are willing to pay the requisite sum. These are the straightforward steps you must follow to download MetaMask, use it to buy, sell, or trade tokens, and link it to DApps.

Advantages of using MetaMask:

- MetaMask has gained a lot of attention in recent years. Not only has this made the software safer and better, but it has incentivized other DApps to support MetaMask. Furthermore, MetaMask has built up a loyal fanbase among customers, which has motivated the MetaMask

team to develop their software even further.

- A big advantage of MetaMask is its easy access. Everyone can create a new wallet or import their existing wallets through MetaMask. All you need is a trusted web browser, access to a computer, and a safe internet environment. Then, simply download MetaMask and create your wallet through some simple clicks.

- Another huge advantage that MetaMask has as a crypto wallet is the fact that it is open-source software. Users of MetaMask can actively engage and access updates and information regarding the software, ensuring a better and safer environment for everyone. Note that not all crypto wallets are open-source.

3.7.9 Solidity

Solidity is a curly-braced, statically-typed programming language created specifically for creating smart contracts that work with Ethereum. The second-largest cryptocurrency market by market capitalization, Ethereum, led by Christian Reitwiessner, debuted Solidity in 2015, a brand-new programming language.

Some key features of solidity are listed below:

- Solidity is a high-level programming language designed for implementing smart contracts.

- It is statically-typed object-oriented(contract-oriented) language.

- Solidity is highly influenced by Python, c++, and JavaScript which runs on the Ethereum Virtual Machine(EVM).

- Solidity supports complex user-defined programming, libraries and inheritance.

- Solidity is the primary language for blockchain-running platforms.

- Solidity can be used to create contracts like voting, blind auctions, crowdfunding, multi-signature wallets, etc.

Any network participant—basically, anyone in the globe who wants to—can do business with people they don't know or trust by using Solidity smart contracts. They might not even exchange

money. The rules of a commercial transaction are specified programmatically in a machine-readable language via Solidity smart contracts. Such unheard-of decentralised business is automated and capable of functioning around the clock, wherever in the world, without human or third-party oversight.

Solidity had to be the most straightforward and user-friendly smart contract language since it was created to make Ethereum more accessible to new users. Solidity smart contracts initially had characteristics that mirrored these aspirations, such as a syntax that is very similar to JavaScript and a notable dearth of functionality. The language has some obvious trade-offs to make it more approachable. Instructions from Solidity smart contracts are translated into bytecode for the EVM. As previously established, the EVM instances that operate on the Ethereum network's nodes enable them to concur on the execution of a certain set of instructions.

3.7.10 TypeScript

Strongly typed programming language TypeScript, which is based on JavaScript, gives you better tools at any size. To facilitate a stronger interaction with your editor, TypeScript extends JavaScript's syntax. Prevent mistakes in your editor. Code written in TypeScript may be converted to JavaScript and executed everywhere JavaScript is supported, including browsers, Node.js, Deno, and your applications. JavaScript is understood by TypeScript, and type inference is used to provide excellent tools without the need for extra code. JavaScript may be written in whatever style you choose thanks to TypeScript. A typed superset of JavaScript, TypeScript compiles to standard JavaScript. With classes, interfaces, and statically typed code like C or Java, TypeScript is purely object-oriented. TypeScript is used to create Angular 2.0, a well-known JavaScript framework.

Microsoft created and maintains TypeScript, an open-source, object-oriented programming language, under the terms of the Apache 2 licence. Anders Hejlsberg, a crucial member of the C language development team, introduced it. Strongly typed superset of JavaScript, TypeScript compiles to standard JavaScript. It is a language for developing JavaScript at the application level, and it can run on any browser, any host, and any operating system. The browser does not directly run TypeScript. To compile and create a JavaScript file, a compiler is required. The ES6 version of JavaScript with some added capabilities is TypeScript.

An open-source, entirely object-oriented programming language is TypeScript. It compiles to plain JavaScript and is a strongly typed superset of JavaScript. It includes every component of JavaScript. It is a language created for the creation of massive JavaScript applications that can run on any browser, any host, and any operating system. Both a language and a collection of tools exist in TypeScript. The ES6 version of JavaScript with some added capabilities is TypeScript.

An open-source, entirely object-oriented programming language is TypeScript. It compiles to plain JavaScript and is a strongly typed superset of JavaScript. It includes every component of JavaScript. It is a language created for the creation of massive JavaScript applications that can run on any browser, any host, and any operating system. Both a language and a collection of tools exist in TypeScript. The ES6 version of JavaScript with some added capabilities is TypeScript.

