1. INTRODUCTION

1.1 EVOLUTION OF WEB:

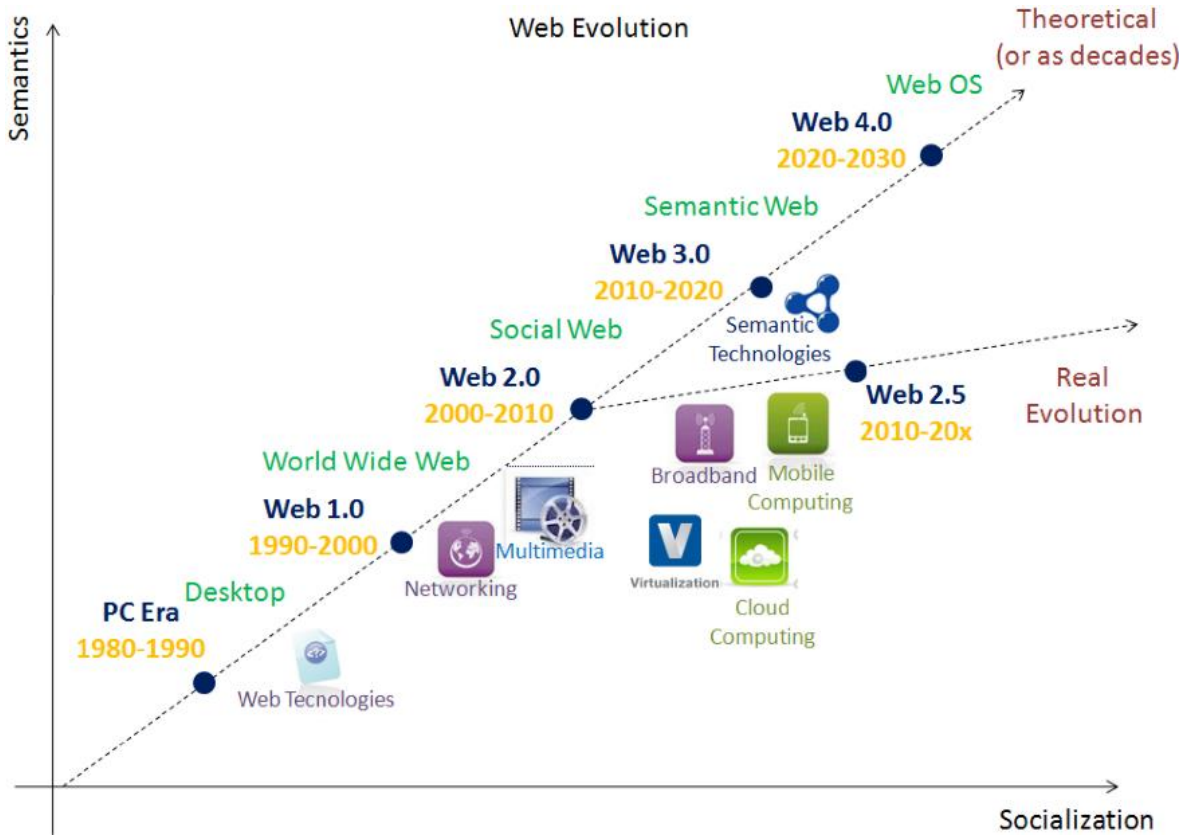

Fig 1.1: Web evolution diagram

The first version of web Web 1.0 also referred as Syntactic web or read only web is the era(1990–2000) where the role of a user is limited to reading information provided by the content producers.

Fig 1.2 : Internet explorer

There is no option given for user or consumer to communicate back the information to the content producers. Example of Web 1.0 are static web sites and personal sites.

The Web 2.0 also referred as Social Web or read-write web is the era(2000–2010 and continues even now) which facilitates interaction between web users and sites which intern allows users to communicate with other users. In this era every user can be a content producers and content is distributed and shared between sites. Some of the famous Web 2.0 applications are Facebook, Youtube, Flickr, Twitter etc.,

Fig 1.3: Twitter

The web technologies like HTML5, CSS3 and Javascript frameworks like ReactJs, AngularJs, VueJs etc.,

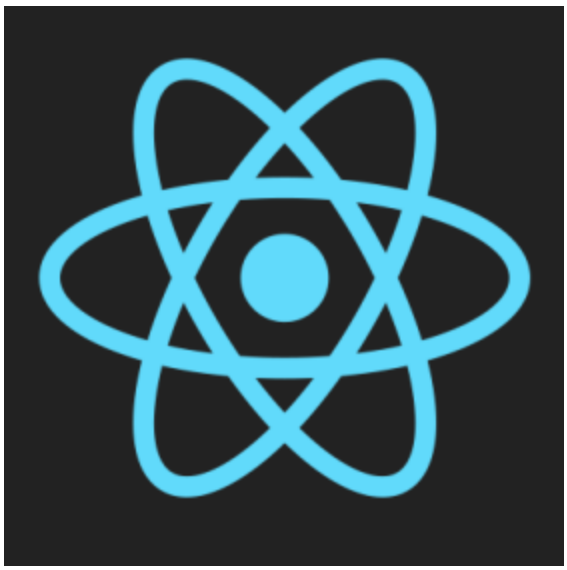

Fig 1.4: React JS

enables startups to innovate new ideas which enables users to contribute more in this Social Web. Web 2.0 is build around the users, producer just need build a way to enable and engage them.

The Web 2.5 also referred as Semantic Web or read-write-execute is the era(2010 and above) which refers to the future of web. In this era computers can interpret information like humans via Artificial Intelligence and Machine Learning. Which help to intelligently generate and distribute useful content tailored to a particular need of a user.

What if computers can understand meaning behind information

What if they can learn “what we are interested in”

Then they can help us find what we want

It can recognise People, Place, Events, Companies, Product, Movies etc.,

It can understand the relationship between things

Some of the examples of web 2.5 are Apple’s Siri, Googles Cloud API, Wolfram Alpha.

Fig 1.5: Apple siri

Web3 (also known as Web 3.0) is an idea for a new iteration of the World Wide Web which incorporates concepts such as decentralization, blockchain technologies, and token-based economics. Some technologists and journalists have contrasted it with Web 2.0, wherein they say data and content are centralized in a small group of companies sometimes

referred to as "Big Tech". The term "Web3" was coined in 2014 by Ethereum co-founder Gavin Wood, and the idea gained interest in 2021 from cryptocurrency enthusiasts, large technology companies, and venture capital firms.

Fig 1.6: Metamask

1.2 BLOCKCHAIN:

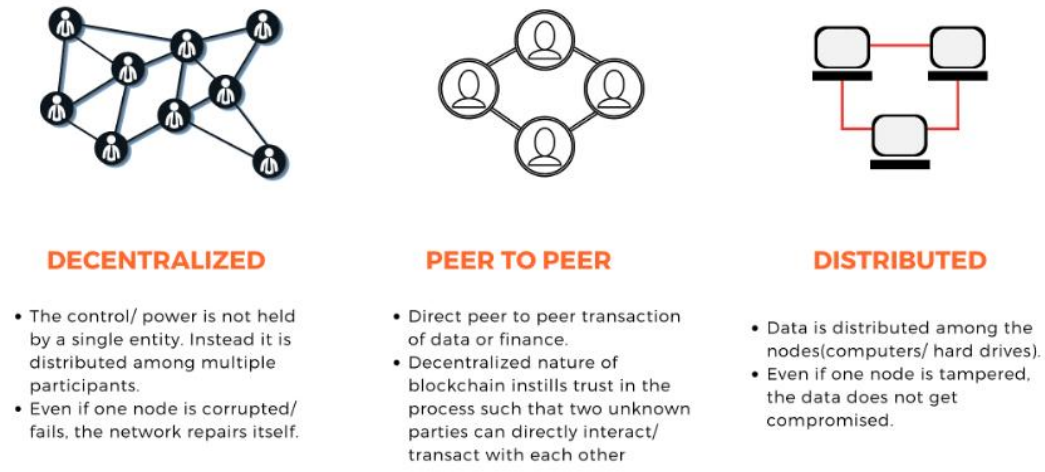

Fig 1.7: Properties of blockchain

A blockchain is a type of distributed ledger technology (DLT) that consists of growing list of records, called blocks, that are securely linked together using cryptography. Some of the popular applications of Blockchain are Decentralized Exchanges (DEX) and Decentralized Finance (DeFi).

Fig 1.8: Vitalik Buterin- founder of Ethereum

In a few words, a blockchain is a digital ever-growing list of data records. Such a list is comprised of many blocks of data, which are organized in chronological order and are linked and secured by cryptographic proofs.

The first prototype of a blockchain is dated back to the early 1990s when computer scientist Stuart Haber and physicist W. Scott Stornetta applied cryptographic techniques in a chain of blocks as a way to secure digital documents from data tampering. The work of Haber and Stornetta certainly inspired the work of Dave Bayer, Hal Finney, and many other computer scientists and cryptography enthusiasts - which eventually lead to the creation of Bitcoin, as the first decentralized electronic cash system (or simply the first cryptocurrency). The Bitcoin whitepaper was published in 2008 under the pseudonym Satoshi Nakamoto.

Although the blockchain technology is older than Bitcoin, it is a core underlying component of most cryptocurrency networks, acting as a decentralized, distributed and public digital ledger that is responsible for keeping a permanent record (chain of blocks) of all previously confirmed transactions.

Blockchain transactions occur within a peer-to-peer network of globally distributed computers (nodes). Each node maintains a copy of the blockchain and contributes to the functioning and security of the network. This is what makes Bitcoin a decentralized digital currency that is borderless, censorship-resistant, and that does not require third-party intermediation.

As a distributed ledger technology (DLT) the blockchain is intentionally designed to be highly resistant to modification and frauds (such as double-spending). This is true because the Bitcoin blockchain, as a database of records, cannot be altered, nor can it be tampered without an impractical amount of electricity and computational power - which means the network can enforce the concept of "original" digital documents, making each Bitcoin a very unique and un-copyable form of digital currency.

The so-called Proof of Work consensus algorithm is what made it possible for Bitcoin to be built as a Byzantine fault tolerance (BFT) system, meaning that its blockchain is able to operate continuously as a distributed network, even if some of the participants (nodes) present dishonest behavior or inefficient functionality. The Proof of Work consensus algorithm is an essential element of the Bitcoin mining process.

The technology of blockchain may also be adapted and implemented in other activities, such as healthcare, insurance, supply chain, IOT, and so on. Although it was designed to operate as a distributed ledger (on decentralized systems), it may also be deployed on centralized systems as a way to assure data integrity or to reduce operational costs.

1.3 CRYPTOCURRENCIES

A digital currency that is secured by cryptography and is, typically, used as a medium of exchange within a peer-to-peer (P2P) digital economic system. The use of cryptographic techniques is what ensures that these systems are completely immune to fraud and counterfeiting.

The first cryptocurrency to be ever created was Bitcoin, introduced by pseudonymous developer Satoshi Nakamoto, in 2009. Nakamoto's goal was to create a novel electronic payment system that would allow digital financial transactions to occur between users without the need for intermediaries, such as banks or governmental institutions.

Most cryptocurrency systems work through a decentralized framework that is collectively maintained by a distributed network of computers. Each computer (or device) that joins the network is referred to as a node. In simple terms, a node is any physical device that is connected to a network and that is able to send, receive and forward information. Each node is categorized according to the functions it performs within the system. For instance, the Bitcoin network is made of, at least, seven different types of nodes, and the nodes that perform all available functions are known as full nodes.

Cryptocurrency systems are considered decentralized because they don't rely on a centralized point of authority. The network nodes are widely distributed around the world and the issuance and management of cryptocurrency units are based on pre-programmed algorithms and mathematical proofs. However, each cryptocurrency works in a particular way, which results in varying degrees of decentralization. In other words, some cryptocurrencies may be considered more decentralized than others, depending on the network structure and on how the nodes are distributed.

Most cryptocurrency systems rely on a public distributed ledger known as a blockchain, which is basically an ever-growing list of records that are highly resistant to modification.

As the name suggests, a blockchain is made of a linear chain of blocks and, in the context of cryptocurrencies, it is responsible for keeping a permanent record of all confirmed transactions (and related data) - all secured by cryptography. Generally speaking, every cryptocurrency works on top of a blockchain that works according to a predefined set of rules (i.e. an underlying protocol). The protocol is what defines how the blockchain and the cryptocurrency system should operate.

1.4 CRYPTO EXCHANGES

An exchange is an organized marketplace in which financial instruments – such as cryptocurrencies, commodities, and securities – are traded. An exchange may operate on a real-world facility or on a digital platform. Many traditional exchanges, which were initially restricted to physical trading, are now providing digitized services as a way to enable electronic trading (also known as paperless trading).

One of the main functions of an exchange is to provide liquidity within a secure and organized trading environment, acting as an intermediary for traders to easily buy and sell their assets while being less susceptible to financial risks.

Exchanges may be classified according to the type of trade being executed. Classical exchanges are the ones that perform spot trades (immediate settlement). On the other hand, there are exchanges that provide derivatives trading, such as futures and options. Exchanges may also be classified according to the financial instruments being traded: cryptocurrency exchange, stock or securities exchange, commodities exchange, and the foreign exchange market (Forex). But, many exchanges provide a variety of services and

trading options. For instance, many commodities exchanges are also offering futures trading.

Within stock exchanges, the most important one in a given country is called the primary exchange. A few examples of primary exchanges include the New York Stock Exchange, the Tokyo Stock Exchange, and the London Stock Exchange. Most stock exchanges present a strict listing criterion, which ensures that only companies that meet certain requirements are effectively listed.

