A PROJECT STAGE-1 REPORT ON

DECENTRALIZED FINANCE APPLICATION

Submitted to

Jawaharlal Nehru Technological University, Hyderabad

in partial fulfillment of requirement for the award of the degree of

BACHELOR OF TECHNOLOGY
In
COMPUTER SCIENCE AND ENGINEERING

By

AKSHAYA CHANCHALA	**(19BD1A05C3)**
A. PEARL PRIYANKA	**(19BD1A05C4)**
CH. ASVITHA REDDY	**(19BD1A05CD)**
S. RAHUL	**(19BD1A05DL)**

Under the guidance of

MS. T. SUDHESHNA

Assistant Professor
Department of CSE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
KESHAV MEMORIAL INSTITUTE OF TECHNOLOGY
(Approved by AICTE, Affiliated to JNTUH)
Narayanaguda, Hyderabad, Telangana-29

2022-23

KESHAV MEMORIAL INSTITUTE OF TECHNOLOGY
(Accredited by NBA & NAAC, Approved By A.I.C.T.E., Reg by Govt of Telangana State & Affiliated to JNTU, Hyderabad)

DEPARTMENT OF COMPUTER SCIENCE AND ENGIEERING

CERTIFICATE

This is to certify that the project entitled **DECENTRALIZED FINANCE APPLICATION** being submitted by

AKSHAYA CHANCHALA	**(19BD1A05C3)**
A. PEARL PRIYANKA	**(19BD1A05C4)**
CH. ASVITHA REDDY	**(19BD1A05CD)**
S. RAHUL	**(19BD1A05DL)**

In partial fulfilment for the award of **Bachelor of Technology** in **Computer Science and Engineering** affiliated to the **Jawaharlal Nehru Technological University, Hyderabad** during the year 2022-23**.**

Internal Guide **Head of the Department**

(Ms. T. Sudheshna) **(Dr. S. Padmaja)**

Submitted for Viva Voce Examination held on _____

External Examiner

Unit of Keshav Memorial Educational Society
#: 3-5-1026 Narayanaguda Hyderabad 500029.
040-3261407 www.kmit.in e-mail: principal@kmit.in

Vision of KMIT

Producing quality graduates trained in the latest technologies and related tools and striving to make India a world leader in software and hardware products and services. To achieve academic excellence by imparting in depth knowledge to the students, facilitating research activities and catering to the fast growing and ever- changing industrial demands and societal needs.

Mission of KMIT

- To provide a learning environment that inculcates problem solving skills, professional, ethical responsibilities, lifelong learning through multi modal platforms and prepare students to become successful professionals.
- To establish industry institute Interaction to make students ready for the industry.
- To provide exposure to students on latest hardware and software tools.
- To promote research-based projects/activities in the emerging areas of technology convergence.
- To encourage and enable students to not merely seek jobs from the industry but also to create new enterprises.
- To induce a spirit of nationalism which will enable the student to develop, understand lndia's challenges and to encourage them to develop effective solutions.
- To support the faculty to accelerate their learning curve to deliver excellent service to students.

Vision & Mission of CSE

Vision of the CSE

To be among the region's premier teaching and research Computer Science and Engineering departments producing globally competent and socially responsible graduates in the most conducive academic environment.

Mission of the CSE

- To provide faculty with state-of-the-art facilities for continuous professional development and research, both in foundational aspects and of relevance to emerging computing trends.
- To impart skills that transform students to develop technical solutions for societal needs and inculcate entrepreneurial talents.
- To inculcate an ability in students to pursue the advancement of knowledge in various specializations of Computer Science and Engineering and make them industry-ready.
- To engage in collaborative research with academia and industry and generate adequate resources for research activities for seamless transfer of knowledge resulting in sponsored projects and consultancy.
- To cultivate responsibility through sharing of knowledge and innovative computing solutions that benefit the society-at-large.
- To collaborate with academia, industry and community to set high standards in academic excellence and in fulfilling societal responsibilities.

PROGRAM OUTCOMES (POs)

1. Engineering Knowledge: Apply the knowledge of mathematics, science, engineering fundamentals and an engineering specialization to the solution of complex engineering problems.

2. Problem analysis: Identify formulate, review research literature, and analyse complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences

3. Design/development of solutions: Design solutions for complex engineering problem and design system component or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural societal, and environmental considerations.

4. Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

5. Modern tool usage: Create select, and, apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.

6. The engineer and society: Apply reasoning informed by the contextual knowledge to societal, health, safety. legal und cultural issues and the consequent responsibilities relevant to professional engineering practice.

7. Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts and demonstrate the knowledge of, and need for sustainable development.

8. Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

9. Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams and in multidisciplinary settings.

10. Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation make effective presentations, and give and receive clear instructions.

11. Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

12. Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO1: An ability to analyse the common business functions to design and develop appropriate Computer Science solutions for social upliftment's.

PSO2: Shall have expertise on the evolving technologies like Mobile Apps, CRM, ERP, Big Data, etc.

PROGRAM EDUCATIONAL OBJECTIVES (PEOs)

PEO1: Graduates will have successful careers in computer related engineering fields or will be able to successfully pursue advanced higher education degrees.

PEO2: Graduates will try and provide solutions to challenging problems in their profession by applying computer engineering principles.

PEO3: Graduates will engage in life-long learning and professional development by rapidly adapting changing work environment.

PEO4: Graduates will communicate effectively, work collaboratively and exhibit high levels of professionalism and ethical responsibility.

PROJECT OUTCOMES

P1: To provide a friendly environment to the user

P2: Accurate results of statistics are provided

P3: Emergency contact info is provided with respect to the location of the user

P4: Government and press releases are delivered

LOW - 1
MEDIUM - 2
HIGH - 3

PROJECT OUTCOMES MAPPING PROGRAM OUTCOMES

PO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
P1			3		3	3			2			2
P2	2	2		1	2							2
P3					2			1		2		1
P4				1	2	1				1		1

PROJECT OUTCOMES MAPPING PROGRAM SPECIFIC OUTCOMES

PSO	PSO1	PSO2
P1	1	3
P2		3
P3		3
P4		3

PROJECT OUTCOMES MAPPING PROGRAM EDUCATIONALOBJECTIVES

PEO	PEO1	PEO2	PEO3	PEO4
P1	1			2
P2	1			2
P3	1			2
P4	1			2

DECLARATION

We hereby declare that the project report entitled "**DECENTRALIZED FINANCE APPLICATION"** is done in the partial fulfillment for the award of the Degree in Bachelor of Technology in Computer Science and Engineering affiliated to Jawaharlal Nehru Technological University, Hyderabad. This project has not been submitted anywhere else.

AKSHAYA CHANCHALA (19BD1A05C3)

A. PEARL PRIYANKA (19BD1A05C4)

CH. ASVITHA REDDY (19BD1A05CD)

S. RAHUL (19BD1A05DL)

ACKNOWLEDGMENT

We take this opportunity to thank all the people who have rendered their full support to our project work.

We render our thanks to **Dr. Maheshwar Dutta**, B.E., M Tech., Ph.D., Principal who encouraged us to do the Project.

We are grateful to **Mr. Neil Gogte**, Director for facilitating all the amenities required for carrying out this project.

We express our sincere gratitude to **Mr. S. Nitin**, Director and **Dr. D. Jaya Prakash**, Dean Academics for providing an excellent environment in the college.

We are also thankful to **Dr. S. Padmaja**, Head of the Department for providing us with both time and amenities to make this project a success within the given schedule.

We are also thankful to our guide **T. Sudheshna**, for her valuable guidance and encouragement given to us throughout the project work.

We would like to thank the entire CSE Department faculty, who helped us directly and indirectly in the completion of the project. We sincerely thank our friends and family for their constant motivation during the project work.

AKSHAYA CHANCHALA **(19BD1A05C3)**
A. PEARL PRIYANKA **(19BD1A05C4)**
CH. ASVITHA REDDY **(19BD1A05CD)**
S. RAHUL **(19BD1A05DL)**

CONTENT

ABSTRACT

Blockchain is a shared, distributed and permanent database that is shared among multiple nodes in a computer network. They record data in such a way that it makes it almost impossible to modify or hack the system. Blockchains also play a crucial role in cryptocurrency systems, such as Bitcoin, for maintaining a secure and Decentralized record of transactions. The goal of blockchain is to allow digital information to be recorded and distributed, but not edited. In this way, a blockchain is the foundation for immutable ledgers, or records of transactions that cannot be altered, deleted, or destroyed.

Decentralized finance (De-Fi) is an emerging financial technology based on secure distributed ledgers similar to those used by cryptocurrencies. The system removes the control banks and institutions have on money, financial products, and financial services. Decentralized applications (also known as "DApps") provide services similar to those offered by typical consumer applications, but they use blockchain technology to grant users more control over their data by eliminating the need for centralized intermediaries to manage the data, thus making the service "Decentralized." DApps built on Ethereum use blockchain technology under the hood to connect users directly. Blockchains are a way to tie together a distributed system, where each user has a copy of the records. With blockchains under the hood, users don't have to go through a third party, meaning they don't have to give up control of their data to someone else. De-Fi applications give users more control over their money through personal wallets and trading services that cater to individuals.

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

1. INTRODUCTION

1.1 PURPOSE OF THE PROJECT

Decentralized finance applications are designed to overcome these risks by distributing power and control among a network of users. This can provide a number of benefits, such as increased security, resilience, and transparency. The need for Decentralized finance applications arises from the fact that traditional financial institutions are centralized. This means that they are subject to the risks associated with centralization, such as single points of failure and the potential for corruption.

This application is a Decentralized Finance app which allows users to exchange one form of cryptocurrency into another without signing in or giving any personal details of the user. This application falls under the category of Decentralized Exchange, a critical concept under the umbrella of Decentralized finance applications, it falls under the section of Decentralized Exchange. Decentralized Exchanges, user anonymity, and fair exchange rates are the key concepts of this application. In this application, we use the concept of automated market making (AMM), whereby the decision-making process is automated. The smart contracts that are being implemented in the applications are the ones who are responsible for deciding the exchange rate for the token. The trade or transaction that is being carried out is between a user and a smart contract, and is a bilateral deal.

1.2 PROBLEMS WITH THE EXISTING SYSTEM

When it comes to cryptocurrency exchanges, there are mainly two types of exchanges, Centralized Exchange and Decentralized Exchange.

A Centralized Exchange requires you to keep your funds in the exchange's own wallet, whose keys are managed by the platform. This relieves you of the burden of managing your private keys – but at what cost? Remember: not your keys, not your coins. Letting the exchange manage the key for your wallet means whatever assets you keep in the wallet are not truly yours. Some of the most popular Centralized 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.