3.7.11 Web3.js

Using HTTP, IPC, or WebSocket, you may communicate with a nearby or distant Ethereum node using the technologies that makeup Web3.js. You may create websites or clients that communicate with the blockchain using Web3.js. For instance, it enables you to read and write data from smart contracts, transmit ether from one account to another, build smart contracts, and much more. The Ethereum Foundation created and supports Web3.js. Web3.js hence has greater support because more developers are using it. Web3.js presumes that the application is linked to a local node.

It is anticipated that the node reads and engages with the Ethereum blockchain, signs transactions, and saves keys. In practice, most users do not run geth locally, therefore this is not frequently the case. The majority of web3 apps need Metamask to retain keys, sign transactions, and communicate with the Ethereum mainnet since it successfully simulates that environment through a browser application.

Checking account balances, making transactions, and even deploying smart contracts can all be done in the frontend and backend using Web3.js. Similar to Web3.js, Ethers.js contains several modules. This JS library specifically has four modules named ethers. contract, ethers.provider, ethers. utils, and ethers. wallets. Web3.js hence has greater support because more developers are using it. Web3.js presumes that the application is linked to a local node. The Ethers.js API's main building blocks are these modules (Application Programming Interface). The majority of

web3 apps need Metamask to retain keys, sign transactions, and communicate with the Ethereum mainnet since it successfully simulates that environment through a browser application.

Criteria	Web3.js vs Ethers.js
Maintenance	Regular updates for both libraries
Performance	Ethers.js give slightly more performance due to the small size
License	Web3.js has an LGLv3 license, while Ethers.js has an MIT license.
The documentation Quality	Documentation of the two ETH JavaScript libraries is not perfect.

Table 3.2: Comparison between Web3.js vs Ethers.js

Chapter 4

Results and Discussions

The decentralized cryptocurrency exchange was implemented using smart contracts, which can verify different types of cryptocurrency transactions sent by different users. The exchange allows the user to swap different cryptocurrencies in his/her wallet. It also allowed users to send crypto to another wallet. Users can send crypto to/her another wallet address or to a wallet address of another person. Users can view a set of previous transactions from the interface itself and a detailed transaction history will be registered in Etherscan.

4.1 Exchange interacting with Metamask Wallet

To use MetaMask to interact with the exchange, you'll need to use the 'Login' button on the platform. After clicking this, you should then see a prompt asking whether you want to let the exchange connect to your wallet. Exchange automatically connects to MetaMask, after successful validation by MetaMask simplifying the connection process. Once the exchange is connected to the metamask wallet, Exchange can get access to the account information of the wallet and can fetch balance and different cryptocurrencies available in the wallet to the exchange using Moralis.

4.2 Swapping Crypto Currencies

In order to swap cryptocurrencies, the user must first select pair(Coin A Coin B) of cryptocurrencies that he/she want to swap. After that, the user must specify, how much amount of Coin

A should be swapped to Coin B. After confirming the amount the user can press the swap crypto button, which will initiate the swapping operation. First, Coin A will be converted to x amount of Eth by interacting with the smart contract and the transaction will be published. After that x amount of eth will be converted to Coin B and the transaction will be published, thereby swapping Coin A to Coin B. If the first coin is Eth, then it will be directly converted to coin B and vice-versa. The user has to pay some amount of eth for each smart contract interaction.