In the context of cryptocurrencies, digital exchanges are responsible for providing a platform where users can trade one cryptocurrency for another or buy and sell their coins for fiat money. Currently, most cryptocurrency exchanges are based on a centralized system, maintained by a private company that acts as an intermediary and is responsible for conducting all trades and transactions. Ease of use and liquidity are the major advantages of centralized exchanges. In regards to disadvantages, these centralized systems are susceptible to downtimes and cyber attacks, making security a major point of concern. Considering that users need to trust their holdings to the company in order to be able to trade, it is important to choose an exchange that has proven to be reliable and secure.

In contrast, decentralized cryptocurrency exchanges (also known as DEX) were created as an alternative for centralized exchanges. DEX platforms remove the need for a middleman and perform trades and transactions within a trustless automatized environment (based on smart contracts). Despite the fact that these trading platforms are less susceptible to cyber attacks and infrastructure downtimes, decentralized exchanges are not able to provide fiat currency services, such as fiat/crypto tradings or fiat withdrawals/deposits. In addition, the trading volume tends to be much lower on these types of exchanges, since they are less popular than centralized ones and have limited functionality.

1.4.1 CENTRALIZED CRYPTO EXCHANGES

For most digital currency investors, the centralized cryptocurrency exchange is one of the most important vehicles for transacting. Centralized cryptocurrency exchanges are online platforms used to buy and sell cryptocurrencies. They are the most common means that investors use to buy and sell cryptocurrency holdings. Some investors may find the concept of a "centralized" exchange to be somewhat misleading, as digital currencies themselves are often billed as "decentralized."

In the term "centralized cryptocurrency exchange," the idea of centralization refers to the use of a middle man or third party to help conduct transactions. Buyers and sellers alike trust this middle man to handle their assets. This is common in a bank setup, where a customer trusts the bank to hold his or her money.

The reason for this setup is that banks offer security and monitoring that an individual cannot accomplish on his or her own. In the case of a centralized cryptocurrency exchange, the same principle applies. Transactors trust not only that the exchange will safely complete their transactions for them, but also that it will make use of the network of users in the exchange in order to find trading partners.

In the case of cryptocurrencies, which are often stored in digital wallets, an individual can lose hundreds or thousands of dollars in digital currency holdings simply by forgetting the key to a wallet. An exchange will not allow this to happen, as it safeguards the holdings in place of the individual investor.

Fig 1.9: famous centralized exchanges

1.4.2 DECENTRALIZED CRYPTO EXCHANGES

A decentralized exchange (better known as a DEX) is a peer-to-peer marketplace where transactions occur directly between crypto traders. DEXs fulfill one of crypto's core possibilities: fostering financial transactions that aren't officiated by banks, brokers, payment processors, or any other kind of intermediary. The most popular DEXs — like

Uniswap and Sushiswap — utilize the Ethereum blockchain and are part of the growing suite of decentralized finance (DeFi) tools, which make a huge range of financial services available directly from a compatible crypto wallet. DEXs are booming — in the first quarter of 2021, $217 billion in transactions flowed through decentralized exchanges. As of April 2021, there were more than two million DeFi traders, a ten-fold increase from May 2020.

Unlike centralized exchanges like Coinbase, DEXs don't allow for exchanges between fiat and crypto — instead, they exclusively trade cryptocurrency tokens for other cryptocurrency tokens. Via a centralized exchange (or CEX), you can trade fiat for crypto (and vice versa) or crypto-crypto pairs — say some of your bitcoin for ETH. You can also often make more advanced moves, like margin trades or setting limit orders. But all of these transactions are handled by the exchange itself via an "order book" that establishes the price for a particular cryptocurrency based on current buy and sell orders — the same method used by stock exchanges like Nasdaq.

Decentralized exchanges, on the other hand, are simply a set of smart contracts. They establish the prices of various cryptocurrencies against each algorithmically and use "liquidity pools" — in which investors lock funds in exchange for interest-like rewards — to facilitate trades.

While transactions on a centralized exchange are recorded on that exchange's internal database, DEX transactions are settled directly on the blockchain.

DEXs are usually built on open-source code, meaning that anyone interested can see exactly how they work. That also means that developers can adapt existing code to create new competing projects — which is how Uniswap's code has been adapted by an entire host of DEXs with "swap" in their names like Sushiswap and Pancakeswap.

1.4.3 DIFFERENCES

Centralized exchanges can be used to conduct trades from fiat to cryptocurrency (or vice versa). They can also be used to conduct trades between two different cryptocurrencies. While this may seem to cover all of the potential transaction types, there is still a market for another type of cryptocurrency exchange as well.

Decentralized exchanges are an alternative; they cut out the middle man, generating what is often thought of as a "trustless" environment. These types of exchanges function as peer-to-peer exchanges. Assets are never held by an escrow service, and transactions are done entirely based on smart contracts and atomic swaps.

The crucial difference between centralized and decentralized exchanges is whether or not a middle man is present. Decentralized exchanges are less widespread and less popular as compared with centralized exchanges. Nonetheless, there are more decentralized exchanges all the time, and it's possible that they will give centralized exchanges a run for their money in the future.

1.5 SOME TERMINOLOGIES TO KNOW

Smart Contracts

A smart contract is a piece of computer software that is designed as an automated self-enforcing contract, which means it triggers certain action after predetermined conditions are met. Smart contracts can be used, for instance, as digital agreements that intermediate the exchange of cryptocurrencies (or any other digital asset) between two parties. Once the terms of the agreement have been set, the smart contract verifies their fulfillment and the assets are distributed in accordance.

Wallets

A crypto wallet is a tool that allows users to interact with blockchain networks. They are necessary when sending and receiving Bitcoin and other digital currencies. Crypto wallets can also be used to generate new blockchain addresses.

Unlike the traditional wallets we use in our everyday life, cryptocurrency wallets don’t really store your funds. In fact, your coins (or tokens) are simply part of a blockchain system as pieces of data, and the wallets serve as a means to access them.

Technically speaking, most crypto wallets are able to generate one or more pairs of public and private keys. The public key is used to generate wallet addresses, which are needed to receive payments. The private keys, on the other hand, are used during the creation of

digital signatures and verification of transactions (private keys are confidential and should never be shared with anyone).

Hash

In cryptography, the word hash refers to the output produced by a hash function after a piece of data is submitted (mapped) through it. Other than simply hash, the output produced by hash functions may also be referred to as hash value, hash code, or digest.

Token

Tokens, generally speaking, are non-mineable digital units of value that exist as registry entries in blockchains.

Tokens come in many different forms – they can be used as currencies for specific ecosystems or encode unique. Additionally, some tokens might be redeemable for off-chain assets (i.e., gold, property, stocks).

Tokens are generally issued by companies using existing third-party blockchains such as the Ethereum blockchain, as exemplified by the many ERC-20 tokens that were issued and sold through ICOs in 2017. Strictly speaking, tokens are not cryptocurrencies like Bitcoin or ether, but transferable units of value issued on top of a blockchain.

Faucet

A crypto faucet is an app or a website that distributes small amounts of cryptocurrencies as a reward for completing easy tasks. They’re given the name “faucets" because the rewards are small, just like small drops of water dripping from a leaky faucet.

However, in the case of crypto faucets, tiny amounts of free or earned cryptocurrency are sent to a user’s wallet. In order to get free crypto, users need to complete tasks as simple as viewing ads, watching product videos, completing quizzes, clicking links (be careful!) or completing a captcha.

Crypto faucets are certainly not a get rich quick scheme. The simpler the task, the lesser the reward. Most websites offer a minimum payout threshold, so the rewards earned by completing tasks are deposited into an online wallet of the site.

A user can withdraw this reward only after reaching the minimum set threshold. With the best crypto faucets, this might take just a day, but often, it can take longer than a week.

Testnet

A Testnet is the Minimum Viable Product (MVP) of a blockchain project. It is built for developers to experiment with new ideas without disturbing or breaking the core features of the product. It exists primarily to gather feedback from developers and community members for the purpose of getting the community's pulse on what the mainnet should look like.

Liquidity pooling

Liquidity pools enable users to buy and sell crypto on decentralized exchanges and other DeFi platforms without the need for centralized market makers. A liquidity pool is a crowdsourced pool of cryptocurrencies or tokens locked in a smart contract that is used to facilitate trades between the assets on a decentralized exchange (DEX). Instead of traditional markets of buyers and sellers, many decentralized finance (DeFi) platforms use automated market makers (AMMs), which allow digital assets to be traded in an automatic and permissionless manner through the use of liquidity pools.

Block Explorer

a block explorer is a tool that provides detailed analytics about a blockchain network since its first day at the genesis block. We can say a block explorer acts as a search engine and browser where users can find information about individual blocks, public addresses, and transactions associated with a specific cryptocurrency.

Some block explorers also provide real-time statistics and market charts, as well as data about mining pools, pending transactions, network hash rate, rich list, block validators, orphan blocks, hard forks, and much more.

In regards to pending transactions, block explorers can be useful for users that are waiting for block confirmations. For instance, many exchanges provide their users with the transaction ID of their deposit or withdrawal request so they can track the movement of their funds in real-time.

One of the popular block explorers is Etherscan.

ERC 20

ERC-20 is a technical standard used to issue and implement tokens on the Ethereum blockchain. It was proposed in November 2015 by Ethereum developer Fabian Vogelsteller. The standard describes a common set of rules that should be followed for a token to function properly within the Ethereum ecosystem. Therefore, ERC-20 should not be considered as a piece of code or software. Instead, it may be described as a technical guideline or specification.

2. AIM AND SCOPE

2.1 EXISTING SYSTEM

Centralized cryptocurrency exchanges act as an intermediary between a buyer and a seller and make money through commissions and transaction fees. You can imagine a CEX to be similar to a stock exchange but for digital assets.

Popular Crypto Exchanges are Coinbase, Crypto.com, Gemini, and Binance. Much like stock trading websites or apps, these exchanges allow cryptocurrency investors to buy and sell digital assets at the prevailing price, called spot, or to leave orders that get executed when the asset gets to the investor's desired price target, called limit orders.

CEXs operate using an order book system, which means that buy and sell orders are listed and sorted by the intended buy or sell price. The matching engine of the exchange then matches buyers and sellers based on the best executable price given the desired lot size. Hence, a digital asset's price will depend on the supply and demand of that asset versus another, whether it be fiat currency or cryptocurrency.

CEXs decide which digital asset it will allow trading in, which provides a small measure of comfort that unscrupulous digital assets may be excluded from the CEX.

Disadvantages of Centralized Cryptocurrency Exchanges

1. Hacking risk

Centralized exchanges are operated by companies that are responsible for the holdings of their customers. Large exchanges usually hold billions of dollars worth of bitcoin, making them a target for hackers and theft.

An example of such an incident is Mt.Gox, which was once the world's largest cryptocurrency exchange company before it reported the theft of 850,000 bitcoins, leading to its collapse.

2. Transaction fees

Unlike peer-to-peer transactions, centralized exchanges often charge high transaction fees for their services and convenience, which can be especially high when trading in large amounts.

3. Custody of digital assets

Lastly and most importantly, most CEXs will hold your digital asset as a custodian in their own digital wallet rather than allow you to store your private keys on your own digital wallet. While more convenient when you want to trade, there are drawbacks, namely the risk of the centralized cryptocurrency exchange failing and fraud.

2.2 PROPOSED SYSTEM

A web app is built with the help of ReactJs for designing the ui and the uniswpa api is used for background operations of the system to develop a functional decentralized web app.

Decentralized exchange is a non-centralized alternative to centralized exchange in which no single entity is in charge of the assets. In contrast to traditional centralized exchanges, smart contracts and decentralized apps are used to automate transactions and trades. This method is far safer since no security breach is possible, provided the smart contract is properly constructed.

Advantages of Decentralized Cryptocurrency Exchanges

1. Custody

Users of decentralized exchanges do not need to transfer their assets to a third party. Therefore, there is no risk of a company or organization being hacked, and users are assured of greater safety from hacking, failure, fraud, or theft.

2. Preventing market manipulation

Due to their nature of allowing for the peer-to-peer exchange of cryptocurrencies, decentralized exchanges prevent market manipulation, protecting users from fake trading and wash trading.

3. Less censorship

Decentralized exchanges do not require customers to fill out know-your-customer (KYC) forms, offering privacy and anonymity to users. Since DEXs don't exercise censorship, more cryptocurrencies and digital assets are available than through a CEX. As a matter of fact, many Altcoins are only available on DEXs.

2.3 FEASIBILITY STUDY

2.3.1 TECHNICAL FEASIBILITY

In an ideal setting/all factors being constant, below are some of the things that could be done to make crypto currencies feasible.

Production; Progress has and is being made, seeing the shift of bitcoin mining from using CPUs, to graphics processing units (GPUs), to FPGAs to Application-Specific Integrated Circuits (ASICs) which were designed specifically for mining bitcoin. This shows progress from the teams that are working towards making the future of bitcoin production more feasible. There's also need to locate near cheap power sources to minimize production costs. This may not be ideal for regions where energy for basic use is already rationed, say in the case of developing countries.

Scaling; The set limits and caps to bitcoin mining would have to be eliminated to enable feasible supply to an extent, perhaps let the market forces dictate its supply, this will require easing the production process as well so that more blocks and coins can be mined.

Skills; Currently, there's a big skills gap in most innovative facets of technology, including blockchain and cryptocurrencies. Some of the ideal and necessary skills for a sustainable bitcoin workforce may include proper understanding of how currencies operate and the economics thereof, familiarity and experience with certain programming system languages like C/C++, understanding data security and anomaly detection, to mention but and few. And to top it all, this being a new thing, there will be plenty of room for trial and error before actual implementation in highly sensitive industry structures, especially in Finance.