Decentralized Exchanges are the next generation of decentralised crypto exchanges. CEXs are susceptible to sanctions and limitations from states. Order Book DEXs use an algorithm (instead of a central platform) to find and route the trades between individual users. Automated Market MDEXs are more recent and use smart contracts to record the exchanges. Decentralized Exchanges don’t need to be KYC compliant. Instead, you simply connect the DEX to your existing wallet, using your own private keys to manage your funds. DEXs have no interaction with fiat money, so don't be asked for ID to begin using one. Your privacy is respected and your details are not porously left on the digital network.

1.3 PROPOSED SYSTEM

The main idea behind the proposed system is to allow users to exchange one form of crypto currency to another and allows a decentralized form of exchange without any authority present. The proposed system would address the following issues: The need for users to trust centralized exchanges when using decentralized exchange apps. The proposed system would provide the following benefits: A single interface for managing your decentralized exchange apps. No need to trust centralized exchanges when using decentralized exchange apps. The ability to interact with decentralized exchange apps directly from website The proposed system would be built on top of the Ethereum blockchain and would use smart contracts to facilitate transactions.

CREX is designed to be as user friendly as possible, with a simple interface that allows users to easily swap between tokens. There is no need to set up an account or login, and the process is designed to be as seamless as possible. The way CREX achieves liquidity is by allowing users to add their own liquidity to the pool. When a user wants to trade a token, they can do so directly from the pool. The trade is then settled between the user and the pool, with the pool providing the liquidity. This system is different from a traditional exchange, where trades are settled between two users. With CREX there is always someone to trade with you, as long as there is liquidity in the pool. This makes CREX a much more liquid platform than a traditional exchange.

If you're tired of going through a lengthy signup process for crypto exchanges CREX will feel like a breath of fresh air. You don't need to provide personal information or create an account. All you do is connect your crypto wallet, and you're ready to trade crypto.

Now, the fact that CREX doesn't require any personal information also has its drawbacks. You can't use it to buy crypto with fiat money, and there's the possibility of regulatory issues. But from a convenience and privacy standpoint, CREX is great.

1.4 SCOPE OF THE PROJECT

The main problem with CE is that most CEs will hold your private keys and will only allow you to buy or sell assets instead of you having control over your funds. 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.

This is application is being built on the Ethereum network, and it allows you to carry out exchanges amongst the combinations of native cryptocurrencies, ERC20 and stable coins. It solves the problem of trusting a platform with sensitive information, maintains anonymity and carry out the action of exchange of currency, the exchange rate is decided by the smart contract (which cannot be altered) thus making it authority free, easier user interface.

1.5 ARCHITECTURE DIAGRAM

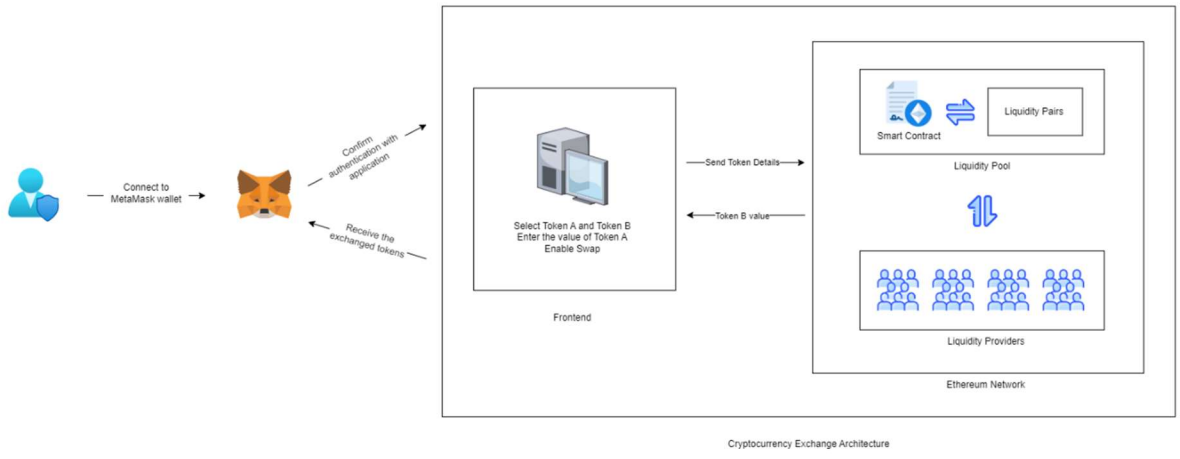


Figure 1.5. Architecture Diagram

This diagram explains the workflow of the application. Before the user is able to

available the tranasaction, they are supposed to connect their digital wallets to the application. This digital wallet will be read by CREX and any updation of tokens after the transaction will be reflected in the user account which can be seen through the wallet.

To explain the functionality of the app, let us consider the two cryptocurrency tokens that are involved in the transaction be ETH and DAI. While dicussing about the workflow we will be comparing CREX with existing Centralized Exchange applications such as Binance. Binance is a Centralized, Cryptocurrency exchange platform. Applications like Binance come under the category of what we call as Market Makers.

Market makers are essentially liquidity providers. In trading, liquidity refers to how quickly and seamlessly an asset can be bought or sold. Let us understand this with an example. Suppose trader A wants to buy one bitcoin. The centralized exchange that oversees the trade provides an automated system to find a seller, trader B, who is willing to sell a bitcoin at the rate quoted by trader A. Here, the exchange is acting as a middleman. But what if no traders are selling a bitcoin that matches trader A's buy order? In this scenario, the liquidity of the asset (in this example, bitcoin) is low. This means that there is less trading activity of the asset, and it is harder to buy or sell it.

This is where the centralised exchange needs market makers. Certain financial institutions or professional traders provide liquidity by creating multiple buy/sell orders to match the orders of retail investors. The entity that provides the liquidity becomes the market maker.

Automated market makers are a part of decentralized exchanges (DEXs) that were introduced to remove any intermediaries in the trading of crypto assets. You can think of AMM as a computer programme that automates the process of providing liquidity. These protocols are built using smart contracts -- a computer code that executes itself -- to mathematically define the price of the crypto tokens and provide liquidity.

Let us say that the user wants to exchange ETH for DAI. For this to happen there has to be enough amount of both the tokens to be present in the liquidity pool. These are provided to the pool by the liquidity providers. If there is insufficient balance of anyone of the tokens. The transaction will not be proceeded further.

In order to make the transaction happen, the user first needs to connect their wallet to

the application, and once this has been successfully done, they enter the amount of ETH that they want to exchange for DAI in the application. It is at this point that the transaction is initiated after the amount entered has been entered. If there are enough ETH-DAI pairs in the pool, then the application interacts with the smart contract to ensure that the transaction is successful. The user will be asked to confirm certain checkboxes for the exchange to take place if the answer is yes. After sometime the balance of both these tokens will be updated and this can be seen in the digital wallet.

CHAPTER 2

2. SOFTWARE REQUIREMENTS AND SPECIFICATIONS

2.1 SRS

Software Requirement Specification (SRS) is the starting point of the software developing activity. As system grew more complex it became evident that the goal of the entire system cannot be easily comprehended. Hence the need for the requirement phase arose. The software project is initiated by the client needs. The SRS is the means of translating the ideas of the minds of clients (the input) into a formal document (the output of the requirement phase.)

The SRS phase consists of two basic activities:

- Problem/Requirement Analysis: The process is order and more nebulous of the two, deals with understand the problem, the goal and constraints.
- Requirement Specification: Here, the focus is on specifying what has been found giving analysis such as representation, specification languages and tools, and checking the specifications are addressed during this activity. The Requirement phase terminates with the production of the validate SRS document. Producing the SRS document is the basic goal of this phase.

2.2 ROLE OF SRS

The purpose of the Software Requirement Specification is to reduce the communication gap between the clients and the developers. Software Requirement Specification is the medium though which the client and user needs are accurately specified. It forms the basis of software development. A good SRS should satisfy all the parties involved in the system.

2.3 REQUIREMENTS SPECIFICATION DOCUMENT

A Software Requirements Specification (SRS) is a document that describes the nature of a project, software or application. In simple words, SRS document is a manual of a project provided it is prepared before you kick-start a project/application. This document is also known by the names SRS report, software document. A software document is primarily prepared for a project, software or any kind of application.

There are a set of guidelines to be followed while preparing the software requirement specification document. This includes the purpose, scope, functional and non-functional

requirements, software and hardware requirements of the project. In addition to this, it also contains the information about environmental conditions required, safety and security requirements, software quality attributes of the project etc.

The purpose of SRS (Software Requirement Specification) document is to describe the external behavior of the application developed or software. It defines the operations, performance and interfaces and quality assurance requirement of the application or software. The complete software requirements for the system are captured by the SRS.

2.4 FUNCTIONAL REQUIREMENTS

For documenting the functional requirements, the set of functionalities supported by the system are to be specified. A function can be specified by identifying the state at which data is to be input to the system, its input data domain, the output domain, and the type of processing to be carried on the input data to obtain the output data.

FR1. User has to enter the amount of money that has to be exchanged.
FR2. User should be given a choice of tokens that can be exchanged with User's present token.
FR3. The Proposed System should be capable of carrying out the transaction without any failure.
FR4. If there are no sufficient tokens present, insufficient balance message should be displayed.

2.5 NON-FUNCTIONAL REQUIREMENTS

A non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors. Especially these are the constraints the system must work within.

Performance:

The performance of the developed applications can be calculated by using following methods: Measuring enables you to identify how the performance of your application stands in relation to your defined performance goals and helps you to identify the bottlenecks that affect your application performance. It helps you identify whether your application is moving toward or away from your performance goals. Defining what you will measure, that is, your metrics, and

defining the objectives for each metric is a critical part of your testing plan.
Performance objectives include the following: Response time, Latency throughput or Resource utilization.
NFR1. Users should be able to carry out the transaction efficiently without browser lags or hardware failure.
NFR2. For the application to work, internet is required.
NFR3. Data like User's wallet address must be protected from leaks.
NFR4. Provide accurate input.

2.6 TECHNOLOGIES USED

2.6.1 ETHEREUM NETWORK

Ethereum is a decentralized blockchain platform that establishes a peer-to-peer network that securely executes and verifies application code, called smart contracts. Smart contracts allow participants to transact with each other without a trusted central authority. Transaction records are immutable, verifiable, and securely distributed across the network, giving participants full ownership and visibility into transaction data. Ethereum offers an extremely flexible platform on which to build decentralized applications using the native Solidity scripting language and Ethereum Virtual Machine.