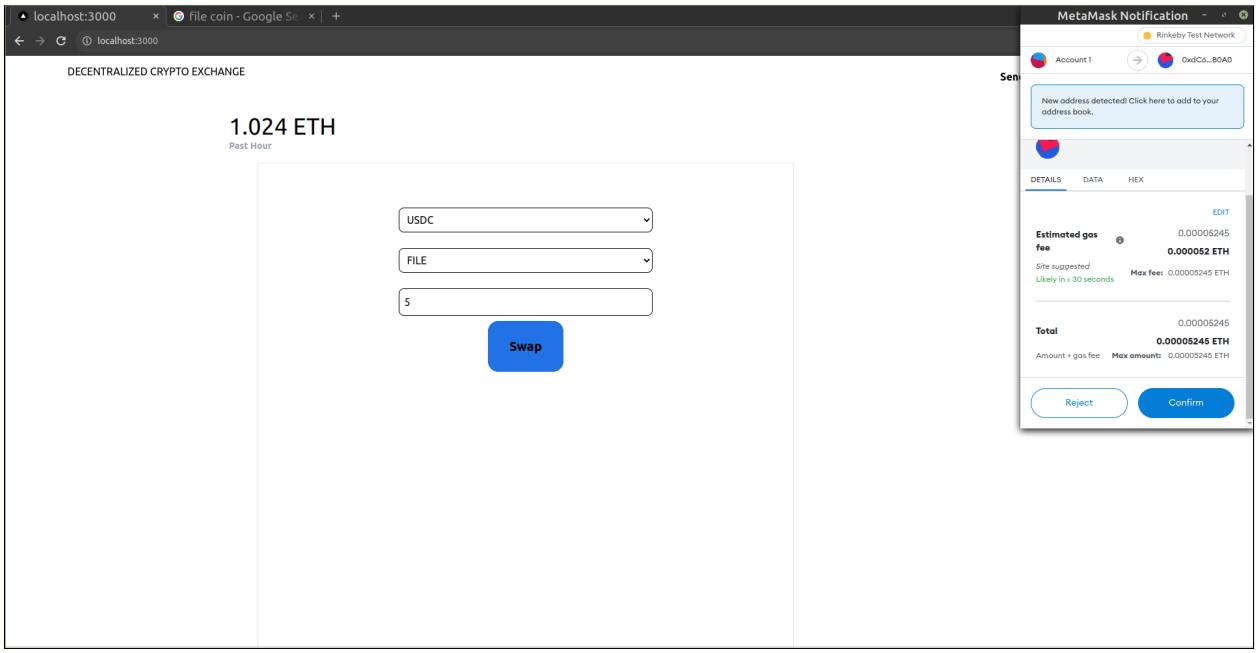

Figure 4.1: Swapping Crypto Confirmation

4.3 Sending Crypto Currencies

While sending the crypto currencies, the user must first enter how much amount of Eth should be sent to another wallet. He/she must also specify the address of the wallet that he wants to send the eth. After initiating the send crypto operations, a particular amount of eth specified by the user will be sent to the wallet address, specified by the user by interacting with the smart contract.

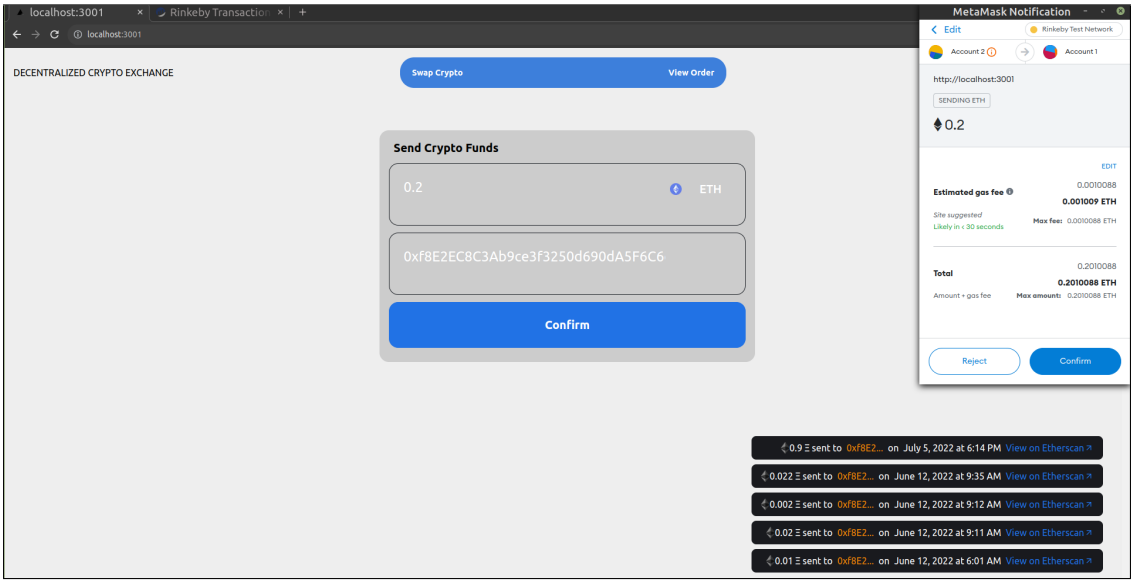

Figure 4.2: Sending Crypto Confirmation

4.4 Gas fees for each transaction

The quantity of ether (ETH), the native coin of Ethereum, that the network needs for a user to connect with the network, is referred to as ”gas.” By making it prohibitively expensive for malevolent users to spam the network, these fees add an extra layer of security to the Ethereum network and pay Ethereum miners for the energy expended to validate transactions. Gas fees are everyone’s least favourite aspect of Ethereum, despite the fact that they are an efficient way to motivate miners to continue confirming transactions and upholding network security.