2.3.2 OPERATIONAL FEASIBILITY

Standard operating procedures and legislation in place for the crypto currency/blockchain industry to provide a respite in case things go wrong. Someone to take clear responsibility.

Get the end users to understand and appreciate the advantages of crypto currencies that are not being provided by existing currencies. Explain this in ways that a normal lay man person understands.

Make them more inflationary-This could happen by removing the existing caps and letting the markets dictate production. Find ways to make it the currency of operation so that demand and supply are driven by the end users.

Regulating blockchain/crypto currencies

Poolers/Bitcoin miners should create a legislative or compliance body which can come up with standard operating policies and procedures across the crypto networks, from a technical point of view so that realistic expectations and realistic rules are set. This would bring us back to the centralized control issues, which the "artistic" techies care so much about, but since they're going all public and commercial with it, I don't suppose they really should be fussy given they are all going commercial now and need to adjust to the commercial scene.

Set up a Superior, independent, incorruptible governing body that is genuinely impartial to monitor and govern the activities of the blockchains and bitcoin networks, and not limit this to individual states as this will leave room for the same issues facing current fiat systems that are trying to be eliminated.

Regulators should not try to curb or totally burn out crypto currencies and label them all bad. Most of the bitcoin mining is happening back doors anyway and trying to stop and prevent it completely will be more like the war on drugs.

Create clear legal guidelines that should abound across the platform to avoid bitcoin miners and traders going off with the end users' investments should something happen. Someone should be liable and take responsibility should things break down or should legal scenarios arise.

There should be an integration of reflagged clients or persons/bodies into the blockchain/bitcoin networks so as the systems flags or blocks potentially illegal activities from being facilitated via the platform. If you have mining pools voting on fraudulent transactions or setting up governing rules on blockchain and crypto currency traders, this could increase trust both from the public and from governments that are necessary if this is to be taken up on a wider scale.

Focusing adaptability to current environment, for example mobile payment systems, setting regulations/ improvements along those lines and growing what's already there instead of reinventing the whole wheel. Gauge current operating areas where integration of aspects of digital currencies and blockchain can be integrated.

Documentation and clear guidelines on what's going on in the back end or policies across all blockchain and cryptocurrency platforms/pools. This will not only phase out the potential pyramid schemes, it will also make sustainability of the networks possible for training purposes and for business continuity.

The benefits and feasibility of cryptocurrencies are all in an idealistic setting and should they be brought to applicability, that would be advantageous and beneficial to various stakeholders. However, if left unchecked, this growing crypto-mania could be hugely destructive to its underlying technology, blockchain, which if well positioned could revolutionize various industries.

2.3.3 ECONOMIC FEASIBILITY

Make cryptocurrencies adaptable to major economic events to ensure a degree of continuity of economic systems should a major imbalance happen.

Incorporate crypto currencies into the existing fiat systems, to create a glue between the fiat currencies/systems with the digital ones; merge with each other to make the most of advantages of each and mitigate the disadvantages.

Quantum computing, to eliminate possible threats that come from security breaches in the cryptocurrency network.

Solve computer security, perhaps through biometric identification for user verification to avoid cryptography bypass and hacking into the networks at their points of weakness.

3. CONCEPTS AND METHODS

3.1 PROBLEM DESCRIPTION

To develop a web application that performs exchange of crypto tokens in a decentralized manner with out involvement of any other intermediaries by making use of blockchain technology.

3.2 PROPOSED SOLUTION

A web app is developed using modern technologies like react and built the smart contracts, deployed them to blockchain which will operate in the backend. The transactions can be monitored on the ether scan portal to confirm them.

3.3 SYSTEM ANALYSIS METHODS

3.3.1 USECASE DIAGRAM

A use case diagram is used to represent the dynamic behavior of a system. It encapsulates the system's functionality by incorporating use cases, actors, and their relationships. It models the tasks, services, and functions required by a system/subsystem of an application. It depicts the high-level functionality of a system and also tells how the user handles a system.

The main purpose of a use case diagram is to portray the dynamic aspect of a system. It accumulates the system's requirement, which includes both internal as well as external influences. It invokes persons, use cases, and several things that invoke the actors and elements accountable for the implementation of use case diagrams. It represents how an entity from the external environment can interact with a part of the system.

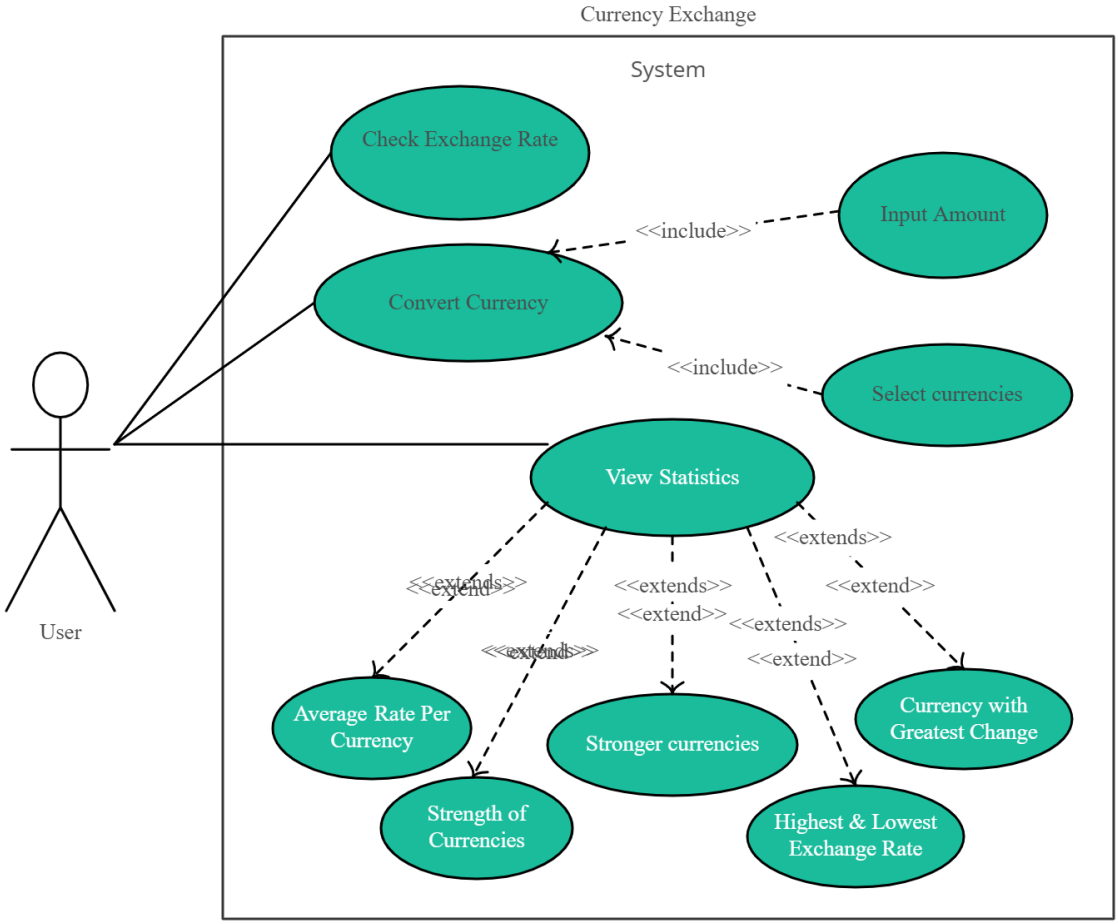

Fig 3.1: Use-case diagram for cryptocurrency exchange

3.3.2 ACTIVITY DIAGRAM

The activity diagram is used to demonstrate the flow of control within the system rather than the implementation. It models the concurrent and sequential activities. The activity diagram helps in envisioning the workflow from one activity to another. It put emphasis on the condition of flow and the order in which it occurs. The flow can be sequential, branched, or concurrent, and to deal with such kinds of flows, the activity diagram has come up with a fork, join, etc.

It is also termed as an object-oriented flowchart. It encompasses activities composed of a set of actions or operations that are applied to model the behavioral diagram.

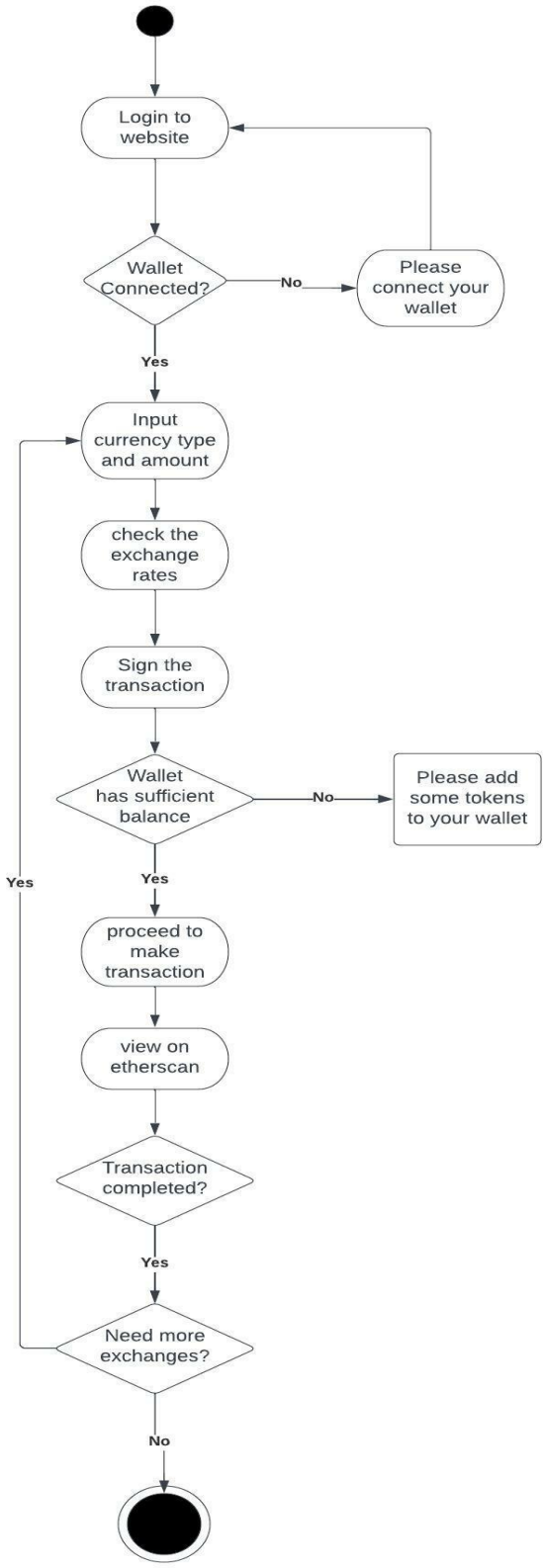

Fig 3.2: Activity diagram for cryptocurrency exchange

3.4 SYSTEM REQUIREMENTS

3.4.1 Hardware Requirements:

The hardware requirements may serve as the basis for a contract for the implementation of the system and should therefore be a complete and consistent specification of the whole system. They are used by the software engineers as the starting point for the system design. It shows what the system does and not how it should be implemented.

- HARD DISK : 12 GB
- MONITOR : 15” LCD
- INPUT DEVICE : KEYBOARD, MOUSE
- RAM : 8GB

3.4.2 Software Requirements:

The software requirements document is the software specification of the system. It should include both a definition and specification of requirements. It is a set of what the system should do rather than how it should do it. It is useful in estimating cost, planning team activities, performing tasks and tracking the team’s progress throughout the development activity.

- OPERATING SYSTEM : Windows/ Mac/ Linux
- PLATFORM : IA-32, X86-64

3.5 SYSTEM DESIGN

System Design is the process of designing the architecture, components, and interfaces for a system so that it meets the end-user requirements. It's a wide field of study in Engineering and includes various concepts and principles that will help you in designing scalable systems.

3.5.1 Class Diagram

Class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application.

Class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modeling of object oriented systems because they are the only UML diagrams, which can be mapped directly with object-oriented languages. Class diagram shows a collection of classes, interfaces, associations, collaborations, and constraints. It is also known as a structural diagram.

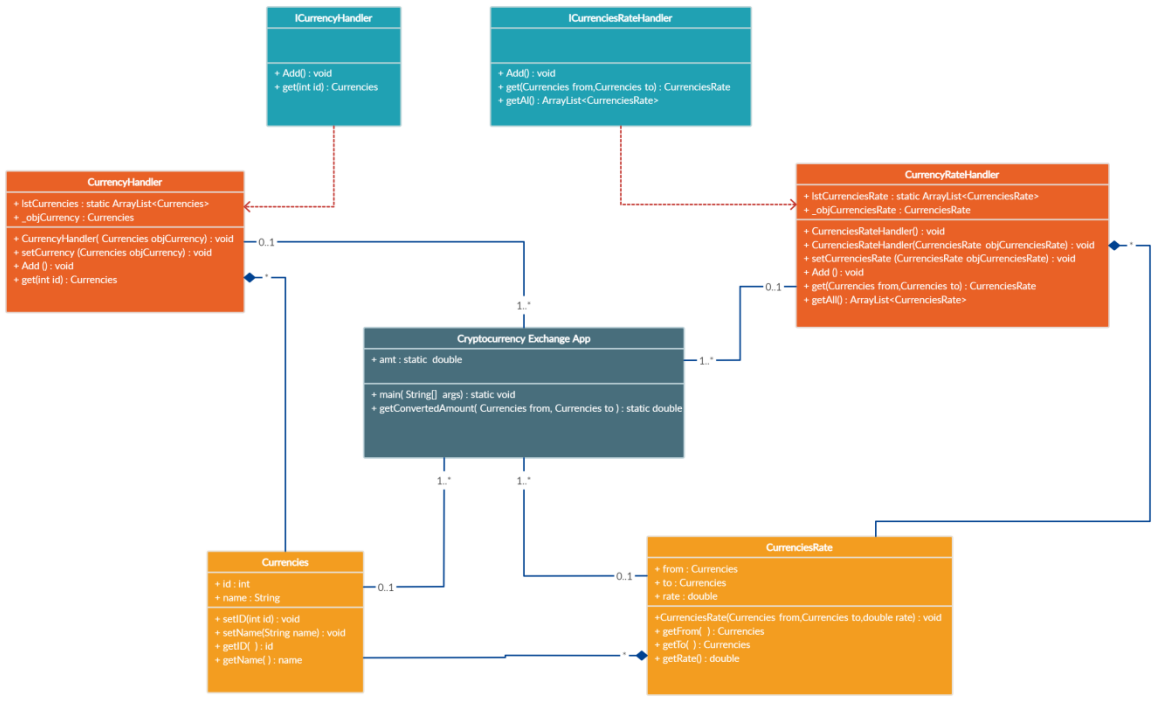

Fig 3.3: class diagram from cryptocurrency exchange

3.5.2 Sequence Diagram

A sequence diagram simply depicts interaction between objects in a sequential order i.e. the order in which these interactions take place. We can also use the terms event diagrams or event scenarios to refer to a sequence diagram. Sequence diagrams describe how and in what order the objects in a system function. These diagrams are widely used to document and understand requirements for new and existing systems.