The Ethereum platform was launched in 2015 by Buterin and Joe Lubin, founder of the blockchain software company ConsenSys. The founders of Ethereum were among the first to consider the full potential of blockchain technology beyond just enabling the secure virtual payment method. Since the launch of Ethereum, ether as a cryptocurrency has risen to become the second-largest cryptocurrency by market value. It is outranked only by Bitcoin.

Ethereum owners use wallets to store their ether. A wallet is a digital interface that lets you access your ether stored on the blockchain. Your wallet has an address, which is similar to an email address in that it is where users send ether, much like they would an email.

Ether is not actually stored in your wallet. Your wallet holds private keys you use as you would a password when you initiate a transaction. You receive a private key for each ether you own. This key is essential for accessing your ether. That's why you hear so much about securing keys using different storage methods.

The Ethereum network currently dominates dApp development for several reasons. Ethereum implements a development interface that reduces programming time and helps quickly launch projects. Beyond this, the Ethereum developer community has grown remarkably since the platform's launch. And Ethereum retains formidable network effects from its global coalition of technologists who remain committed to maintaining the network and actively developing user resources that drive adoption. Further, the ability to adequately monetize dApp projects incentivizes others to partake in the Ethereum ecosystem.

Figure 1.6.1 - Ethereum Blockchain

2.6.2 DIGITAL WALLET

A cryptocurrency wallet is a digital or physical wallet that stores your private keys and allows you to interact with different blockchain to access your funds. Crypto wallets keep your private keys – the passwords that give you access to your cryptocurrencies – safe and accessible, allowing you to send and receive cryptocurrencies like Bitcoin and Ethereum. Keys prove your ownership of your digital money and allow you to make transactions. If you lose your private keys, you lose access to your money.

Crypto wallets range from simple-to-use apps to more complex security solutions. The main types of wallets you can choose from include:

- Paper wallets: Keys are written on a physical medium like paper and stored in a safe place. This of course makes using your crypto harder, because as digital money it can only be used on the internet.
- Hardware wallets: Keys are stored in a thumb-drive device that is kept in a safe place and only connected to a computer when you want to use your crypto. The idea is to try to balance security and convenience.
- Online wallets: Keys are stored in an app or other software – look for one that is protected by two-step encryption. This makes sending, receiving, and using your crypto

as easy as using any online bank account, payment system, or brokerage.

Each type has its tradeoffs. Paper and hardware wallets are harder for malicious users to access because they are stored offline, but they are limited in function and risk being lost or destroyed. Online wallets offered by a major exchange like Coinbase are the simplest way to get started in crypto and offer a balance of security and easy access. (Because your private info is online, your protection against hackers is only as good as your wallet provider's security – so make sure you look for features like two-factor verification.)

2.6.3 CRANQ

Graphical and intuitive IDE that allows us to compile and deploy smart contracts. It has a GUI where you can visually track how the code is being executed by looking at its flow. This way you can understand how smart contracts truly work BTS. It makes it easier for you to build on other people's code. We used CRANQ and built our application on the top of Uniswap's code base. Cranq helps us abstract the most critical functionality from these smart contracts and allow us to build on the top of them. CRANQ has drag and drop features, which allows us compile and deploy smart contracts from the Uniswap's code base easily.

2.6.4 METAMASK

MetaMask is the trailblazing tool enabling user interactions and experience on Web3. It is currently available as a browser extension and as a mobile app on both Android and iOS devices. MetaMask was created to meet the needs of secure and usable Ethereum-based web sites. In particular, it handles account management and connecting the user to the blockchain.

Advantages of MetaMask :

- Popular - It is commonly used, so users only need one plugin to access a wide range of Dapps.
- Simple - Instead of managing private keys, users just need to remember a list of words, and transactions are signed on their behalf.
- Saves space - Users don’t have to download the Ethereum blockchain, as MetaMask sends requests to nodes outside of the user’s computer.
- Integrated - Dapps are designed to work with MetaMask, so it becomes much easier to send Ether in and out.

2.6.5 USE-DAPPS

useDApp is a framework to make DApp UX & DX better, built on top of modern DApp stack: React, ethers.js, TypeScript, TypeChain, and Multicall. Lowering the entry barrier to start building DApps means more developers in the blockchain industry and higher quality apps in the ecosystem. useDApp makes existing blockchain developers' work easier & faster, and helps to bridge the gap between web2 and web3, abstracting away basic blockchain operations for new developers. Writing quality DApps is somewhat challenging.

2.6.6 NETLIFY

Netlify is a cloud platform for static websites and apps. It provides a simple, git-based workflow for deploying and managing modern web projects. Netlify Is Less Expensive, and You Get a Faster Site. Netlify Build Enables Developers to Build With Any Integration. Netlify makes it super easy for developers to host websites, in a way that is scalable and secure. The best thing about Netlify is that it selects the best CDN and distributes content. This results in pre-built websites that load faster than on traditional hosting networks. Instead of loading the site for each visit, the visitor gets a pre-loaded version straight from the closest server.

2.7 HARDWARE REQUIREMENTS

Processor	Intel core i3 or Above, AMD Ryzen 5 or Above
SSD RAM	8 GB or above
Internet	4 Mbps or above (Wireless)
Hard Disk	1 TB HDD or 256 GB
GPU	Intel Integrated Graphics or higher

2.8 SOFTWARE REQUIREMENTS

Platform	CRANQ, Brave Browser
Operating System	Windows or Mac OS
IDE	VSCode

CHAPTER 3

3. LITERATURE SURVEY

The process of summarization of the previous research works on the particular field of research. The aim of every research is deriving a new solution or invention. We can get research papers from different sources like IEEE, SPRINGER, ACM.

There are mainly five steps in literature survey they are as follows

- Collect
- Study
- Analyze
- Identify
- Summarization

TITLE OF THE SURVEY PAPER -1 : Decentralized Finance: On Blockchain - and Smart Contract-Based Financial Markets.

Author / Year : Schär, Fabian [2021]

SUMMARY OF THE PAPER :

The author show that DeFi[1] has distinct properties in terms of efficiency, transparency, accessibility and composability . Therefore, DeFi can potentially contribute to a more robust and transparent financial infrastructure.

DESCRIPTION :

This article highlights opportunities and potential risks of the DeFi ecosystem. I propose a multi-layered framework to analyze the implicit architecture and the various DeFi building blocks, including token standards, decentralized exchanges, decentralized debt markets, blockchain[2] derivatives, and on-chain asset management protocols. I conclude that DeFi still is a niche market with certain risks but that it also has interesting properties in terms of efficiency, transparency, accessibility, and composability.

DeFi consists of numerous highly interoperable protocols and application every individual can verify all transactions and data is readily available for users and researchers to analyze. On the one hand, developers are using smart contracts[3] and the decentralized settlement layer to

create trustless versions of traditional financial instruments. On the other hand, they are creating entirely new financial instruments that could not be realized without the underlying public blockchain. Smart contracts can have security issues that may allow for unintended usage, and scalability issues limit the number of users. However, if these issues can be solved, DeFi may lead to a paradigm shift in the financial industry and potentially contribute toward a more robust, open, and transparent financial infrastructure.

TITLE OF THE SURVEY PAPER - 2 : Cryptocurrencies and decentralized finance (DeFi)

Author / Year : Makarov and Schoar [2021]

SUMMARY OF THE PAPER :

The authors state that while the DeFi architecture has the potential to reduce transaction costs, similar to the traditional financial system, there are several layers where rents can accumulate due to endogenous constraints to competition. They show that the permissionless and pseudonymous design of DeFi generates challenges for enforcing tax compliance, anti-money laundering laws, and preventing financial malfeasance.

DESCRIPTION :

The paper explains how decentralized finance works and the mechanics behind it, such as the security protocols of different cryptocurrency blockchains and smart contracts (embedded computer code that automatically executes transactions when predetermined conditions are met). The authors lay out the potential benefits and challenges of the new system, including the difficulty of providing effective consumer financial protections.

TITLE OF THE PAPER - 3 : Decentralized finance: the possibilities of a blockchain

Author / Year : T Katona [2021]

SUMMARY OF THE PAPER :

The author defined DeFi as the provision of financial services in a decentralized environment enabled by blockchain technology. The author further explained that DeFi puts the data of all financial transactions in the hands of users and it is solely up to users to decide how deeply they process data in a blockchain environment.

DESCRIPTION :

With the adoption of blockchain technology, initiatives to provide financial, investment and insurance services to a wide range of users in a decentralized manner have emerged. But can decentralized finance be the alternative to the traditional financial system[4], or has it only created another "technology playground" for users who are biasedly enthusiastic about crypto-assets such as Bitcoin[5] and Ethereum[6]? The study examines the key definitions of decentralized finance and then synthesizes them to formulate a new, more complete definition. This is followed by a presentation of the different layers of decentralized finance and their prevalence, as well as an analysis of its benefits and risks.

In the conclusions, the author finds that decentralized finance has the potential to provide financial services with an open, transparent and robust infrastructure, and has the possibility of reaching a broad range of users with its basic financial services. However, this requires further development of the sector and effective management of emerging risks.

TITLE OF THE SURVEY PAPER - 4 : Do we still need financial intermediation? The case of decentralized finance – DeFi.

Author / Year : Grassi et al. [2022]

SUMMARY OF THE PAPER :

DeFi does not eliminate financial intermediation; rather, DeFi enables financial intermediation to be performed in new ways.

DESCRIPTION :

Decentralized finance (DeFi), enabled by blockchain, could bring about a new financial system, where peers will interact directly, with little or no place for traditional intermediation. However, some crucial tasks cannot be left solely to an algorithm and, consequently, most DeFi applications still require human decisions. The aim of this research is to assess the role of intermediation in the light of DeFi, analyzing how humans and algorithms will interact.

DeFi does not eliminate financial intermediation but enables it to be performed in new ways, where decentralization means that no single entity can hold too much power or monopoly. DeFi has, however, inherited risks from the underlying technologies that unintentionally facilitate

illegal behavior and can hamper the authorities supervision. The complex duality algorithm- vs human-based actions will not be solved indisputably in favor of the former, as DeFi solutions can range from requiring algorithms to play a dominant role, to enabling greater human interaction by actively involving more people.

TITLE OF THE SURVEY PAPER – 5 : A Functional Perspective of Financial Intermediation

Author / Year : Merton [1995]

SUMMARY OF THE PAPER :

By engaging in producing and spreading information, brokering, monitoring, providing liquidity, pooling savings, risk management and other operations, financial intermediation was able to reduce transaction costs and information asymmetry.

DESCRIPTION :

New financial product designs, improved computer and telecommunications technology, and advances in the theory of finance have led to dramatic and rapid changes in the structure of global financial markets and institutions. This paper offers a functional perspective as the conceptual framework for analyzing the dynamics of institutional changes in financial intermediation and uses a series of examples to illustrate the range of institutional change that is likely to occur. These examples are used to frame the managerial issues surrounding the production process for intermediaries and to discuss the regulatory process for those intermediaries.