On a blockchain technology, gas fees are the transaction costs that users pay to miners to get their transaction added to the block. The system operates using the typical supply and demands dynamic. Forcing users to pay more to have their transactions processed swiftly and effectively, miners may decide to include the transactions that pay out more if there is a greater demand for transactions. As seen in the graphic below, Ethereum users can also decide to pay extra for transactions that happen more quickly. Users may engage directly with the Ethereum network and choose how much gas to pay by using wallets like MetaMask.

Users must pay a certain amount of Ether as gas costs for each transaction they make with the exchange, and Etherscan will track all of the exchange’s transactions. If payment is neces-

sary within the exchange, a pop-up window requesting confirmation of the transaction from the MetaMask account will display.

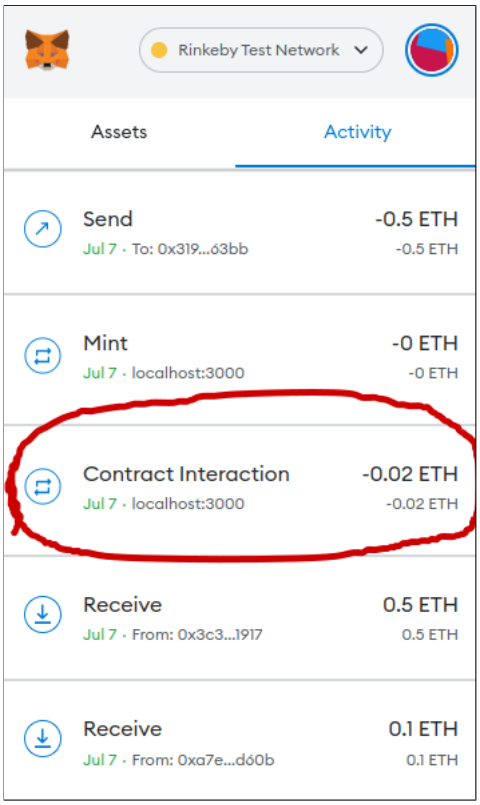

Figure 4.3: Sending Crypto Confirmation

Chapter 5

Conclusion

The decentralized cryptocurrency exchange was implemented using smart contracts based on Ethereum, which verifies different types of cryptocurrency transactions sent by different users. The exchange allows the user to swap different cryptocurrencies in his/her wallet. It also allowed users to send crypto to another wallet. Users can send crypto to his/her wallet address or a wallet address of another person. Users can view a set of previous transactions from the interface itself and a detailed transaction history will be registered in Etherscan.

5.1 Advantages

- Transactions between any two types of cryptocurrencies may be completed in single-user and multi-user situations using decentralised cross-cryptocurrency exchange methods based on smart contracts that can utilise randomly chosen users as intermediates.
- The Rinkeby Ethereum test network is used for the implementation and deployment of exchange.
- Various transactions sent by different users can be verified using smart contracts built on the Ethereum platform.
- Exchange can avoid a variety of drawbacks related to a centralised cryptocurrency exchange, such as the single point of failure, concerns related to identity theft, expensive exchange costs, etc.

Chapter 6

Future Enchancements

Future upgrades to the exchange should include the ability to purchase cryptocurrencies and let customers use their bank accounts to pay for crypto assets. A different Ethereum test network than Rinkeby should be used for the exchange because Rinkeby will be deprecated. Goerli will be the most popular test network as of right now. Exchange should be made to support more cryptocurrencies, including the recently established one, thereby allowing users to perform different types of transactions using any cryptocurrencies. The exchange should try to be put on the Ethereum Mainnet, and the exchange's cost should be kept an eye on. The exchange should be tested to be installed on Ethereum 2.0 once it is released to compare the cost of exchange between the two Ethereum versions.