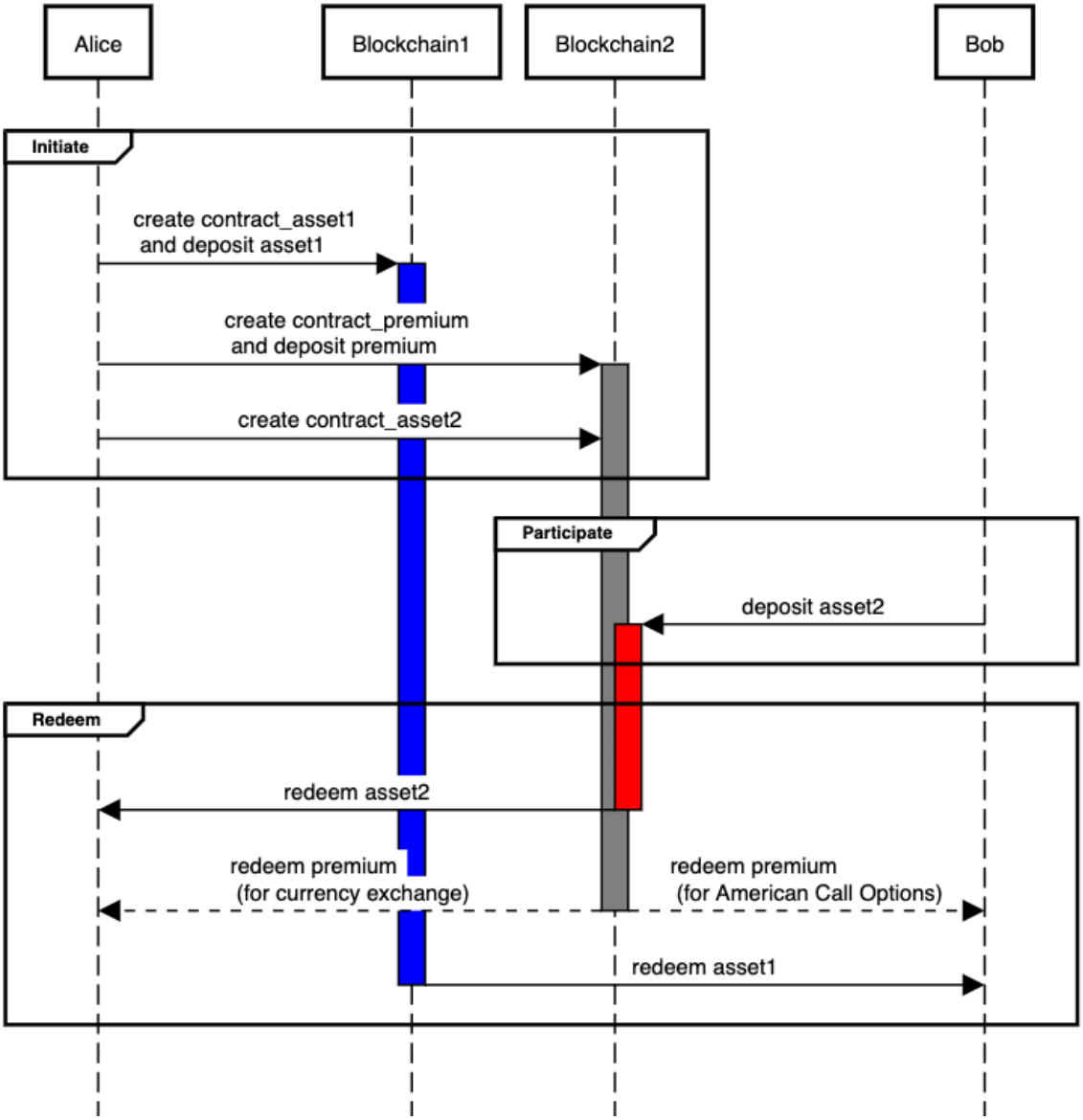

Fig 3.4: sequence diagram for cross chain swap

4. IMPLEMENTATION

4.1 TOOLS USED

Visual Studio Code:

The visual studio code is a powerful development environment that lets you edit, debug, and publish apps. Although it is known as IDE for C, C++, C# etc.. it can be easily used for Javascript development environment.

Some of the best VSCode extensions for javascript developers:

- JavaScript(ES6) code snippets
- ESLint
- Prettier
- REST client
- Live Share
- Live Server

Git:

Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency. Git is easy to learn and has a tiny footprint with lightning fast performance.

Github:

GitHub is a code hosting platform for version control and collaboration. It lets you and others work together on projects from anywhere.

Browser:

In conventional programming languages, the source code is passed through a program called compiler, which translates it into byte code and that the machine understands and code can execute. In contrast, JavaScript has no compilation step, instead an interpreter in the browser reads over the javascript code, interprets each line and executes it.

CranQ IDE:

CRANQ is the graphical and intuitive IDE designed for discovery, and re-use of existing code and smart contracts.

NodeJS SDK:

Node.js is a cross-platform JavaScript runtime environment for servers and applications. It is built on a single-threaded, non-blocking event loop, the Google Chrome V8 JavaScript engine, and a low-level I/O API.

WEBSITES USED

Alchemy:

Alchemy is the platform layer needed to empower developers to build great applications that tap into the blockchain revolution

Use-Dapp:

useDApp is a framework to make DApp UX & DX better, built on top of modern DApp stack: React, ethers. js, TypeScript, TypeChain, and Multicall.

Metamask:

Metamask is a software cryptocurrency wallet used to interact with the Ethereum blockchain. It allows users to access their Ethereum wallet through a browser extension or mobile app, which can then be used to interact with decentralized applications.

Goerli Faucet:

Goerli is a testnet, a decentralized computing network who's ledger is separate from the main Ethereum ledger, so transactions do not cross over between the two. It runs on a different consensus system, proof of authority, rather than the Ethereum mainnet proof of stake (PoS).

Goerli Scan:

Etherscan allows you to explore and search the Goerli blockchain for transactions, addresses, tokens, prices and other activities taking place on Goerli testnet.

LANGUAGES USED

ReactJS:

React JS is a JavaScript library used in web development to build interactive elements on websites. React is a JavaScript library that specializes in helping developers build user interfaces, or UIs. In terms of websites and web applications, UIs are the collection of on-screen menus, search bars, buttons, and anything else someone interacts with to USE a website or app.

In 2011, Facebook engineer Jordan Walke created React JS specifically to improve UI development.

In addition to providing reusable React library code (saving development time and cutting down on the chance for coding errors), React comes with two key features that add to its appeal for JavaScript developers:

- JSX : JSX stands for JavaScript XML. JSX allows us to write HTML in React. JSX makes it easier to write and add HTML in React.
- Virtual DOM : The virtual DOM (VDOM) is a programming concept where an ideal, or “virtual”, representation of a UI is kept in memory and synced with the “real” DOM by a library such as ReactDOM.

Tailwind CSS:

Tailwind CSS is basically a utility-first CSS framework for rapidly building custom user interfaces. It is a highly customizable, low-level CSS framework that gives you all of the building blocks you need to build bespoke designs without any annoying opinionated styles you have to fight to override.

4.2 COMPONENT DIAGRAM

A component diagram is used to break down a large object-oriented system into the smaller components, so as to make them more manageable. It models the physical view of a system such as executables, files, libraries, etc. that resides within the node.

It visualizes the relationships as well as the organization between the components present in the system. It helps in forming an executable system. A component is a single unit of the system, which is replaceable and executable. The implementation details of a component are hidden, and it necessitates an interface to execute a function. It is like a black box whose behavior is explained by the provided and required interfaces.

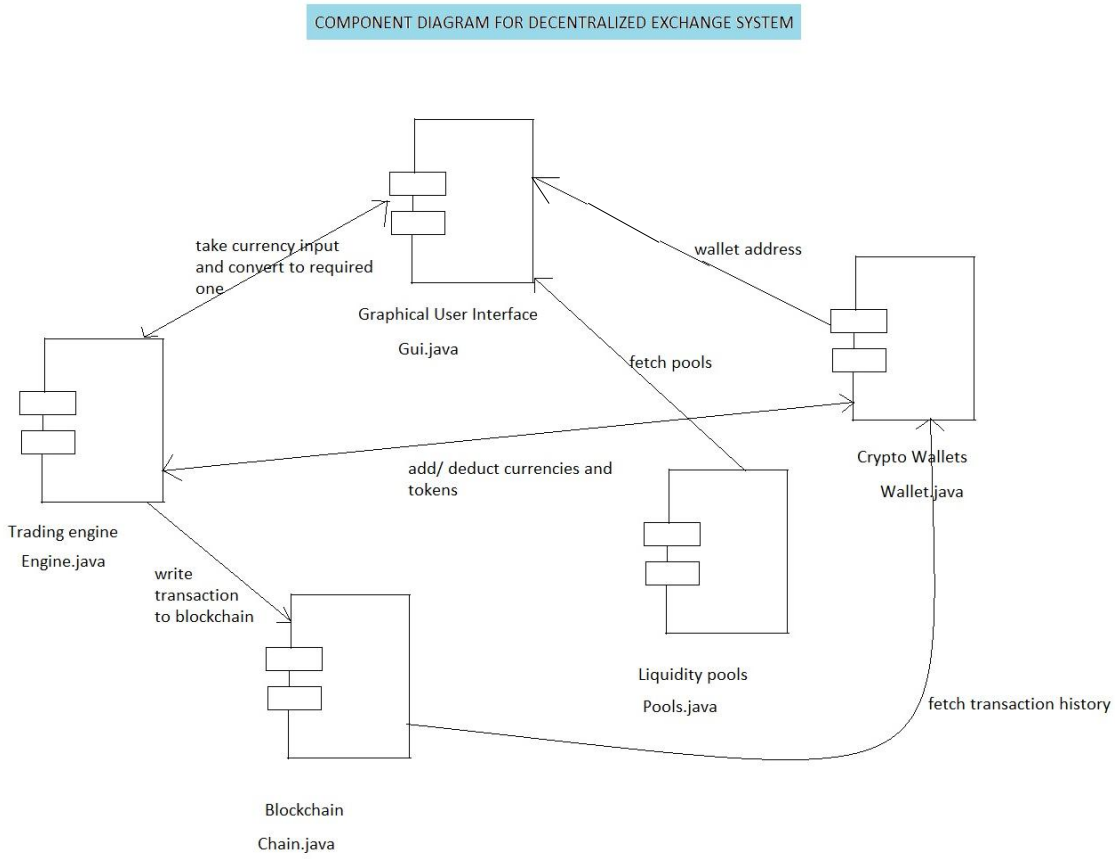

Fig 4.1: component diagram

4.3 PSEUDO CODE/ ALGORITHMS

4.3.1 CODING ENVIRONMENT SETUP

To start working with useDapp you need to have a working React environment. To get started, add the following npm package @usedapp/core and its peer dependency in your project:

```
npm install @usedapp/core ethers
```

The first thing you need to do is set up **DAppProvider** with optional config and wrap your whole App in it.

```
<DAppProvider>
<App />
</DAppProvider>
```

Then you need to activate the provider using **activateBrowserWallet**. It's best to do when the user clicks "Connect" button.

Install other necessary dependencies:

```
"@my-app/contracts": "^1.0.0",
"@apollo/client": "^3.5.10",
"@ethersproject/contracts": "^5.6.0",
"@ethersproject/providers": "^5.6.0",
"@testing-library/dom": "^8.11.3",
"@testing-library/jest-dom": "^5.16.2",
"@testing-library/react": "^12.1.4",
"@testing-library/user-event": "^13.5.0",
"@types/react": "^17.0.40",
"ethers": "^5.7.0",
"graphql": "^16.3.0",
"ipfs-deploy": "^11.2.0",
"react": "17.0.2",
"react-dom": "17.0.2",
```

```
"react-scripts": "4.0.3",
"styled-components": "^5.3.3",
"@usedapp/core":"^1.1.5",
"@uniswap/sdk":"^3.0.2",
"@uniswap/v2-core":"^1.0.1",
"@uniswap/v2-periphery":"^1.1.0-beta.0",
"web3":"^1.7.5"
```

- The goal of ***ipfs-deploy*** is to make it as easy as possible to deploy a static website to IPFS.
- ***Apollo Client*** is a fully-featured caching GraphQL client with integrations for React, Angular, and more. It allows you to easily build UI components that fetch data via GraphQL.
- ***@ethersproject/providers*** is part of the ethers project. It contains common Provider classes, utility functions for dealing with providers and re-exports many of the classes and types needed to implement a custom Provider.
- ***@ethersproject/contracts*** is part of the ethers project. It is creating (at run-time) an object which interacts with an on-chain contract as a native JavaScript object. If you are familiar with ORM for Databases, this is similar, but for smart contracts.
- ***@types/react*** package contains type definitions for React
- ***Ethers*** is a complete Ethereum wallet implementation and utilities in JavaScript (and TypeScript).
- ***Graphql*** is the JavaScript reference implementation for GraphQL, a query language for APIs created by Facebook.
- ***React*** is a JavaScript library for creating user interfaces. The react package contains only the functionality necessary to define React components. It is typically used together with a React renderer like react-dom for the web, or react-native for the native environments.
- ***React-scripts*** package includes scripts and configuration used by Create React App.
- ***Styled-components*** provides visual primitives for the component age. It use the best bits of ES6 and CSS to style your apps without stress.