TITLE OF THE SURVEY PAPER – 6 : The blockchain oracle problem in decentralized finance - a multivocal approach

Author / Year : Chen and Bellavitis [2019]

SUMMARY OF THE PAPER :

They find that blockchain technology can reduce transaction costs, expand transaction scope, and empower peer-to-peer transactions, which can create a new paradigm for decentralized business models.

DESCRIPTION :

Decentralized Finance (DeFi) takes the promise of blockchain a step further and aims to transform traditional financial products into trustless and transparent protocols that run without involving intermediaries. The decentralized platforms utilize oracles to retrieve asset data from the external world, but their choice and management criteria are often unknown to the .end-users. If oracles are poorly selected or managed, the funds of a rising number of investors are inevitably in danger. The issue, known as "the oracle problem", which makes real-world applications controversial and debated due to the loss of decentralization, had recently drawn attention to DeFi, given the crescent number of related hacks that caused the loss of millions of dollars held in DeFi projects. Through a multivocal approach that considers academic papers, whitepapers, preprints, and opinion posts, this study aims to shed light on the pattern that identifies the oracle problem in DeFi and outline the most promising ways to overcome the related weaknesses. This research supports the view that the oracle problem in decentralized finance bears specific characteristics which require standardization and appropriate economic incentives to be addressed.

TITLE OF THE SURVEY PAPER – 7 : An open protocol for decentralized exchange on the Ethereum blockchain

Author / Year : Will Warren, Amir Bandeali, 2017

SUMMARY OF THE SURVEY :

- Publicly accessible smart contracts that any dApp[7] can hook into.
- Relayers can create their own liquidity pools and charge transaction fees on volume.

- Standardization + decoupling = Shared protocol layer → – provides interoperability between dApps – creates network effects around liquidity that are mutually beneficial – reduces barriers-to-entry, driving down costs for market participants – eliminates redundancy, improves user experience and smart contract security
- Decentralized update mechanism allows improvements to be continuously and safely integrated into the protocol without disrupting dApps or end users.

DESCRIPTION :

Describes a protocol that facilitates low friction peer-to-peer exchange of ERC20[8] tokens on the Ethereum blockchain. The protocol is intended to serve as an open standard and common building block, driving interoperability among decentralized applications (dApps) that incorporate exchange functionality. Trades are executed by a system of Ethereum smart contracts that are publicly accessible, free to use and that any dApp can hook into. DApps built on top of the protocol can access public liquidity pools or create their own liquidity pool and charge transaction fees on the resulting volume. The protocol is unopinionated: it does not impose costs on its users or arbitrarily extract value from one group of users to benefit another. Decentralized governance[9] is used to continuously and securely integrate updates into the base protocol without disrupting dApps or end users. These types reflect AMM Model[10].

CHAPTER 4

4. SYSTEM DESIGN

4.1 INTRODUCTION TO UML

The Unified Modeling Language allows the software engineer to express an analysis model using the modeling notation that is governed by a set of syntactic, semantic and pragmatic rules. A UML system is represented using five different views that describe the system from distinctly different perspective. Each view is defined by a set of diagram, which is as follows:

- User Model View

  This view represents the system from the users' perspective. The analysis representation describes a usage scenario from the end-users' perspective.
- Structural Model View

  In this model, the data and functionality are arrived from inside the system. This model view models the static structures.
- Behavioral Model View

  It represents the dynamic of behavioral as parts of the system, depicting he interactions of collection between various structural elements described in the user model and structural model view.
- Implementation Model View

  In this view, the structural and behavioral as parts of the system are represented as they are to be built.
- Environmental Model View

  In this view, the structural and behavioral aspects of the environment in which the system is to be implemented are represented.

4.2.1 USE CASE DIAGRAMS

To model a system, the most important aspect is to capture the dynamic behaviour. To clarify a bit in details, dynamic behaviour means the behaviour of the system when it is running/operating.

So only static behaviour is not sufficient to model a system rather dynamic behaviour

is more important than static behaviour. In UML there are five diagrams available to model dynamic nature and use case diagram is one of them. Now as we have to discuss that the use case diagram is dynamic in nature there should be some internal or external factors for making the interaction.

These internal and external agents are known as actors. So, use case diagrams are consisting of actors, use cases and their relationships. The diagram is used to model the system/subsystem of an application. A single use case diagram captures a particular functionality of a system. So, to model the entire system numbers of use case diagrams are used. Use case diagrams are used to gather the requirements of a system including internal and external influences. These requirements are mostly design requirements. So, when a system is analyzed to gather its functionalities use cases are prepared and actors are identified. In brief, the purposes of use case diagrams can be as follows:

- Used to gather requirements of a system.
- Used to get an outside view of a system.
- Identify external and internal factors influencing the system.
- Show the interacting among the requirements are actors.

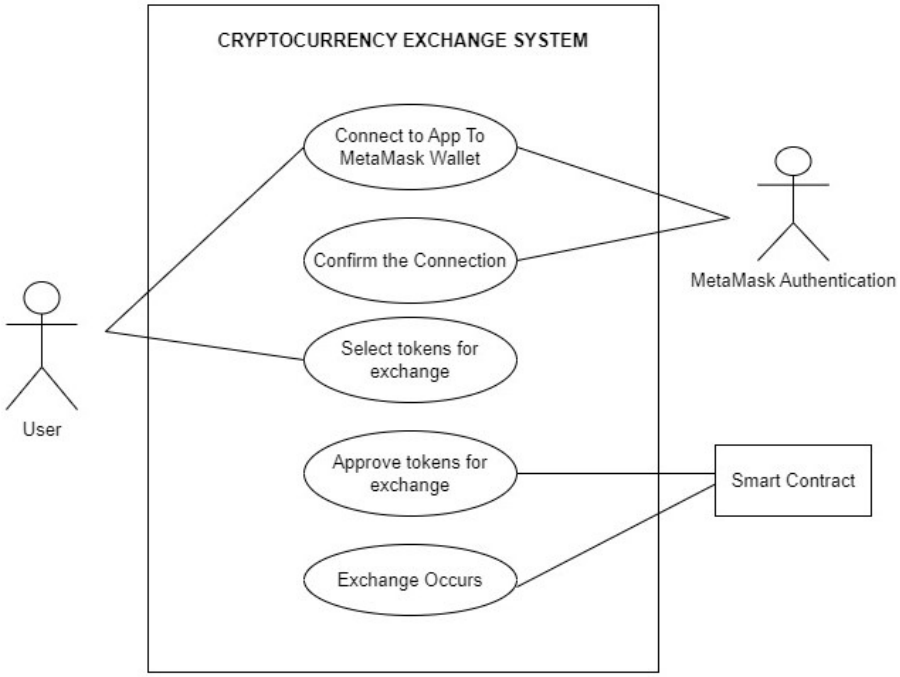


Figure 4.2.1 - Use Case Diagram

4.2.2 SEQUENCE DIAGRAM

Sequence diagrams describe interactions among classes in terms of an exchange of messages over time. They're also called event diagrams. A sequence diagram is a good way to visualize and validate various runtime scenarios. These can help to predict how a system will behave and to discover responsibilities a class may need to have in the process of modelling a new system.

The aim of a sequence diagram is to define event sequences, which would have a desired outcome. The focus is more on the order in which messages occur than on the message per se. However, the majority of sequence diagrams will communicate what messages are sent and the order in which they tend to occur.

Basic Sequence Diagram Notations

- **Class Roles or Participants**
  Class roles describe the way an object will behave in context. Use the UML object symbol to illustrate class roles, but don't list object attributes.
- **Activation or Execution Occurrence**
  Activation boxes represent the time an object needs to complete a task. When an object is busy executing a process or waiting for a reply message, use a thin grey rectangle placed vertically on its lifeline.
- **Messages**
  Messages are arrows that represent communication between objects. Use half-arrowed lines to represent asynchronous messages. Asynchronous messages are sent from an object that will not wait for a response from the receiver before continuing its tasks.
- **Lifelines**
  Lifelines are vertical dashed lines that indicate the object's presence over time.

Destroying Objects

Objects can be terminated early using an arrow labelled "<< destroy >>" that points to an X. This object is removed from memory. When that object's lifeline ends, you can place an X at the end of its lifeline to denote a destruction occurrence.

Loops

A repetition or loop within a sequence diagram is depicted as a rectangle. Place the condition for exiting the loop at the bottom left corner in square brackets []. When modelling object interactions, there will be times when a condition must be met for a message to be sent to an object. Guards are conditions that need to be used throughout UML diagrams to control flow.

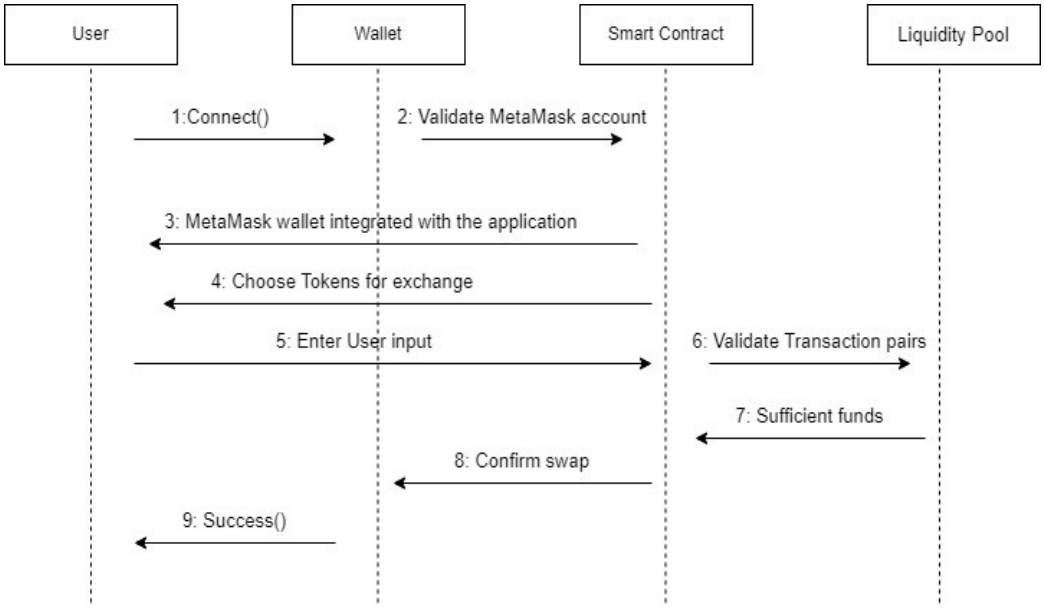


Figure 4.2.2 - Sequence Diagram

4.2.3 ACTIVITY DIAGRAM

Activity Diagrams describe how activities are coordinated to provide a service which can be at different levels of abstraction. Typically, an event needs to be achieved by some operations, particularly where the operation is intended to achieve a number of different things that require coordination, or how the events in a single use case relate to one another, in particular, use cases where activities may overlap and require coordination. It is also suitable for modelling how a collection of use cases coordinates to represent business workflows