References

[1] Buterin, Vitalik. “A NEXT GENERATION SMART CONTRACT DECENTRALIZED APPLICATION PLATFORM.” (2015). Journal of Information Security, Vol.10 No.1

[2] ”DOCUMENT SECURITY AND STORAGE ON BLOCKCHAIN”, International Journal of Emerging Technologies and Innovative Research (www.jetir.org), ISSN:2349-5162, Vol.5, Issue 4, page no.1037-1039, April-2018, Available :http://www.jetir.org/papers/JETIR1804208.pdf

[3] Dannen, Chris. (2017). Introducing Ethereum and Solidity. 10.1007/978-1-4842-2535-6.

[4] F. Vogelsteller and V. Buterin. (2015). EIP 20: ERC-20 Token Standard. Available: https://eips.ethereum.org/EIPS/eip-20

[5] T. Zhang and L. Wang. (2017). Republic Protocol. Accessed: Apr. 2021. [Online]. Available: https://republicprotocol.github.io/whitepaper/ republic-whitepaper.pdf

[6] J. Teutsch, M. Straka, and D. Boneh, “Retrofitting a two-way peg between blockchains,” 2019, arXiv:1908.03999. [Online]. Available: http://arxiv.org/abs/1908.03999

[7] Lo, Yuen and Medda, Francesca, Uniswap and the Emergence of the Decentralized Exchange (October 20, 2020). Available at SSRN: https://ssrn.com/abstract=3715398 or http://dx.doi.org/10.2139/ssrn.3715398

[8] H. Tian et al., ”Enabling Cross-Chain Transactions: A Decentralized Cryptocurrency Exchange Protocol,” in IEEE Transactions on Information Forensics and Security, vol. 16, pp. 3928-3941, 2021, DOI: 10.1109/TIFS.2021.3096124.