- ***@usedapp/core*** is a framework for rapid Dapp development. Simple. Robust. Extendable. Testable.

- ***@uniswap/v2-core*** contains core contracts for the UniswapV2 protocol.

- ***@uniswap/v2-periphery*** contains peripheral smart contracts for interacting with Uniswap V2

- ***Web3*** is the main package for web3.js

4.3.2 PSEUDO CODES, FUNCTIONS AND STYLING

Tailwind styles used: tailwind styling is used as internal styles

- mx : horizontal margin
- px : padding horizontal
- mb : margin bottom
- container : sets max width of an element
- grid : By default, Tailwind includes grid-template-column utilities for creating basic grids with up to 12 equal width columns
- lg: : specifies how the styles should appear on the large screen
- bg : background of an element
- border – b : indicates black border(short hand notation of colours)
- w-full : short hand operator for width of a component
- inline-block : used to control the flow of text and elements.
- float-left/right : The float class defines the flow of content for controlling the wrapping of content around an element.
- cursor-pointer : In this class, the cursor is a pointer and indicates a link
- hidden : to set an element to display: none and remove it from the page layout
- overflow : utilities for controlling how an element handles content that is too large for the container.

- Transition : Utilities for controlling which CSS properties transition.

- Duration : Utilities for controlling the duration of CSS transitions.
- Hover : Style elements on hover, focus, and active using the hover, focus, and active modifiers
- Flex : Utilities for controlling how flex items both grow and shrink.
- Align-items : Utilities for controlling how flex and grid items are positioned along a container's cross axis.
- Justify : Utilities for controlling how flex and grid items are positioned along a container's main axis.
- Rounded : Utilities for controlling the border radius of an element.

Keywords and Tags in reactjs:

- className : class is a keyword in javascript and JSX is an extension of javascript. That's the principal reason why React uses className instead of class

eg:
```
import "./App.css";
function App() {
return <h1 className="heading1">This is an example code</h1>;
}
export default App;
```

- Link : Client-side transitions between routes can be enabled via the Link component exported by next/link.
  - Eg:

```
import Link from 'next/link';
function Home() {
  return (
    <ul>
      <li>
<Link
        href={{
          pathname: '/about',
          query: { name: 'test' },
        }}
```

```
          >
            <a>About us</a>
          </Link>
        </li>
        <li>
          <Link
            href={{
              pathname: '/blog/[slug]',
              query: { slug: 'my-post' },
            }}
          >
            <a>Blog Post</a>
          </Link>
        </li>
      </ul>
    )
}
export default Home
```

- moment : MomentJS is a JavaScript library which helps is parsing, validating, manipulating and displaying date/time in JavaScript in a very easy way.
    - eg: .format('MM DD YYYY')
- html-react parser : The parser converts an HTML string to one or more React elements.

Key functions used:

- map() : map() creates a new array from calling a function for every array element. map() calls a function once for each element in an array. map() does not execute the function for empty elements. map() does not change the original array.

```
Eg: const numbers = [65, 44, 12, 4];
const newArr = numbers.map(myFunction)
function myFunction(num) {
  return num * 10;
}
```

- {props} : It is an object which stores the value of attributes of a tag and work similar to the HTML attributes. It gives a way to pass data from one component to

other components. It is similar to function arguments. Props are passed to the component in the same way as arguments passed in a function.

- Children props : children are a special prop that is used to pass the data from the parent component to the children component but this data must be enclosed within the parent's opening and closing tag.
- useEffect() : useEffect(callback, dependencies) is the hook that manages the side-effects in functional components. callback argument is a function to put the side-effect logic. dependencies is a list of dependencies of your side-effect: being props or state values.
- useState() : The React useState Hook allows us to track state in a function component. State generally refers to data or properties that need to be tracking in an application.
- useRef() : The useRef Hook allows you to persist values between renders. It can be used to store a mutable value that does not cause a re-render when updated. It can be used to access a DOM element directly.
- Async - await : The word "async" before a function means one simple thing: a function always returns a promise. Other values are wrapped in a resolved promise automatically.

  The keyword await makes JavaScript wait until that promise settles and returns its result.

```
Eg: async function f() {
        let promise = new Promise((resolve, reject) => {
        setTimeout(() => resolve("done!"), 1000)
        });
        let result = await promise; // wait until the promise resolves
(*)
        alert(result); // "done!"
        }
        f();
```

- then : The then() method in JavaScript has been defined in the Promise API and is used to deal with asynchronous tasks such as an API call. Previously, callback functions were used instead of this function which made the code difficult to maintain.

- getStaticProps() : You should use getStaticProps if: The data required to render the page is available at build time ahead of a user's request. The data comes from a headless CMS. The page must be pre-rendered (for SEO) and be very fast — getStaticProps generates HTML and JSON files, both of which can be cached by a CDN for performance.

Writing a basic react function component:

New.jsx

```
import React from 'react'
const New = () => {
  return (
    <div>new</div>
  )
}
export default New;
```

Exploring the COMPONENTS folder:

- ***AmountIn.jsx:*** renders the input amount field, takes the amount and currency type.
- ***AmountOut.jsx:*** calculates the market price and display the amount obtained in the converted currency.
- ***Balance.jsx:*** renders the balance of the tokens in the wallet.
- ***Exchange.jsx:*** holds all the other components and contains other backend functions.
- ***Loader.jsx:*** it contains the loader svg which will render when the connection is slow.
- ***WalletButton.jsx:*** connects the wallet to website.

4.4 DEPLOYMENT DIAGRAM

The term Deployment itself describes the purpose of the diagram. Deployment diagrams are used for describing the hardware components, where software components are deployed. Component diagrams and deployment diagrams are closely related. Component diagrams are used to describe the components and deployment diagrams shows how they are deployed in hardware.