- Identify candidate use cases, through the examination of business workflows
- Identify pre- and post-conditions (the context) for use cases
- Model workflows between/within use cases

- Model complex workflows in operations on objects
- Model in detail complex activities in a high-level activity Diagram

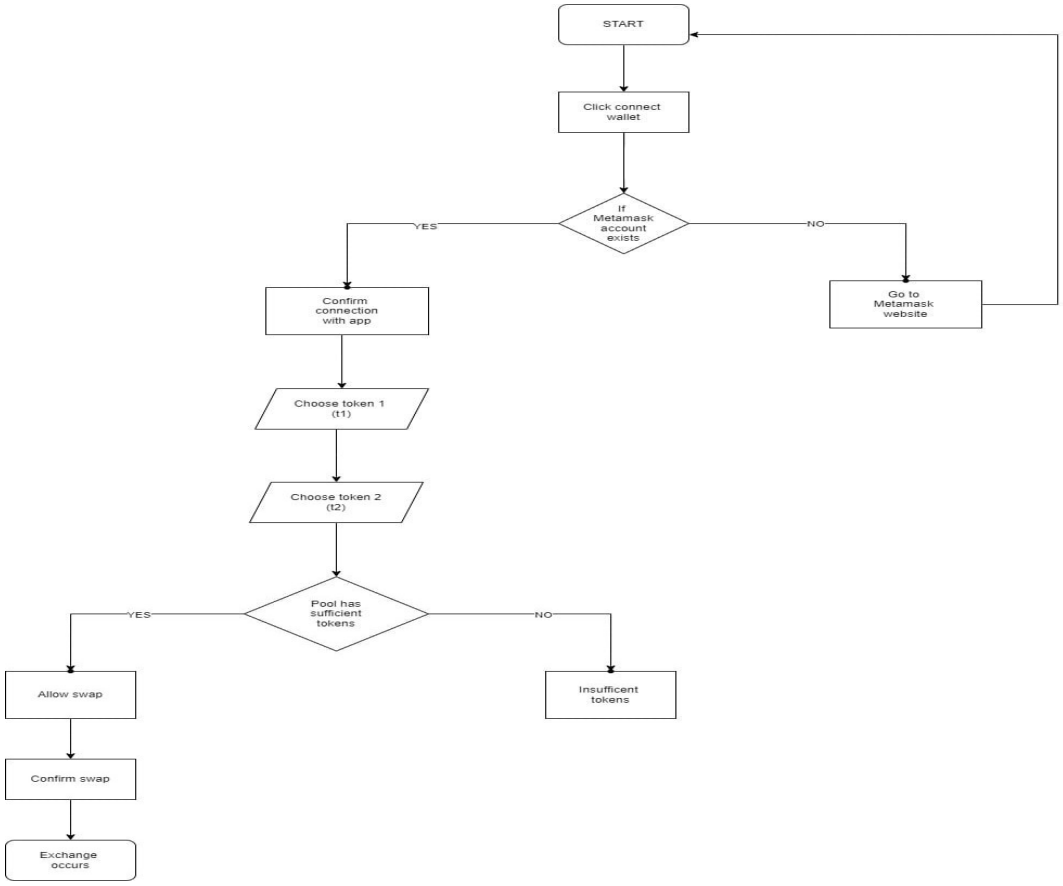


Figure 4.2.3 - Activity Diagram

4.2.4 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.

Purpose of Class Diagrams

The purpose of class diagram is to model the static view of an application. Class

diagrams are the only diagrams which can be directly mapped with object-oriented languages and thus widely used at the time of construction.

UML diagrams like activity diagram, sequence diagram can only give the sequence flow of the application, however class diagram is a bit different. It is the most popular UML diagram in the coder community.

The purpose of the class diagram can be summarized as −

- Analysis and design of the static view of an application.
- Describe responsibilities of a system.
- Base for component and deployment diagrams.
- Forward and reverse engineering.

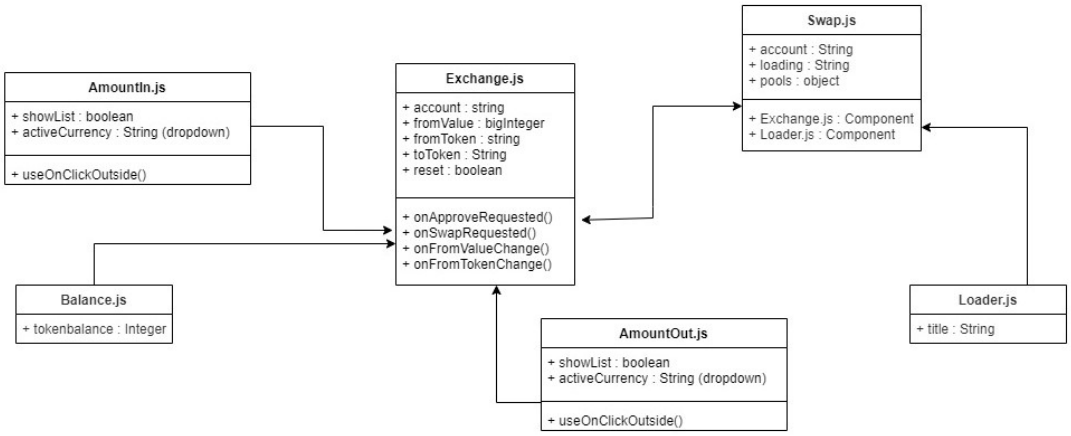


Figure 4.2.4 - Class Diagram

CHAPTER 5

5. IMPLEMENTATION

5.1 Exchange.js

```
const Exchange = ({ pools }) => {

 const { state: swapApproveState, send: swapApproveSend } =
useContractFunction(fromTokenContract, "approve", {
        transactionName: "onApproveRequested",
        gasLimitBufferPercentage: 10
    })

    const { state: swapExecuteState, send: swapExecuteSend } =
useContractFunction(routerContract, "swapExactTokensForTokens", {
        transactionName: "swapExactTokensForTokens",
        gasLimitBufferPercentage: 10
    })

    const isApproving = isOperationPending(swapApproveState)
    const isSwapping = isOperationPending(swapExecuteState)
    const canApprove = !isApproving && approvedNeeded
    const canSwap = !approvedNeeded && !isSwapping && fromValueIsGreaterThan0
&& hasEnoughBalace

    const successMessage = getSuccessMessage(swapApproveState,
swapExecuteState)
    const failureMessage = getFailureMessage(swapApproveState,
swapExecuteState)

    const onApproveRequested = () => {
        swapApproveSend(ROUTER_ADDRESS, ethers.constants.MaxUint256)
    }

    const onSwapRequested = () => {
        swapExecuteSend(fromValueBigNumber, 0, [fromToken, toToken], account,
Math.floor(Date.now() / 1000 + 60 * 2))
            .then(() => {
                setFromValue("0")
            })
    }

    const onFromValueChange = (value) => {
        const trimmedValue = value.trim()
        try {
            if (trimmedValue) {
                parseUnits(value)
                setFromValue(value)
```

```
            }
        } catch (error) {
            console.log(error)
        }
    }

    const onFromTokenChange = (value) => {
        setFromToken(value)
    }

    const onToTokenChange = (value) => {
        setToToken(value)
    }

    useEffect(() => {
        if (failureMessage || successMessage) {
            setTimeout(() => {
                setResetState(true)
                setFromValue("0")
                setToToken("")
            }, 5000)
        }
    }, [failureMessage, successMessage])

    return (
        <div className='flex flex-col w-full items-center'>
            <div className='mb-8'>
                <AmountIn
                    value={fromValue}
                    onChange={onFromValueChange}
                    currencyValue={fromToken}
                    onSelect={onFromTokenChange}
                    currencies={availableTokens}
                    isSwapping={isSwapping && hasEnoughBalace}
                />
                <Balance tokenBalance={fromTokenBalance} />
            </div>
            <div className='mb-8 w-[100%]'>
                <AmountOut
                    fromToken={fromToken}
                    toToken={toToken}
                    amountIn={fromValueBigNumber}
                    pairContract={pairAddress}
                    currencyValue={toToken}
                    onSelect={onToTokenChange}
                    currencies={counterpartTokens}
                />
                <Balance tokenBalance={toTokenBalance} />
```

```
            </div>
            {approvedNeeded && !isSwapping ? (
                <button
                    disabled={!canApprove}
                    onClick={onApproveRequested}
                    className={
                        `${canApprove ? "bg-site-pink text-white" : "bg-site-
dim2 text-site-dim2"
                        }
                        ${styles.actionButton}`
                    }
                >
                    {isApproving ? "Approving..." : "Approve"}
                </button>
            ) : <button
                disabled={!canSwap}
                onClick={onSwapRequested}
                className={
                    `${canSwap ? "bg-site-pink text-white" : "bg-site-dim2
text-site-dim2"
                    }
                        ${styles.actionButton}`
                }
            >
                {isSwapping ? "Swapping..." : hasEnoughBalace ? "Swap" :
"Insufficient Balace"}
            </button>
            }
            {
                failureMessage && !resetState ? AmountIn(
                    <p className={styles.message}>{failureMessage}</p>
                ) : successMessage ? (
                    <p className={styles.message}>{successMessage}</p>
                ) : ""
            }
        </div>
    )
}

export default Exchange
```