APPENDIX

Screenshots

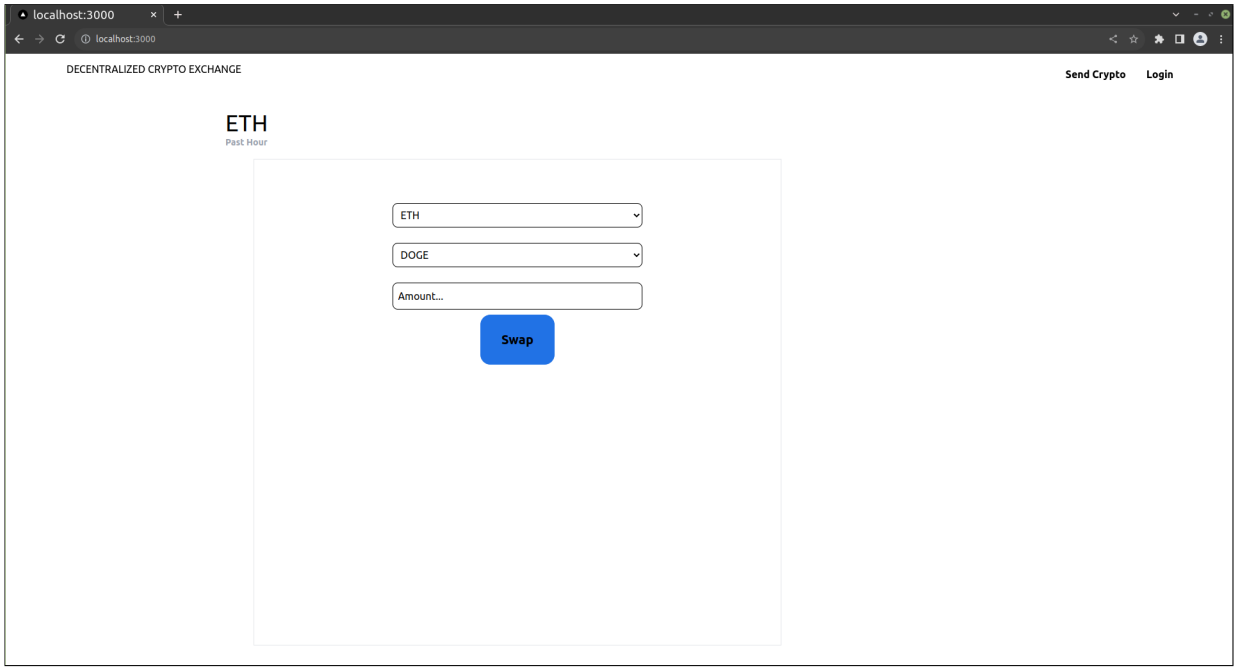

Figure A.1 : Decentralized Crypto Exchange Interface

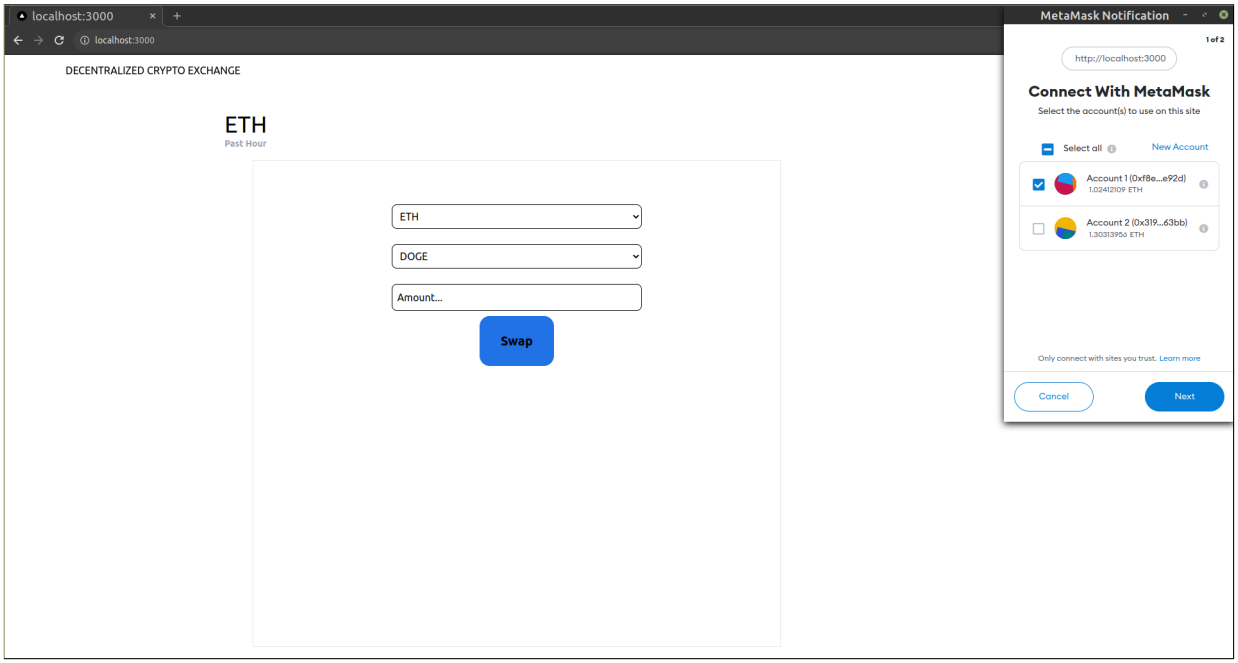