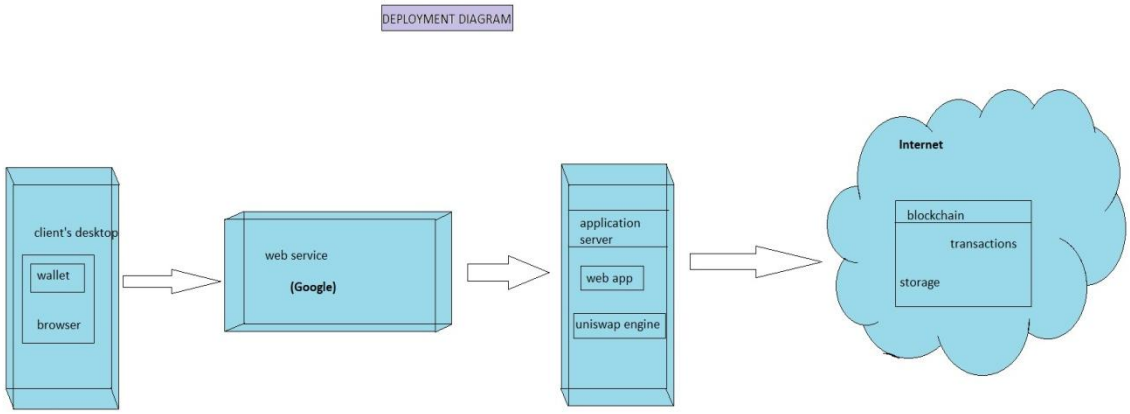

Fig 4.2: Deployment diagram

4.5 SCREENSHOTS

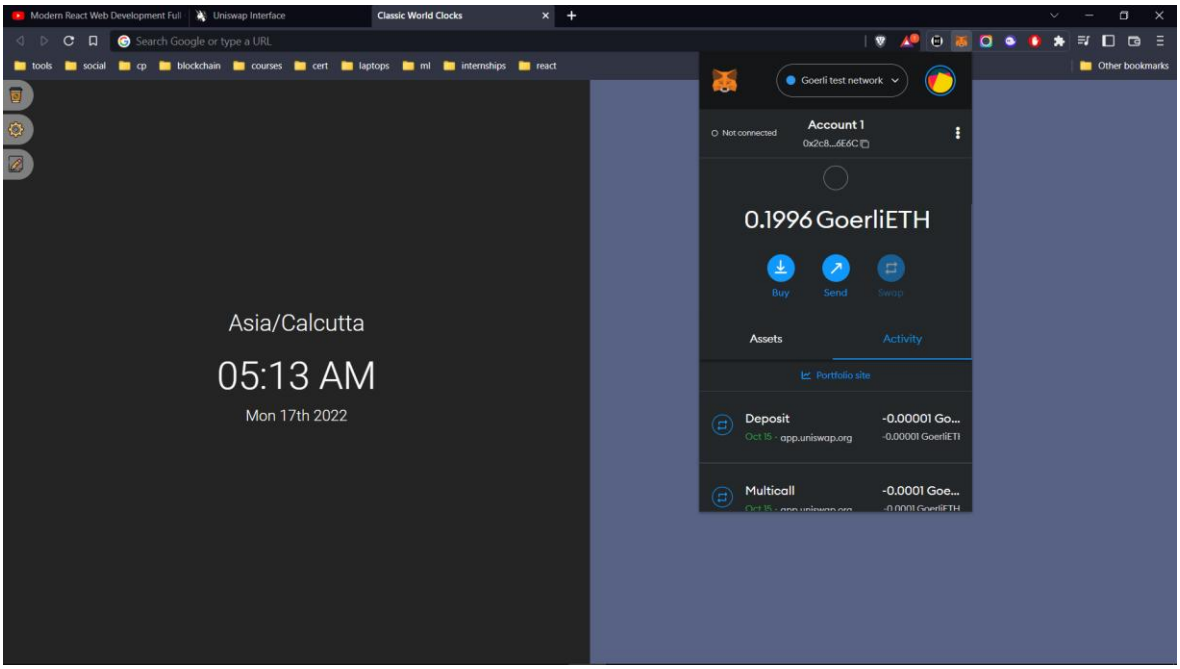

Fig 4.5.1: Check balance in wallet

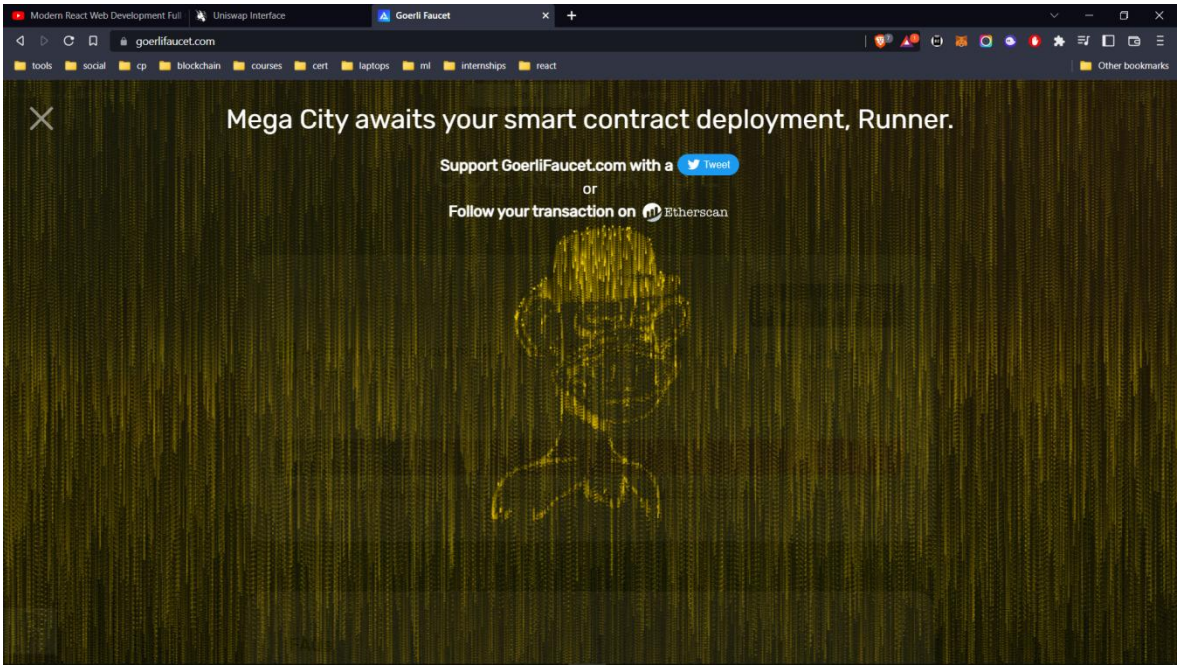

Fig 4.5.2: Request faucet

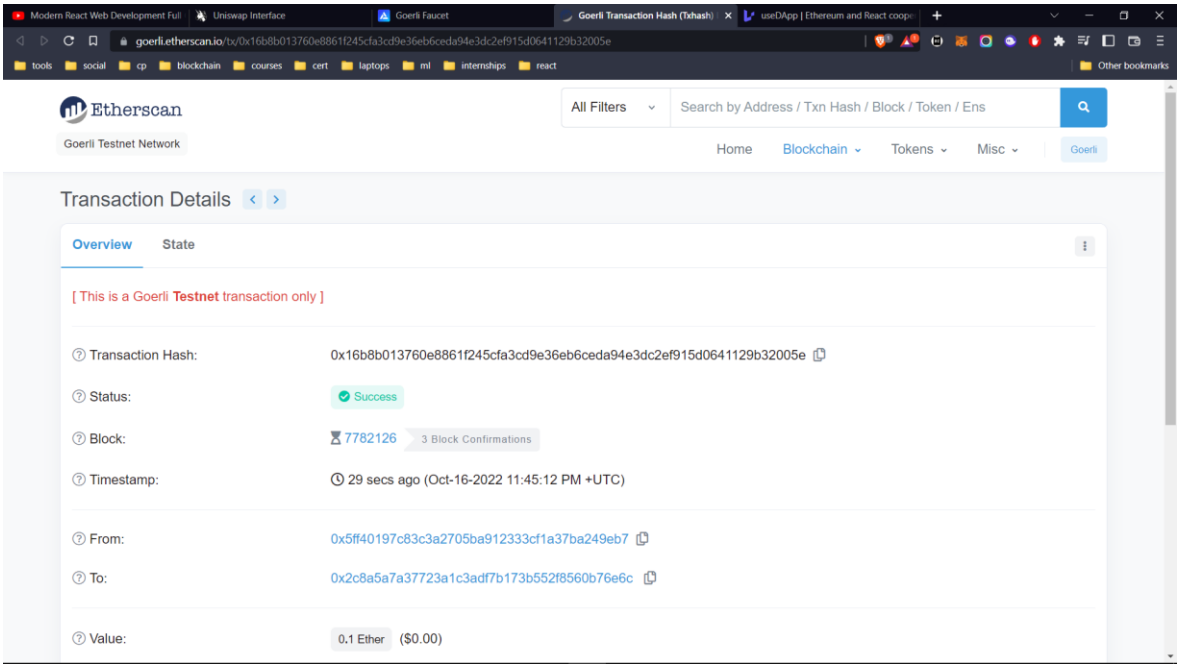

Fig 4.5.3: view transaction on Etherscan

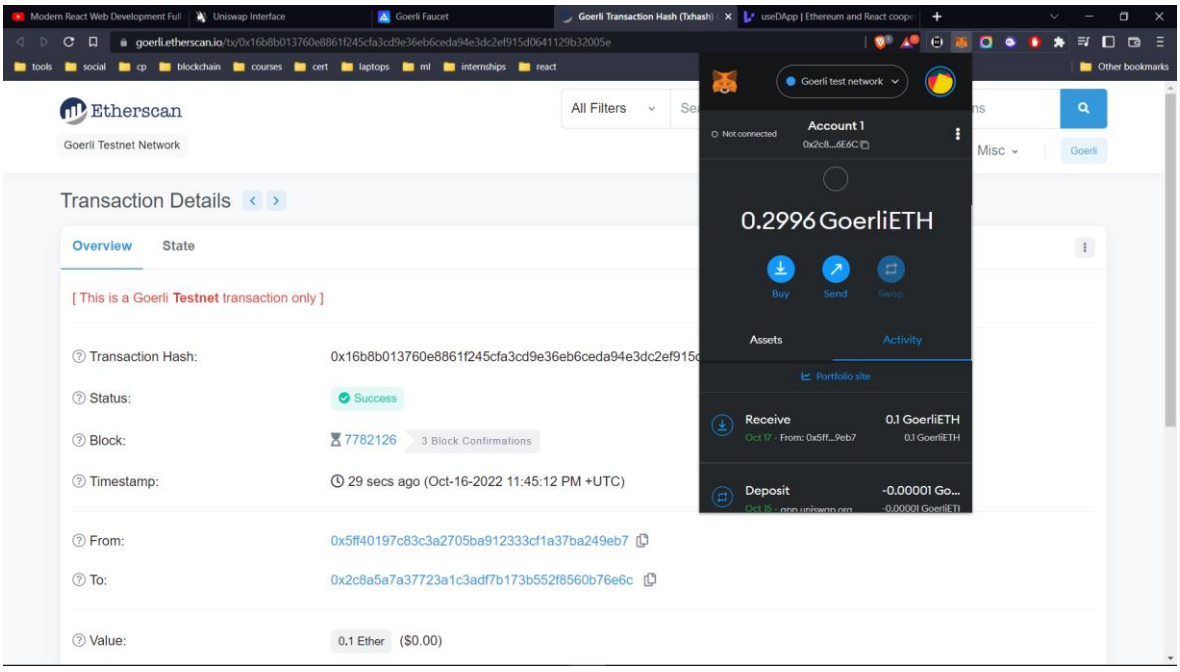

Fig 4.5.4: switch between networks

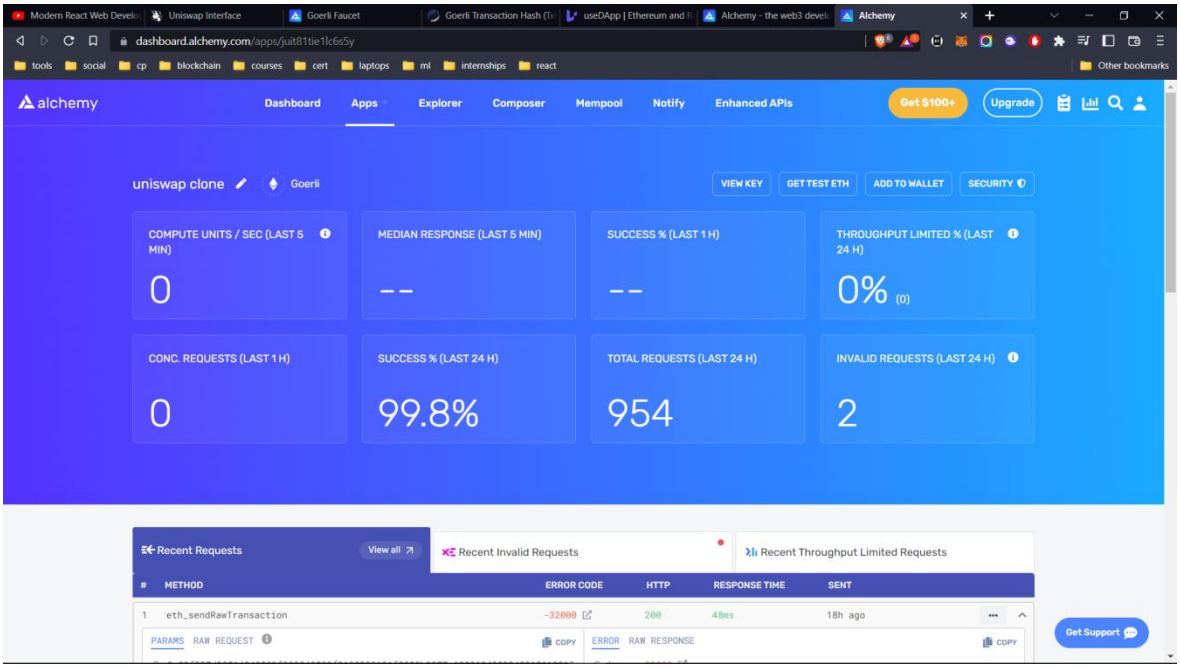

Fig 4.5.5: create and configure Alchemy

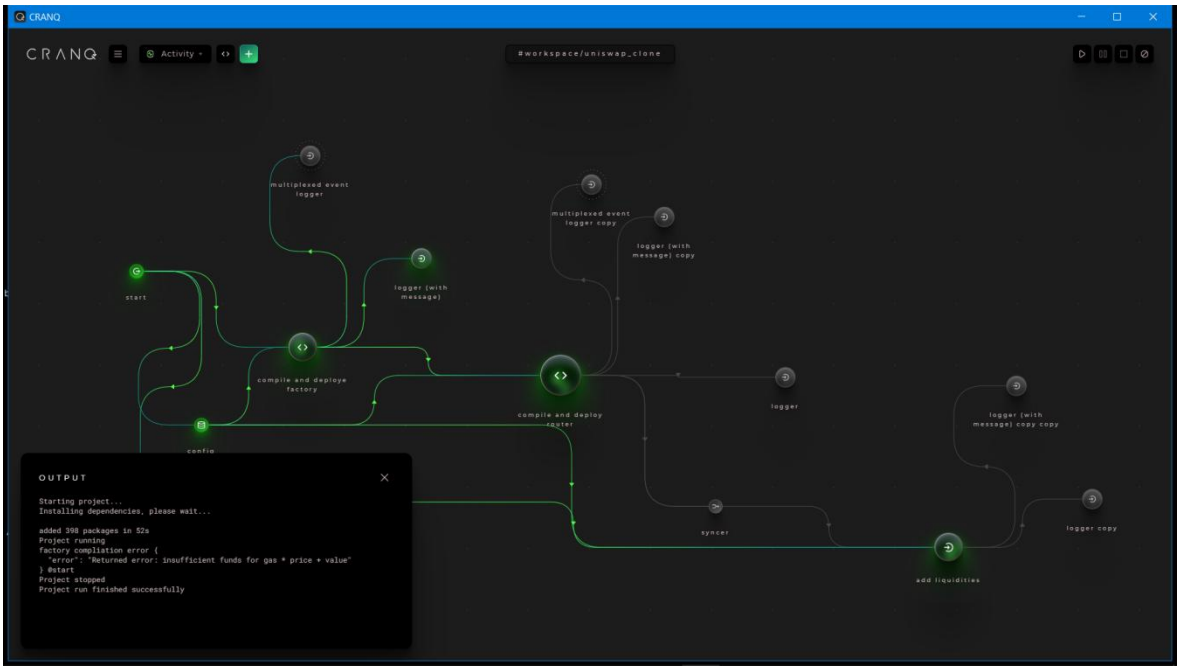

Fig 4.5.6: Execute and deploy contract to the network in CranQ IDE

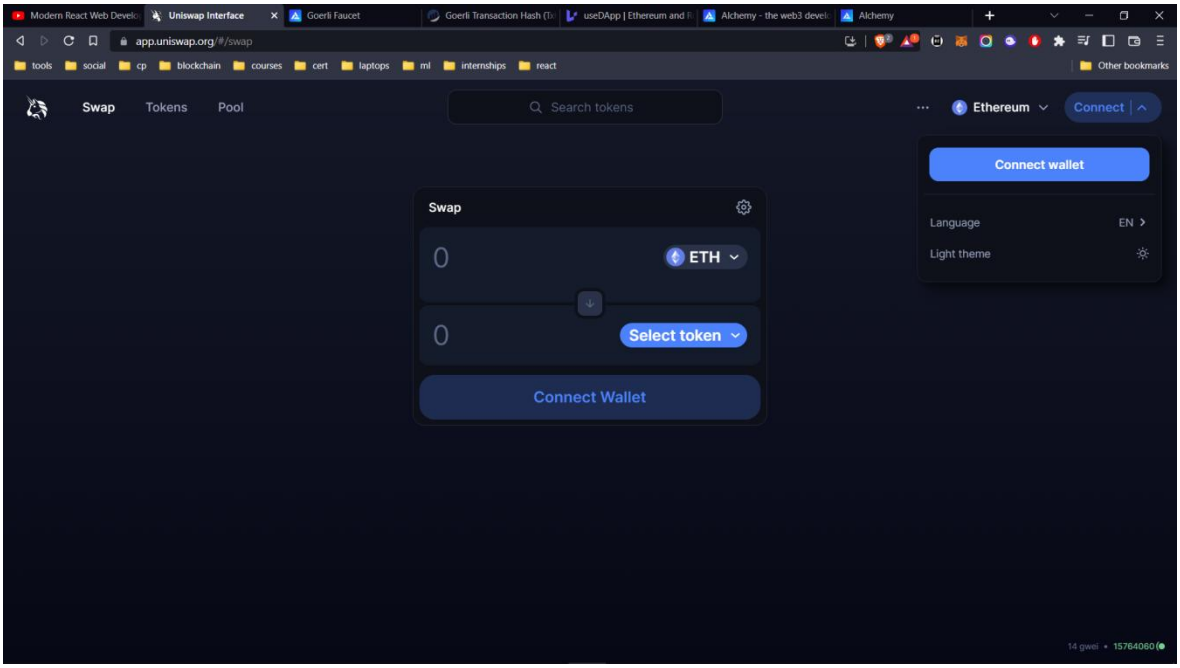

Fig 4.5.7: Connect wallet to the web app

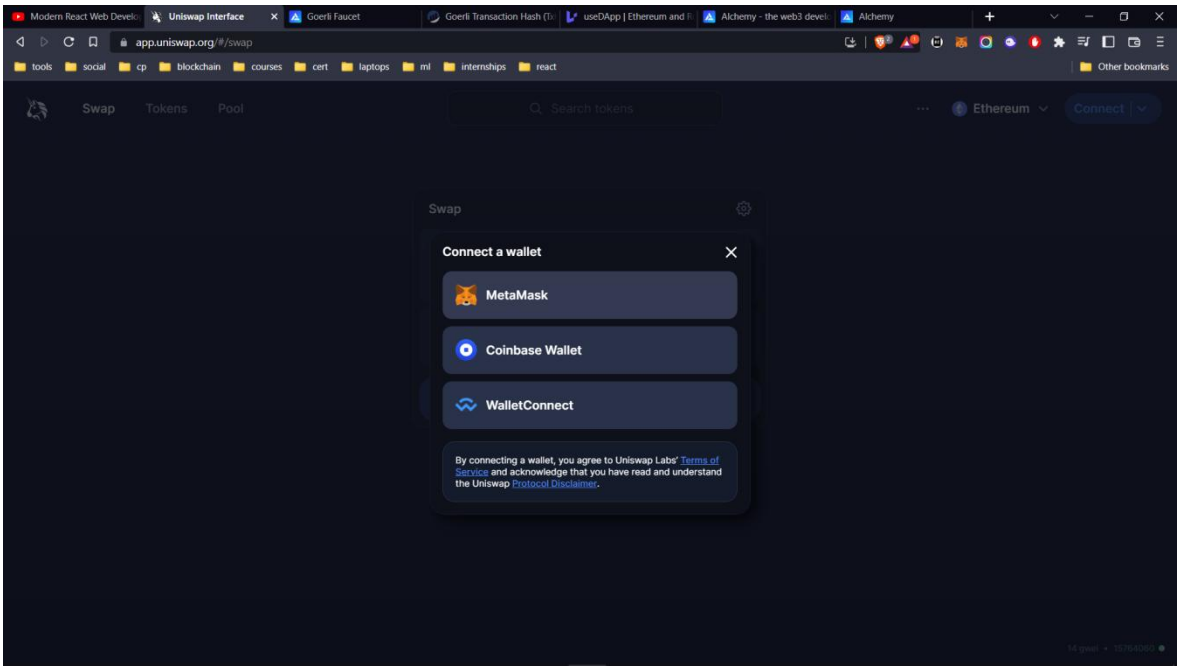

Fig 4.5.8: Choose wallet

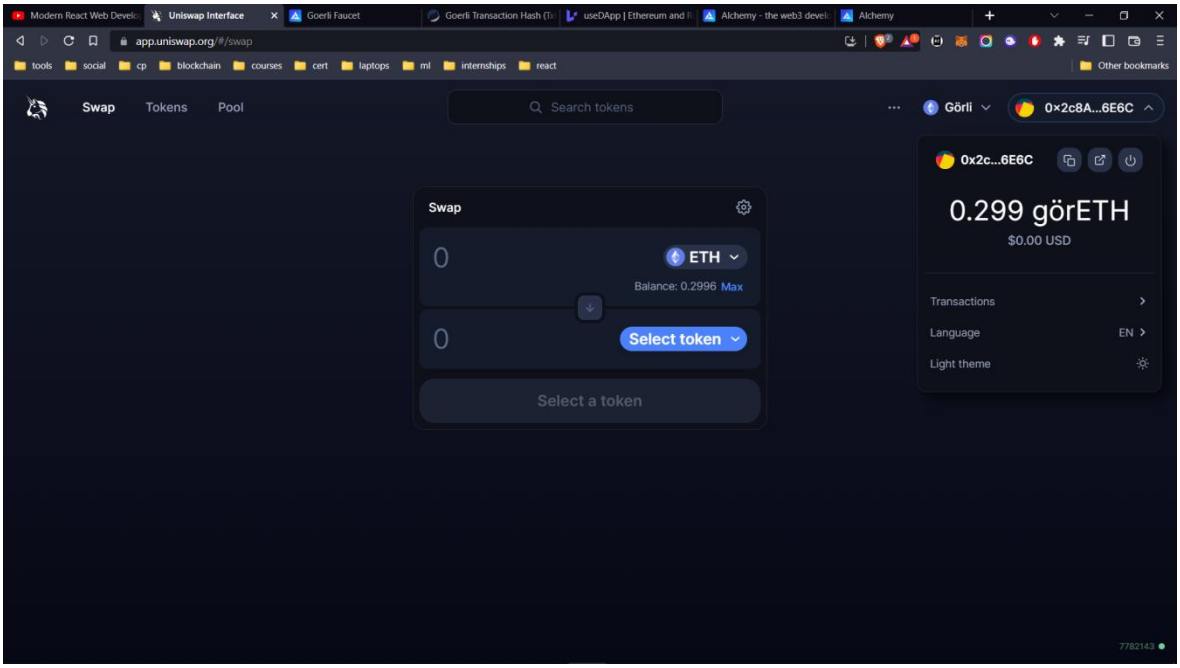

Fig 4.5.9: Check for sufficient balance

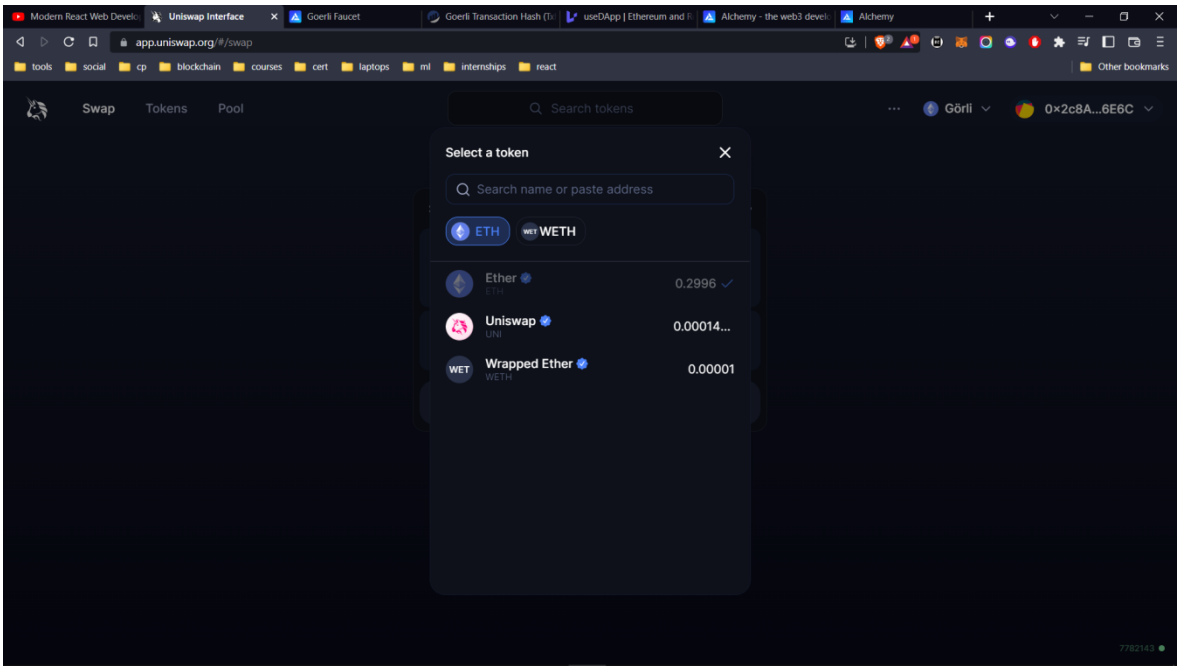

Fig 4.5.10: choose input amount, currency types

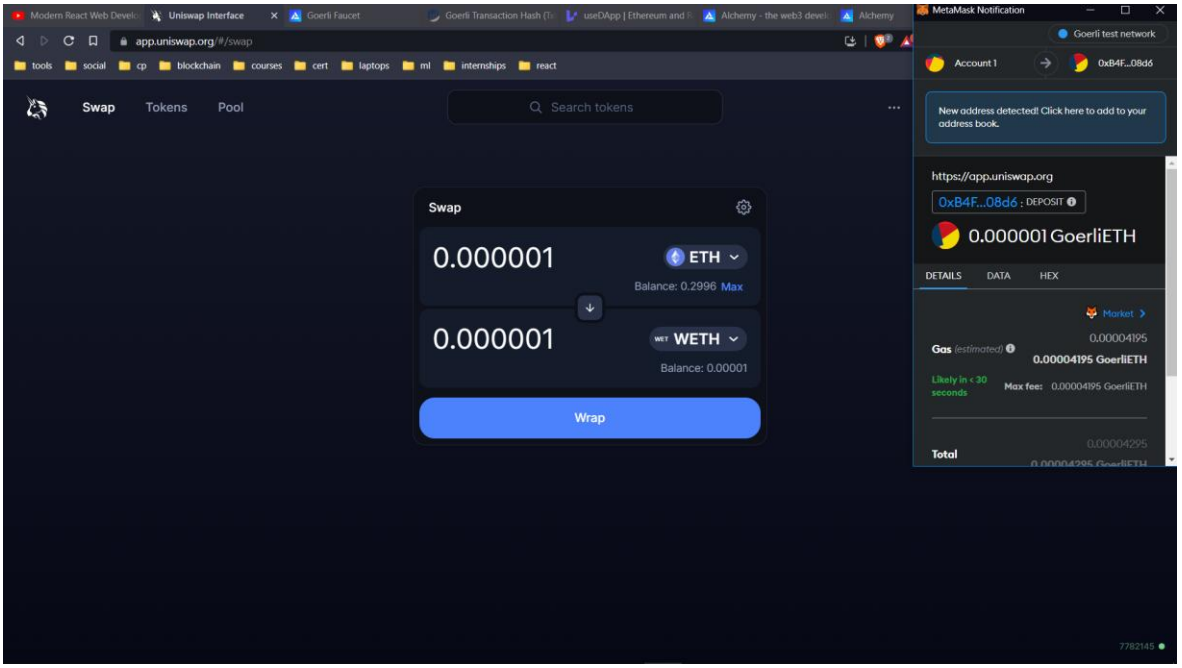

Fig 4.5.11: obtain conversion rates, sign transaction

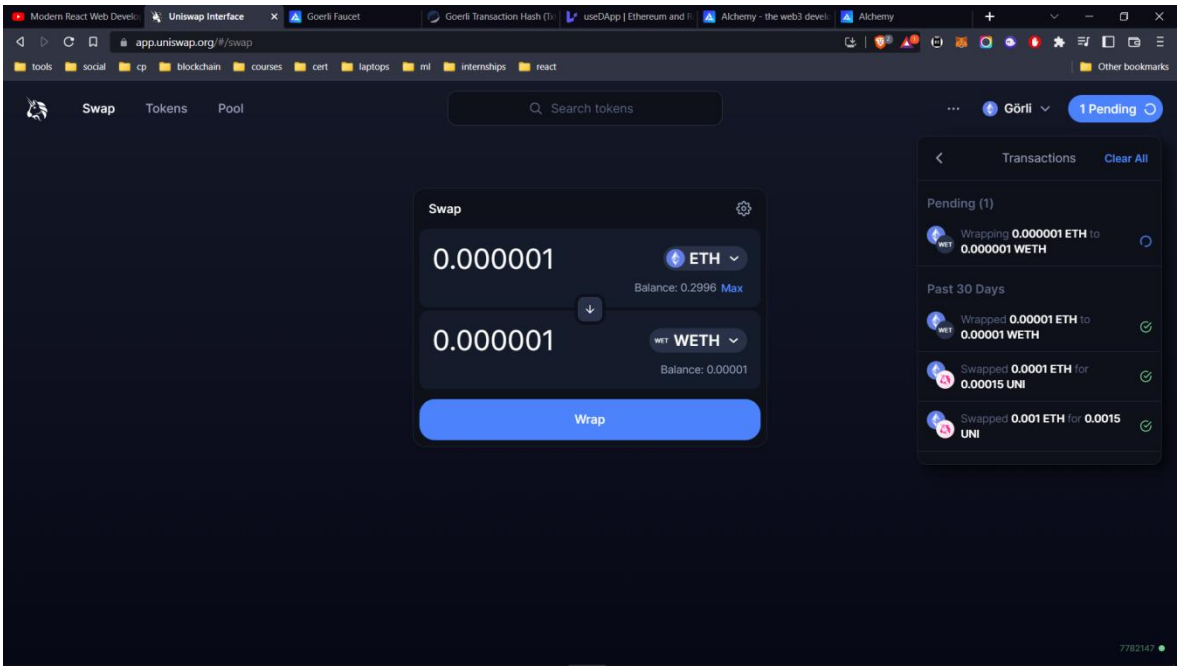

Fig 4.5.12: monitor the status of transaction

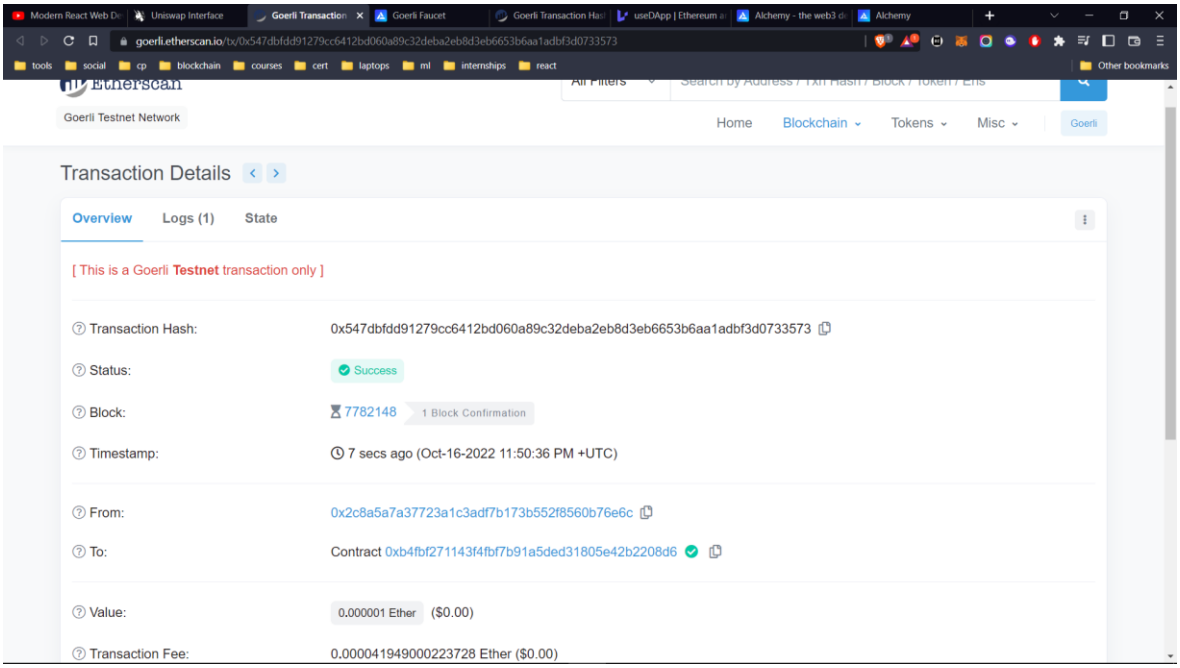

Fig 4.5.13: view the transaction on etherscan

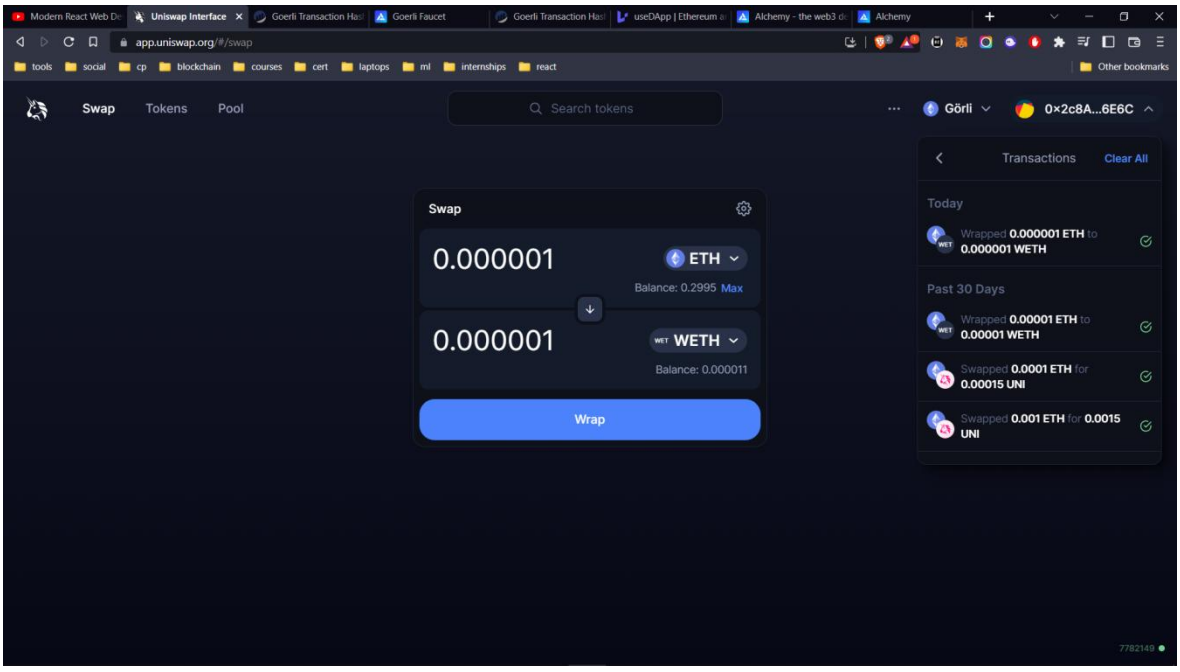

Fig 4.5.14: check balances post swap

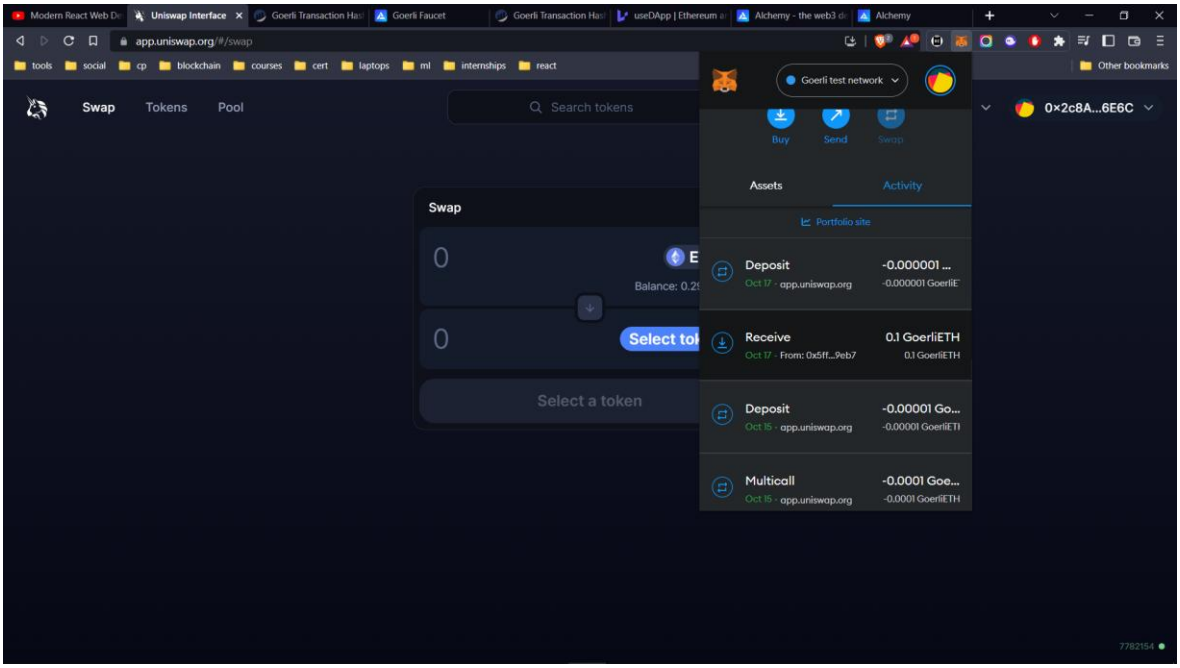

Fig 4.5.15: view transaction history

5. TESTING

Testing is a fault detection technique that tries to create failure and erroneous states in a planned way. This allows the developer to detect failures in the system before it is released to the customer. Note that this definition of testing implies that a successful test is test that identifies faults. We will use this definition throughout the definition phase. Another often used definition of testing is that it demonstrates that faults are not present. In this wok, we tested manually and haven't used automated test cases.

Testing can be done in two ways:

1. Top-down Approach.

2. Bottom-Up Approach.

Top-down approach: This type of testing starts from upper-level modules. Since the detailed activities usually performed in the lower-level routines are not provided stubs are written.

Bottom-up Approach: Testing can be performed starting from smallest and lowest level modules and proceeding one at a time. For each module in bottom up testing a short program executes the module and provides the needed data so that the module is asked to perform the way it will when embedded within the larger system.

Testing Methodologies:

The following are the Testing Methodologies:

1. Unit Testing.

2. Integration Testing.

3. User Acceptance Testing.

4. System Testing.

5. Output Testing.

1. Unit Testing:

Unit testing focuses verification effort on the smallest unit of Software design that is the module. Unit testing exercises specific paths in a modules control structure to ensure complete coverage and maximum error detection. This test focuses on each module individually, ensuring that it functions properly as a unit. Hence, the naming is Unit Testing. During this testing, each module is tested individually and the module interfaces are verified for the consistency with design specification. All-important processing paths are tested for the expected results. All error handling paths are also tested.

In this work, a lot of modules have been tested individually. One of the examples is a function for Positional encoding. We have written the code for Positional encoding and first tested it with known examples to check whether the function is working properly or not. In the same way, a lot of modules have been tested immediately after they are written.

2. Integration Testing:

Integration testing addresses the issues associated with the dual problems of verification and program construction. After the software has been integrated a set of high order tests are conducted. The main objective in this testing process is to take unit tested modules and builds a program structure that has been dictated by design.