5.2 AmountIn.js

```
const AmountIn = ({ value, onChange, currencyValue, onSelect, currencies,
isSwapping }) => {
    const [showList, setShowList] = useState(false)
    const [activeCurrency, setActiveCurrency] = useState("Select")
```

```
    const ref = useRef()

    useOnClickOutside(ref, () => setShowList(false))

    useEffect(() => {
        if (Object.keys(currencies).includes(currencyValue)) {
            setActiveCurrency(currencies[currencyValue])
        }
        else {
            setActiveCurrency("Select")
        }
    }, [currencies, currencyValue])

    return (
        <div className={styles.amountContainer}>
            <input
                placeholder='0'
                type="number"
                value={value}
                disabled={isSwapping}
                onChange={(e) => typeof onChange === "function" &&
onChange(e.target.value)}
                className={styles.amountInput}
            />
            <div className='relative' onClick={() => { setShowList((prevState)
=> !prevState) }}>
                <button className={styles.currencyButton}>
                    {activeCurrency}
                    <img
                        src={chevronDown}
                        alt="chevronDown"
                        className={`w-4 h-4 object-contain ml-2 ${showList ?
'rotate-180' : 'rotate-0'}`}
                    />
                </button>
                {showList && (
                    <ul ref={ref} className={styles.currencyList}>
                        {Object.entries(currencies).map(([token, tokenName],
index) => (
                            <li
                                key={index}
                                className={`${styles.currencyListItem}
${activeCurrency === tokenName ? 'bg-site-dim2' : ''} cursor-pointer`}
                                onClick={() => {
                                    if (typeof onSelect === "function")
onSelect(token)
                                    setActiveCurrency(tokenName)
                                    setShowList(false)
```

```
                                }}
                            >
                                {tokenName}
                            </li>
                        ))
                        }
                    </ul>
                )}
            </div>
        </div>
    )
}

export default AmountIn
```

5.3 AmountOut.js

```
const AmountOut = ({ fromToken, amountIn, toToken, currencyValue,
pairContract, currencies, onSelect
}) => {
    const [showList, setShowList] = useState(false)
    const [activeCurrency, setActiveCurrency] = useState("select")
    const ref = useRef()

    const amountOut = useAmountsOut(pairContract, amountIn, fromToken,
toToken) ?? 0

    useEffect(() => {
        if (Object.keys(currencies).includes(currencyValue)) {
            setActiveCurrency(currencies[currencyValue])
        }
        else {
            setActiveCurrency("Select")
        }
    }, [currencies, currencyValue])

    useOnClickOutside(ref, () => setShowList(false))

export default AmountOut
```

CHAPTER 6

6. TESTING

6.1 INTRODUCTION TO TESTING

Testing is the process of evaluating a system or its component(s) with the intent to find whether it satisfies the specified requirements or not. Testing is executing a system in order to identify any gaps, errors, or missing requirements in contrary to the actual requirements.

According to ANSI/IEEE 1059 standard, Testing can be defined as - A process of analyzing a software item to detect the differences between existing and required conditions (that is defects/errors/bugs) and to evaluate the features of the software item.

Who does Testing?

It depends on the process and the associated stakeholders of the project(s). In the IT industry, large companies have a team with responsibilities to evaluate the developed software in context of the given requirements. Moreover, developers also conduct testing which is called Unit Testing.

In most cases, the following professionals are involved in testing a system within their respective capacities:

- Software Tester
- Software Developer
- Project Lead/Manager
- End User

Levels of testing include different methodologies that can be used while conducting software testing. The main levels of software testing are:

- Functional Testing
- Non-functional Testing

Functional Testing

This is a type of black-box testing that is based on the specifications of the software that is to be tested. The application is tested by providing input and then the results are examined that need to conform to the functionality it was intended for. Functional testing of a software is conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements.

Software Testing Life Cycle

The process of testing a software in a well-planned and systematic way is known as software testing lifecycle (STLC). Different organizations have different phases in STLC however generic Software Test Life Cycle (STLC) for waterfall development model consists of the following phases.

- Requirements Analysis
- Test Planning
- Test Analysis
- Test Design

Requirements Analysis

In this phase testers analyze the customer requirements and work with developers during the design phase to see which requirements are testable and how they are going to test those requirements.

Test Planning

In this phase all the planning about testing is done like what needs to be tested, how the testing will be done, test strategy to be followed, what will be the test environment, what test methodologies will be followed, hardware and software availability, resources, risks etc. A high-level test plan document is created which includes all the planning inputs mentioned above and circulated to the stakeholders.

Test Analysis

After test planning phase is over test analysis phase starts, in this phase we need to dig

deeper into project and figure out what testing needs to be carried out in each SDLC phase. Automation activities are also decided in this phase, if automation needs to be done for software product, how will the automation be done, how much time will it take to automate and which features need to be automated. Non-functional testing areas (Stress and performance testing) are also analyzed and defined in this phase.

Test Design

In this phase various black-box and white-box test design techniques are used to design the test cases for testing, testers start writing test cases by following those design techniques, if automation testing needs to be done then automation scripts also needs to written in this phase.

6.2 TEST CASES

6.2.1 - Launching Home Page

Test Scenario ID	Home Page	**Test Case ID**	Home Page
Test Case Description	Launching Home Page	**Test Priority**	High
Pre-Requisite	NA	**Post Requisite**	NA

Test Execution Steps :

No	**Action**	**Inputs**	**Expected Output**	**Actual Output**	**Test Browser**	**Test Result**	**Test Comments**
1	Open Home Page	https://crex-exchange.netlify.app/	Open Home Page on using the URL	Open Home Page on using the URL	Brave	Pass	NA

6.2.2 - Launching Exchange App Page

Test Scenario ID	Swap Page		**Test Case ID**		Swap Page	
Test Case Description	Launching Exchange Page		**Test Priority**		High	
Pre-Requisite	NA		**Post Requisite**		NA	
Test Execution Steps :						
No	**Action**	**Inputs**	**Expected Output**	**Actual Output**	**Test Browser**	**Test Result**	**Test Comments**
1	Open Swap Page	https://crex-exchange.netlify.app/swap	Open Swap Page on Clicking the Launch Button	Open Swap Page on Clicking the Launch Button	Brave	Pass	NA

6.2.3 - Connect MetaMask Wallet

Test Scenario ID	Connect MetaMask Wallet		**Test Case ID**		Connect Wallet	
Test Case Description	Connecting MetaMask Wallet with Application		**Test Priority**		High	
Pre-Requisite	NA		**Post Requisite**		NA	
Test Execution Steps :						
No	**Action**	**Inputs**	**Expected Output**	**Actual Output**	**Test Browser**	**Test Result**	**Test Comments**

1	Connect wallet	https://crex-exchange.netlify.app/swap	Wallet Connected Successfully	Wallet Connected Successfully	Brave	Pass	NA

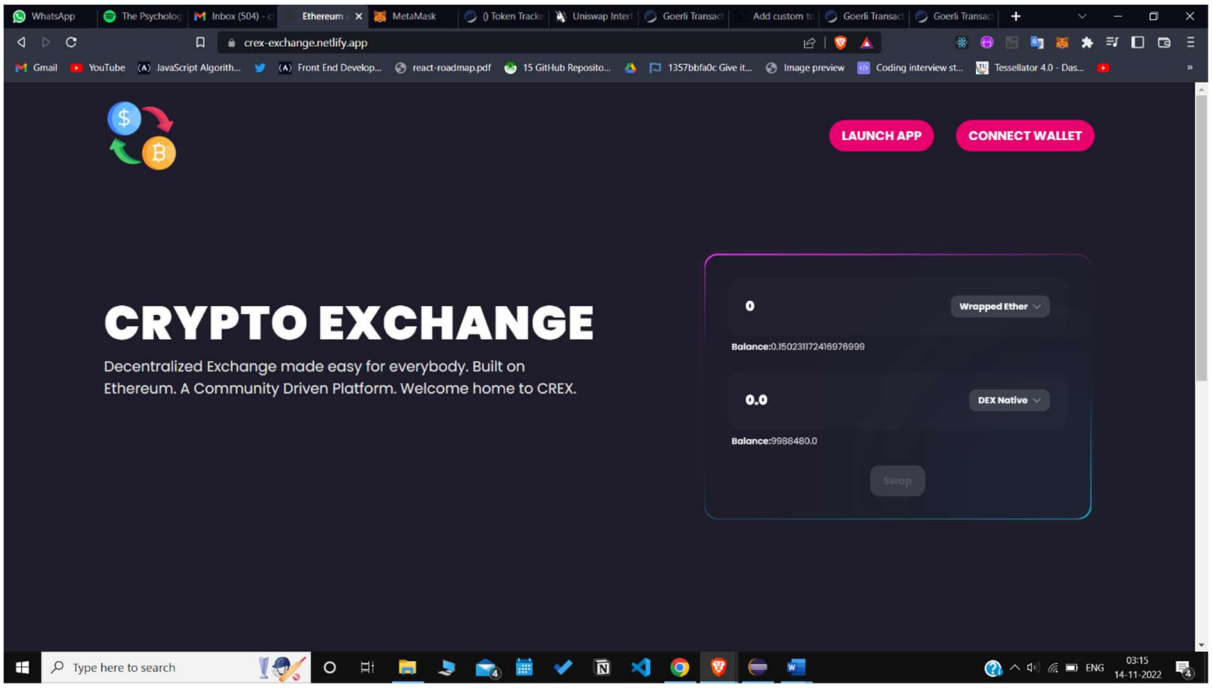
6.2.4 - Perform Exchange

Test Scenario ID	Exchange		**Test Case ID**	Exchange		
Test Case Description	Enter currency to be exchanged		**Test Priority**	High		
Pre-Requisite	NA		**Post Requisite**	NA		
Test Execution Steps :						
No	**Action**	**Inputs**	**Expected Output**	**Actual Output**	**Test Browser**	**Test Result**	**Test Comments**
1	Perform Exchange Action	https://crex-exchange.netlify.app/swap	Select the token, and enter amount to be exchanged	Select the token, and enter amount to be exchanged	Brave	Pass	NA