Figure A.2 : Connecting to a MetaMask Account

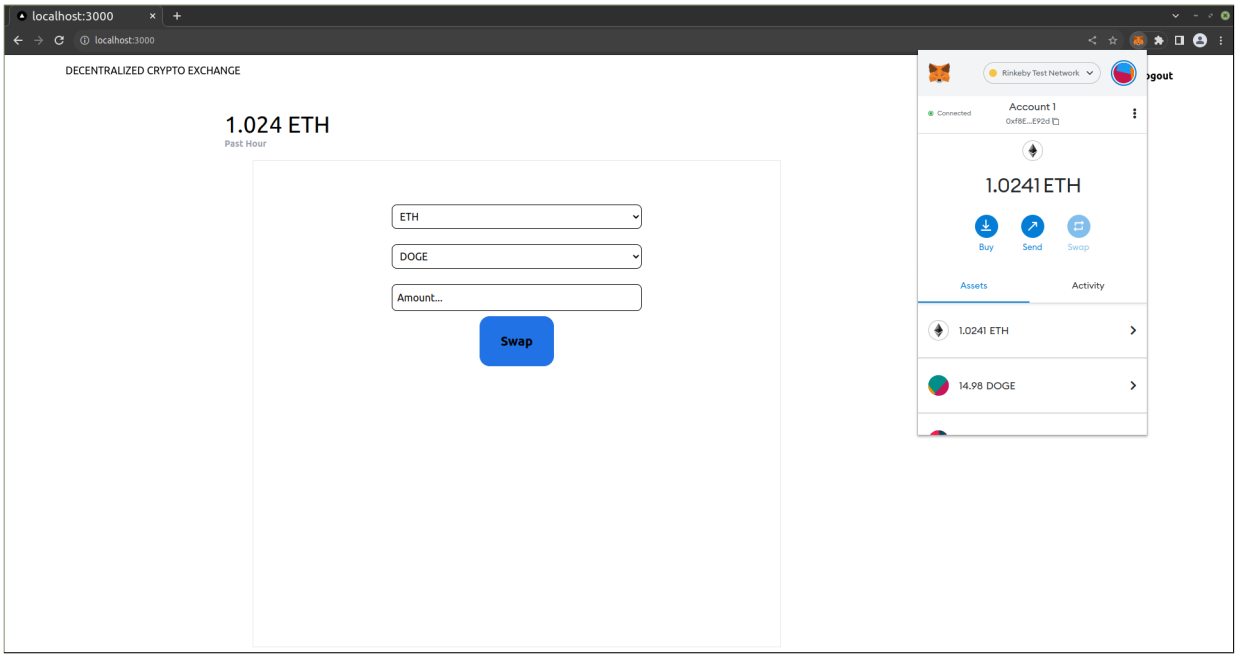

Figure A.3 : Connected Account Info

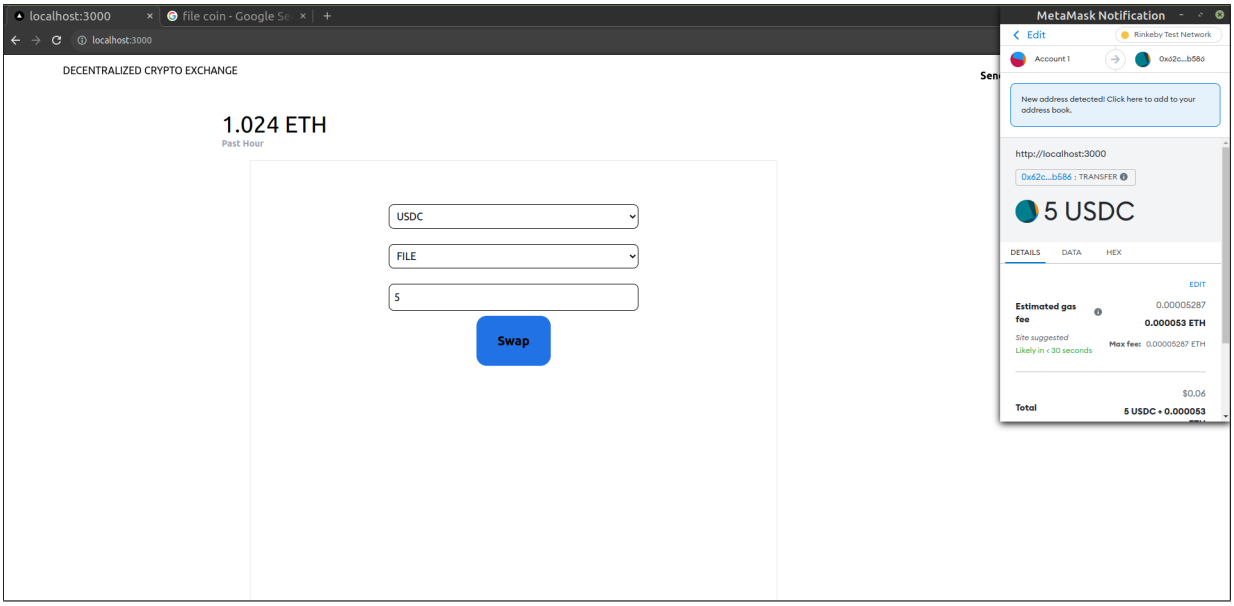

Figure A.4 : Swapping Two Crypto Currencies