The following are the types of Integration Testing:

i)Top-Down Integration:

This method is an incremental approach to the construction of program structure. Modules are integrated by moving downward through the control hierarchy, beginning with the main program module. The module subordinates to the main program module are in corporate into the structure in either a depth first or breadth first manner. In this method, the software is tested from main module and individual stubs are replaced when the test proceeds downwards.

ii) Bottom-Up Integration:

This method begins the construction and testing with the modules at the lowest level in the program structure. Since the modules are integrated from the bottom up, processing required for modules subordinate to a given level is always available and need

for stubs is eliminated. The bottom-up integration strategy may be implemented with the following steps:

a. low-level modules are combined into clusters that perform a specificSoftware sub-function.

b. A driver (i.e.) the control program for testing is written to coordinate test case input and output.

c. The cluster is tested. d. Drivers are removed and clusters are combined moving upward in the program structure. The bottom-up approaches test each module individually and then each module is integrated with a main module and tested for functionality.

3. User Acceptance Testing:

User acceptance of a system is the key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with the prospective system users at the time of developing and making changes whenever required. The system developed provides a friendly user interface that can easily be understood even by a person who is new to the system.

4. System Testing:

System Testing is a type of software testing that is performed on a complete integrated system to evaluate the compliance of the system with the corresponding requirements. In system testing, integration testing passed components are taken as input. The goal of integration testing is to detect any irregularity between the units that are integrated together. System testing detects defects within both the integrated units and the whole system. The result of system testing is the observed behavior of a component or a system when it is tested. System testing tests the design and behavior of the system and also the expectations of the customer. It is end- to-end testing where the testing environment is similar to the production environment.

5. Output Testing:

After performing the validation testing, the next step is output testing of the proposed system, since no system could be useful if it does not produce the required output in the specified format. Asking the users about the format required by them tests the

outputs generated or displayed by the system under consideration. Hence the output format is considered in 2 ways – one is on screen and another in printed format.

5.1 TEST CASES

Test Case Id	1
Test Scenario	Collect and install various dependencies through npm command line tool
Pre Condition	Install node and npm
Test Steps	Go through the npm website and browse for the packages
Test Data	Run npm commands in the command line
Expected Results	No Error
Post Conditon	Package.json contains all the mentioned packages
Actual Result	Successful installation
Test Status(P/F)	P

Table 5.1 : INSTALL DEPENDENCIES

Test Case Id	2
Test Scenario	Import the collected packages into the required components you are writing
Pre Condition	Install the latest versions to avoid deprecation warnings and errors
Test Steps	Write the import statements in the files you require them
Test Data	Enter the exact component / class names you require or import the entire package