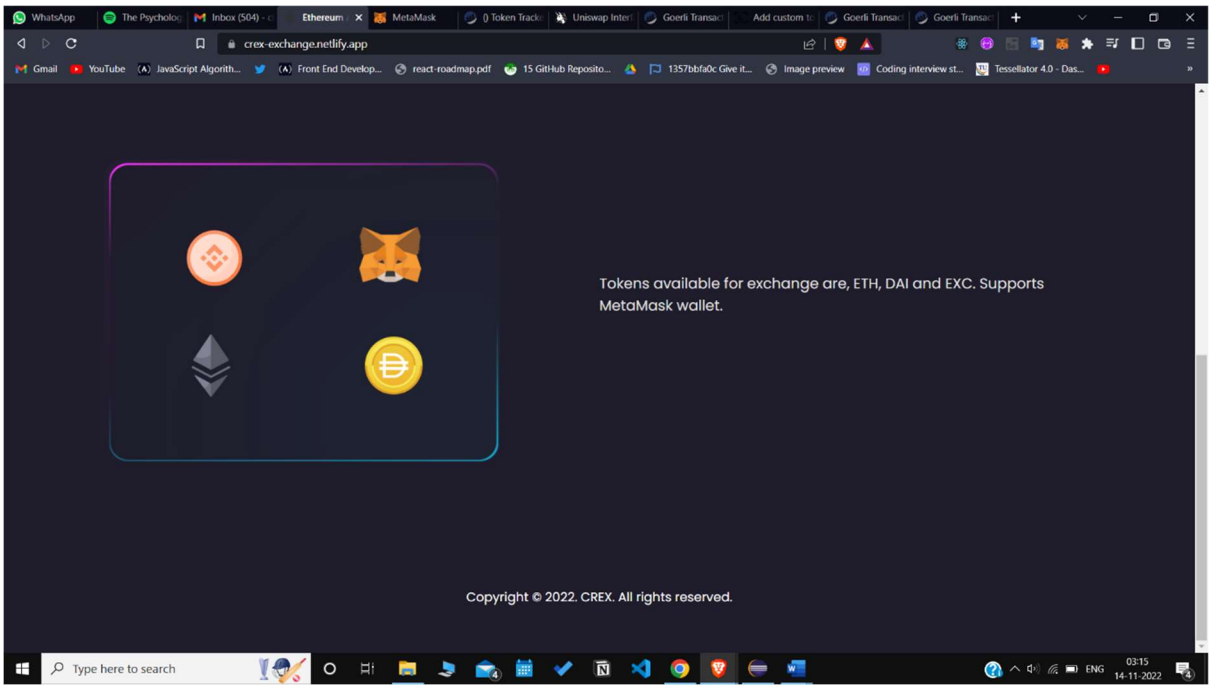
CHAPTER 7

7. SCREENSHOTS

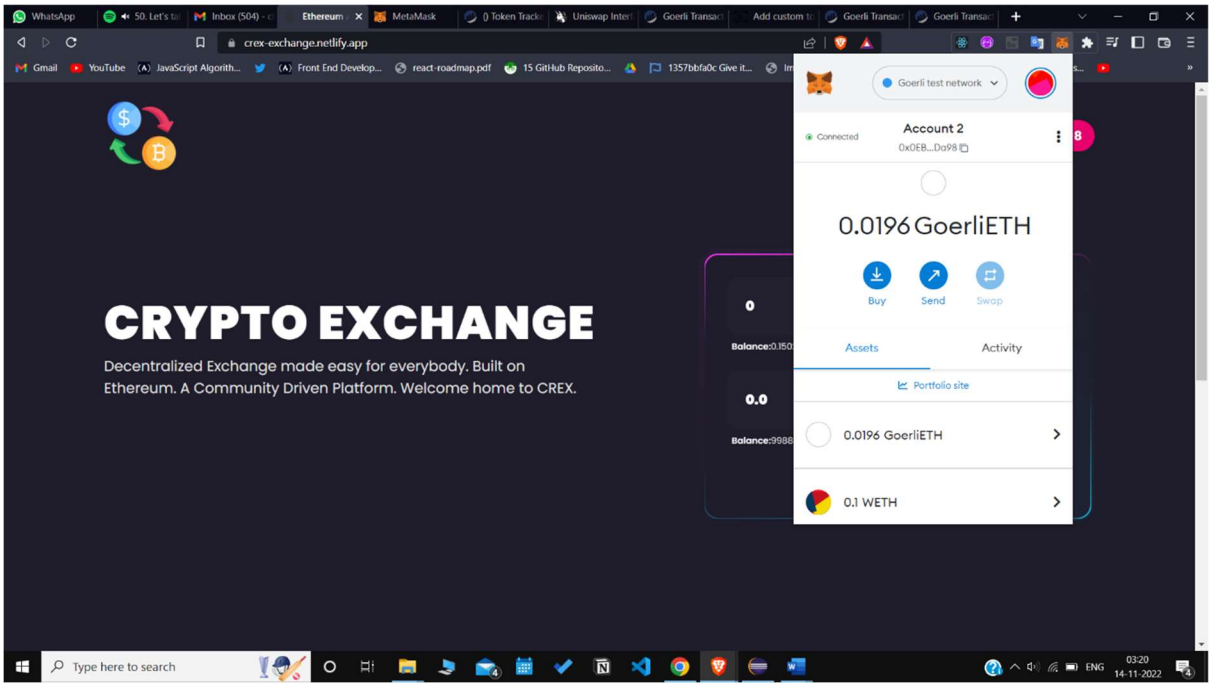
7.1- HOME PAGE - 1



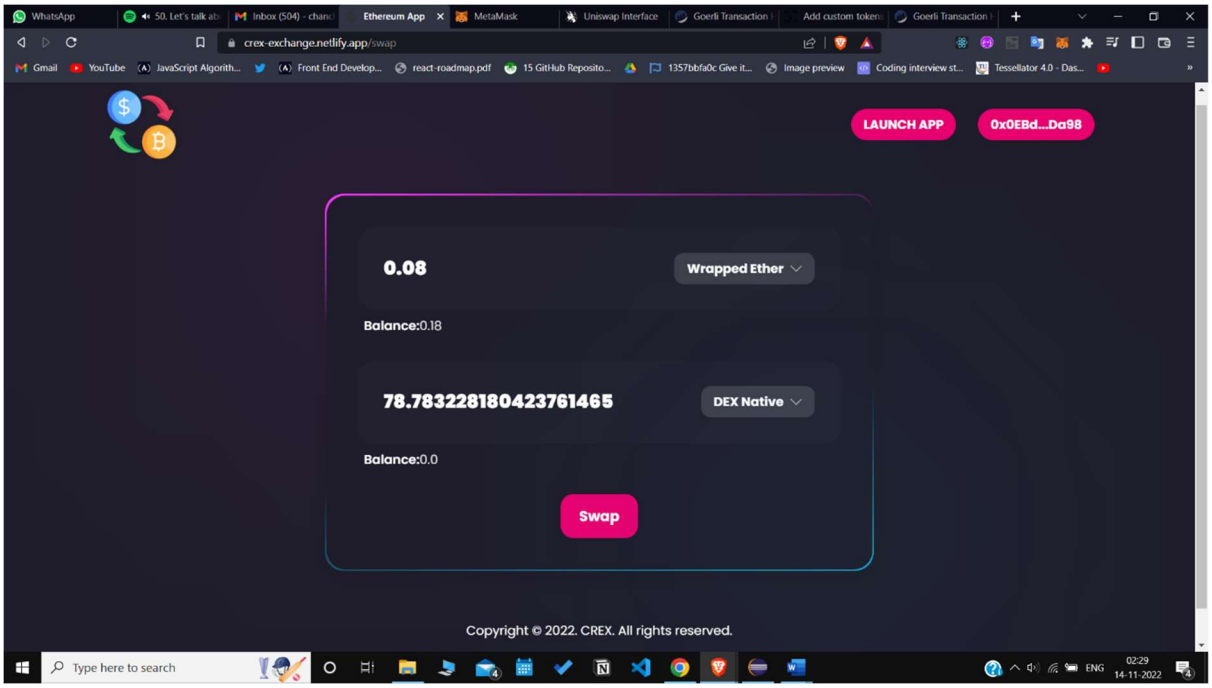
7.2 - HOME PAGE – 2



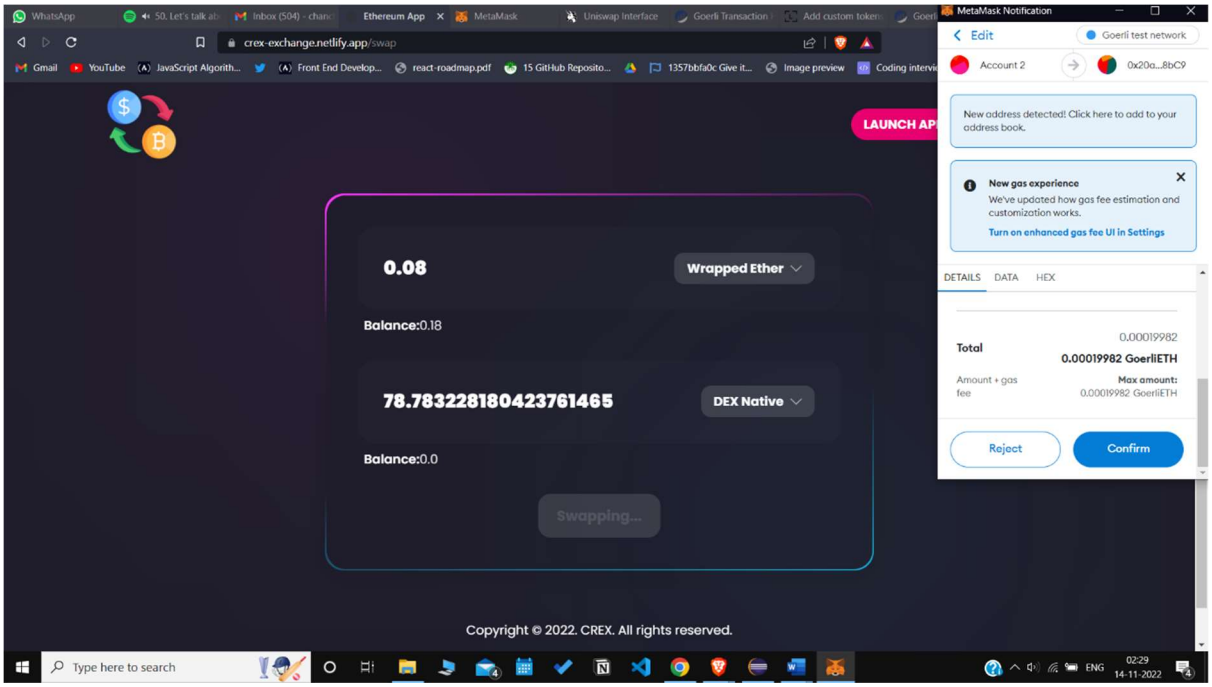
7.3 – CONNECT WALLET



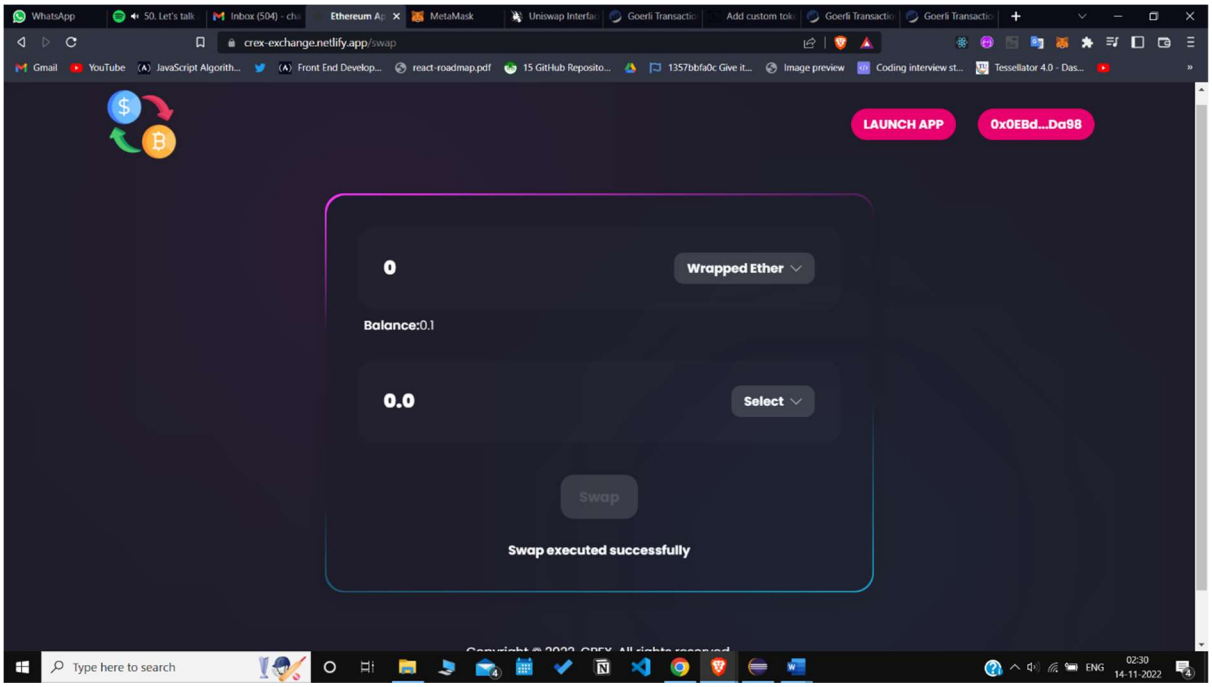
7.4 – ENTER TRANSACTION TOKEN DETAILS



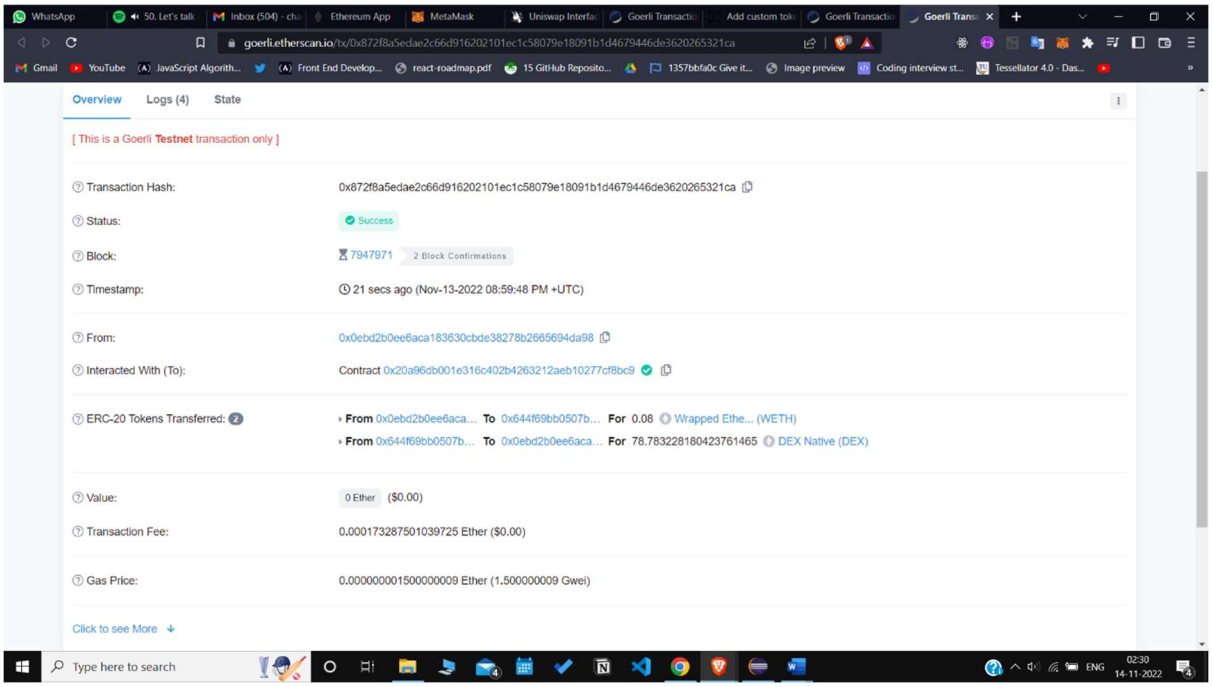
7.5 CONFIRM EXCHANGE



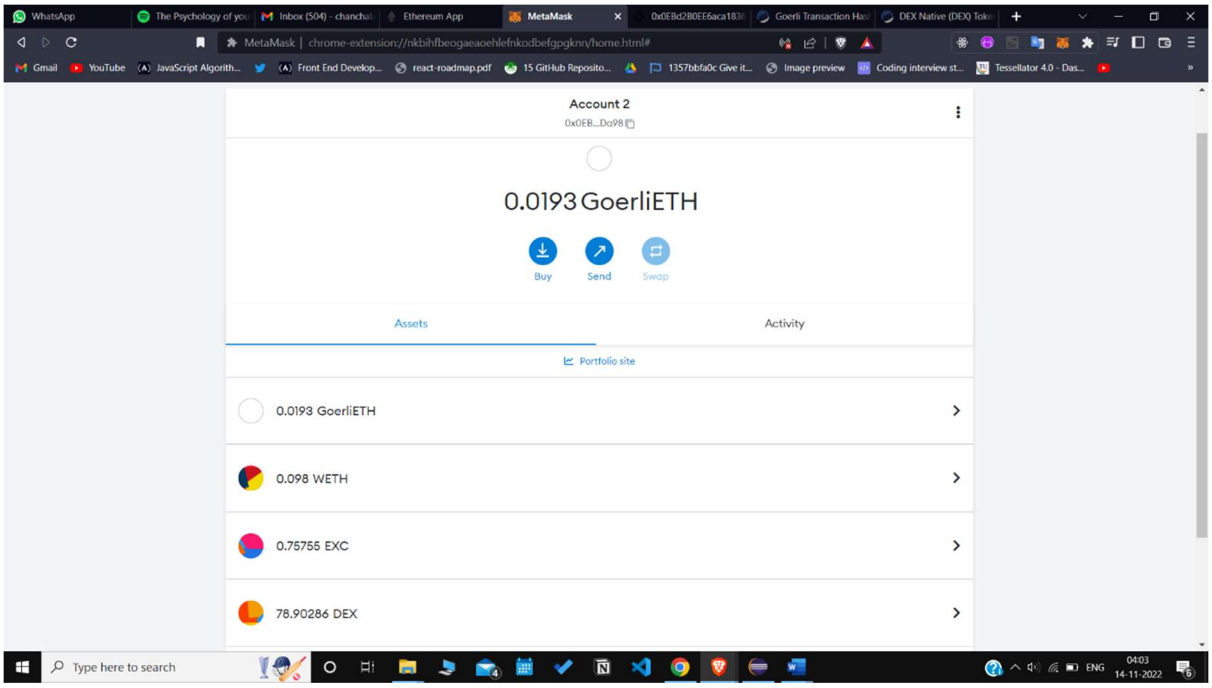
7.6 SUCCESS MESSAGE



7.7 TRASACTION DETAILS ON ETHERSCAN



7.8 FINAL BALANCE



CHAPTER 8

8. FUTURE ENCHANCEMENTS

In this application we have seen how one form of cryptocurrency can be swapped with another. Future enhancements of this application would be to implement more features of an automated market maker features.

Primary enhancement would be to include more forms of cryptocurrency liquidity pairs into the application.

Further enhancement would be is to develop and implement protocols which would allow users to register themselves as liquidity providers. Also, to provide application specific tokens and loyalty points to users who actively provide liquidity to the smart contract.

CHAPTER 9

9. CONCLUSION

The innovation of trustless financial exchange that serves as the foundation for intriguing prospects for the future of DeFi. Despite the emergence of various Decentralized Financial Exchanges, there are applications that still holds out as one of the biggest, hosting arguably the largest trading volume of all DEXs.

Providing liquidity in traditional finance markets is generally an investment form reserved to professional traders and institutions. The decentralized nature of the blockchain, on the other hand, allows for many individual liquidity providers to join together to facilitate trustless cryptocurrency exchanges on the blockchain while earning fees. Previous works have shown that providing liquidity on DEXs utilizing the original CPMM design can be a profitable investment, which is accessible to retail traders and only requires a few simple considerations from their side.