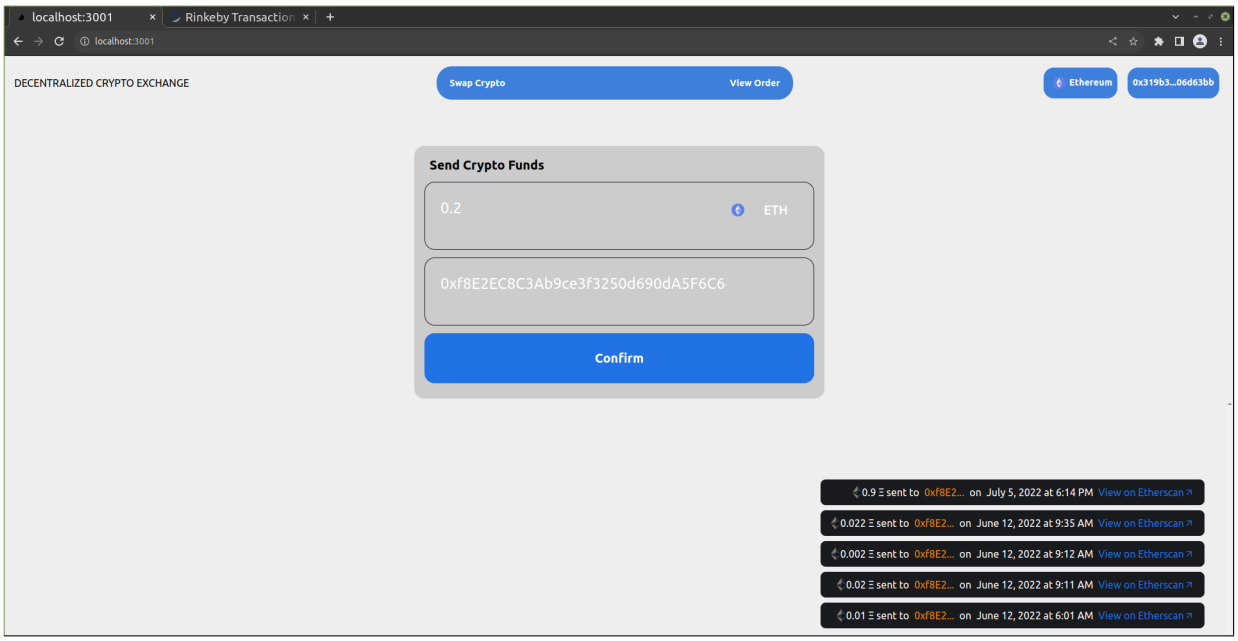

Figure A.5 : Sending Crypto to an account

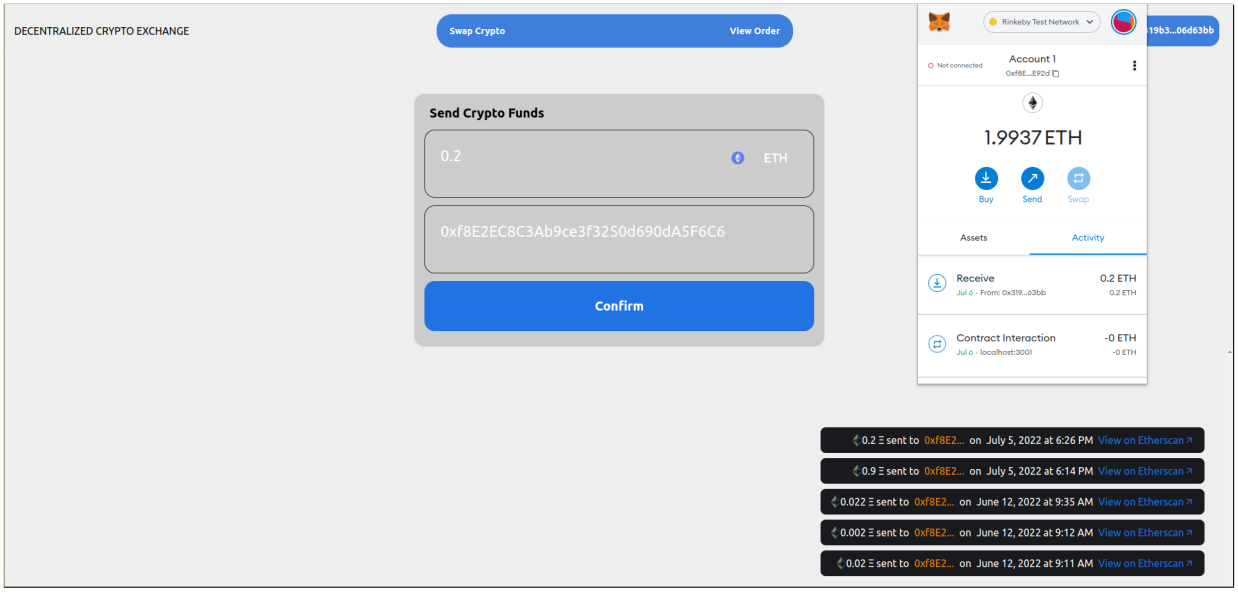

Figure A.6 : After Successful Transaction