Expected Results	No Error
Post Conditon	Enter other details
Actual Result	Succssful import
Test Status(P/F)	P

Table 5.2 : IMPORT PACKAGES

Test Case Id	3
Test Scenario	Connect your ethereum app to the Goerli testnet through alchemy end point
Pre Condition	Create an account and configure a connection in Alchemy
Test Steps	Obtain the API key and store it in the .env file for security
Test Data	Check the api calls to connect the application
Expected Results	No Error, connection established.
Post Conditon	Make API calls through alchemy.
Actual Result	Successful calls made from the application.
Test Status(P/F)	P

Table 5.3 : CONNECT TO THE NETWORK

Test Case Id	4
Test Scenario	Swap coins in the application and view the transaction on block explorer.
Pre Condition	Make calls through uniswap API.
Test Steps	Input the amount in the AmountIn field
Test Data	Obtain the conversion rates and output amount
Expected Results	No Error, Transaction success on the block explorer
Post Conditon	Success is shown in the transaction state in Goerli scan.
Actual Result	Transaction executed and the coins are swapped
Test Status(P/F)	P

Table 5.4 : SWAP COINS

6. SUMMARY

In this study, we propose a web application to perform the crypto currency exchanges in the decentralized way without involvement of intermediaries. By eliminating intermediaries to facilitate crypto transactions, the best DEXs provide users with an easy way to exchange their holdings.

Not all centralized exchanges have had issues. There are many currently operating that have had years of successful business operations and many happy customers. Centralized exchanges have their own negatives, however. They're forced to obey know-your-consumer (KYC) regulations of the country in which they reside, they have limits in terms of order book size and they require the user to trust the solvency of the business, something often seen as a negative by native crypto users.

DEXs are very complex smart contracts but they have simple goals: to provide liquidity to anyone who wishes to trade cryptocurrencies. The most popular DEX is Uniswap, which is built entirely on the Ethereum blockchain. Uniswap provides a decentralized trading platform for any crypto user who wishes to trade Ethereum-based tokens.

DEXs have a few significant benefits over centralized exchanges. They do not require KYC and they operate 24/7. They also provide investors with yield farming opportunities, which are opportunities to help facilitate decentralized swapping – or trading – of digital assets in exchange for a small fee. And the smart contract code (Uniswap is written in Solidity) is open and transparent, allowing crypto natives to simply verify the code instead of trusting a centralized business to be solvent.

6.1 BIBILOGRAPHY

[1] Evolution of web and internet [reference] [online] available at https://ethereum.org/evolution-of-web

[2] Evolution of web and internet [reference] [online] available at https://medium.com/history-of-web

[3] Significance of CEX and DEX and their differences [reference] [online] available at https://binance.com/dex

[4] Developments in web3 [reference] [online] available at https://wikipedia.com/web3

[5] Advantages of DEX over CEX [reference] [online] available at https://investopedia/dex

[6] Terminologies in web3 [reference] [online] available at https://coinbase.com/glossary

[7] Terminologies in web3 [reference] [online] available at https://gemini.com/glossary

[8] Fundamentals of blockchain [reference] [online] available at https://freecodecamp.org/blockchain-concepts

[9] Types of UML diagrams [reference] [online] available at https://researchgate.net/UML

[10] Types of UML diagrams [reference] [online] available at https://javatpoint.com/UML-diagrams

[11] Create your ethereum app [online] [web] available at https://usedapp.com/create-eth-app