CHAPTER 10

10. REFERENCES

[1] Robin Fritsch. 2021. Concentrated Liquidity in Automated Market Makers. In Proceedings of the 2021 ACM CCS Workshop on Decentralized Finance and Security (DeFi@CCS), Virtual Event, Republic of Korea.

[2] Lioba Heimbach, Ye Wang, and Roger Wattenhofer. 2021. Behavior of Liquidity Providers in Decentralized Exchanges. In 2021 Crypto Valley Conference on Blockchain Technology (CVCBT), Rotkreuz, Switzerland.

[3] S. Wang, Y. Yuan, X. Wang, J. Li, R. Qin and F. -Y. Wang, "An Overview of Smart Contract: Architecture, Applications, and Future Trends," *2018 IEEE Intelligent Vehicles Symposium (IV)*, 2018, pp. 108-113, doi: 10.1109/IVS.2018.8500488.

[4] Yao, M., Di, H., Zheng, X. and Xu, X., 2018. Impact of payment technology innovations on the traditional financial industry: A focus on China. *Technological Forecasting and Social Change*, *135*, pp.199-207.

[5] Böhme, Rainer, Nicolas Christin, Benjamin Edelman, and Tyler Moore. "Bitcoin: Economics, technology, and governance." *Journal of economic Perspectives* 29, no. 2 (2015): 213-38.

[6] Dannen, C. (2017). *Introducing Ethereum and solidity* (Vol. 1, pp. 159-160). Berkeley: Apress.

[7] Wu, Kaidong. "An empirical study of blockchain-based decentralized applications." *arXiv preprint arXiv:1902.04969* (2019).

[8] Somin, Shahar, Goren Gordon, and Yaniv Altshuler. "Network analysis of erc20 tokens trading on ethereum blockchain." In *International Conference on Complex Systems*, pp. 439-450. Springer, Cham, 2018.

[9] Chen, Yan, Jack I. Richter, and Pankaj C. Patel. "Decentralized governance of digital platforms." *Journal of Management* 47, no. 5 (2021): 1305-1337.

[10] Bartoletti, M., Chiang, J. H. Y., & Lluch-Lafuente, A. (2021, June). A theory of automated market makers in defi. In *International Conference on Coordination Languages and Models* (pp. 168-187). Springer, Cham.