Decentralised Exchange

Final Year Project – Final Report

Session 2019-2023

**A 4th Year Student**

A project submitted in Partial fulfillment of the

COMSATS University Degree

of

BSc. (Hons.)BS in Computer Science / Software Engineering (CUI)



Department of Computer Science

COMSATS University Islamabad, Lahore Campus

12 June 2023

# Project Detail

| Type (Nature of project) | | | [ ] **D**evelopment [ ] **R**esearch [ \* ] **R**&**D** | | |
| --- | --- | --- | --- | --- | --- |
| Area of specialization | | | Decntralised exchange on blockchain | | |
| **Project Group Members** | | | | | |
| Sr.# | Reg. # | Student Name | | Email ID | \*Signature |
| (i) | CIIT/FA19-BCS-118/LHR | Muhammad Mujtaba Rehman | | fa19-bcs-118@cuilahore.edu.pk | A picture containing text  Description automatically generated |
| (ii) | CIIT/FA19-BCS-063/LHR | Muhammad Abdullah Bin Zubair | | fa19-bcs-063@cuilahore.edu.pk |  |
| (iii) | CIIT/FA19-BCS-136/LHR | Muhammad Bilal | | fa19-bcs-136@cuilahore.edu.pk | Text, letter  Description automatically generated |

\*The candidates confirm that the work submitted is their own and appropriate credit has been given where reference has been made to the work of others

# Plagiarism Free Certificate

This is to certify that, I am Muhammad Mujtaba Rehman S/D/o Mirza Razzi Ur Rehman, group leader of FYP under registration no CIIT/FA19-BCS-118/LHR at Computer Science Department, COMSATS University Islamabad, Lahore Campus. I declare that my FYP proposal is checked by my supervisor and the similarity index is 6% that is less than 20%, an acceptable limit by HEC. Report is attached herewith as Appendix A.A picture containing text

Description automatically generated

Date: 04/01/2023 Name of Group Leader: Muhammad Mujtaba Rehman Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_

Name of Supervisor: Akhzar Nazir Co-Supervisor (if any):\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Designation: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Designation: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Abstract**

The biggest drawback of a centralized banking system or any other centralized finance sector is that in a centralized system there is a controlling party that works as an intermediary between their end users or consumers. This intermediary has complete control over the money and other valuable resources of its end users. Therefore, to break free from the limitations of a centralized system that has the complete authority over our valuables, we are making a DeFi (Decentralized Finance) that lets its end users to exchange money in the form of cryptocurrency among themselves and lets its end users to provide liquidity to the liquidity pools. It will be based on smart contracts which are programs that are executed when predetermined conditions are met and are deployed on blockchain therefore, cannot be tempered.

Contents

[**Project Detail 2**](#_3dy6vkm)

[**Plagiarism Free Certificate 2**](#_4d34og8)

[1 Introduction 11](#_17dp8vu)

[1.1 Introduction: 12](#_3rdcrjn)

[1.2 Objective: 13](#_26in1rg)

[1.3 Problem Statement 13](#_35nkun2)

[1.4 Assumptions and Constraints 13](#_1ksv4uv)

[1.1.1 Assumptions 13](#_44sinio)

[1.1.2 Constraints 14](#_2jxsxqh)

[1.5 Motivation and Scope 14](#_z337ya)

[1.6 Tools and Technologies 15](#_3j2qqm3)

[2 Requirement Analysis: 15](#_1y810tw)

[2.1 Literature Review 15](#_4i7ojhp)

[1.1.3 Uniswap: 15](#_2xcytpi)

[1.1.4 1-inch: 15](#_1ci93xb)

[1.1.5 Balancer: 15](#_3whwml4)

[1.1.6 MakerDAO: 16](#_2bn6wsx)

[1.1.7 Curve: 16](#_qsh70q)

[2.2 Stakeholders List 16](#_3as4poj)

[2.2.1 Users: 16](#_1pxezwc)

[2.2.2 Token holders: 16](#_49x2ik5)

[2.2.3 Developers: 16](#_2p2csry)

[2.2.4 Investors: 17](#_147n2zr)

[2.2.5 Liquidity providers: 17](#_3o7alnk)

[2.2.6 Community members: 17](#_23ckvvd)

[2.3 Requirements Elicitations 17](#_ihv636)

[2.3.1 Functional Requirements 17](#_32hioqz)

[2.3.1.1 FR01: Automated Market Maker 17](#_1hmsyys)

[2.3.1.2 FR02: Smart Contracts 17](#_2grqrue)

[2.3.1.3 FR03: Low Fees 18](#_vx1227)

[2.3.1.4 FR04: Cross-chain 18](#_3fwokq0)

[2.3.1.5 FR05: Integration with Wallets 18](#_1v1yuxt)

[2.3.1.6 FR06: Customers Support 19](#_4f1mdlm)

[2.1.2 Non-Functional Requirements 21](#_28h4qwu)

[2.2 Use Case Description 24](#_nmf14n)

[2.3 Software Development Life Cycle (SDLC) 29](#_37m2jsg)

[2.3.1 Agile Model: 29](#_1mrcu09)

[2.3.2 Why Agile Model: 29](#_2lwamvv)

[**There are several reasons why an agile model might be beneficial for developing a decentralized exchange: 29**](#_111kx3o)

[3 SYSTEM DESIGN 30](#_3l18frh)

[3.1 WORK BREAKDOWN STRUCTURE (WBS) 30](#_206ipza)

[3.2 Use Case Diagrams 31](#_4k668n3)

[3.2.1 Connect Wallet 31](#_2zbgiuw)

[3.2.2 Swap Tokens 32](#_1egqt2p)

[3.2.3 Create Token 33](#_3ygebqi)

[3.2.4 Create Pool 34](#_2dlolyb)

[3.2.5 Bridge 35](#_sqyw64)

[3.2.6 Add Liquidity To Bridge 36](#_3cqmetx)

[3.2.7 Check Position 37](#_1rvwp1q)

[3.2.8 Trending Tokens 38](#_4bvk7pj)

[3.2.9 Trending Pools 39](#_2r0uhxc)

[3.3 ACTIVITY DIAGRAMS 40](#_1664s55)

[3.3.1 Connect Wallet Activity 40](#_3q5sasy)

[3.3.2 Disconnect Wallet Activity 41](#_25b2l0r)

[3.3.3 Swap Token Activity 42](#_kgcv8k)

[3.3.4 Token Activity 43](#_34g0dwd)

[3.3.5 Create Token Activity 44](#_1jlao46)

[3.3.6 Pool Activity 45](#_43ky6rz)

[3.3.7 Create Pool Activity 46](#_2iq8gzs)

[3.3.8 Check Position Activity 47](#_xvir7l)

[3.3.9 Bridge Activity 48](#_3hv69ve)

[3.3.10 Bridge Liquidity Activity 49](#_1x0gk37)

[3.4 Sequence Diagrams: 50](#_4h042r0)

[3.4.1 Connect Wallet 50](#_2w5ecyt)

[3.4.2 Swap Token 51](#_1baon6m)

[3.4.3 Token Creation 52](#_3vac5uf)

[3.4.4 Pool creation 53](#_2afmg28)

[3.4.5 Check position 54](#_pkwqa1)

[3.4.6 Bridge swap 55](#_39kk8xu)

[3.4.7 Bridge pool 56](#_1opuj5n)

[4 SYSTEM TESTING 57](#_48pi1tg)

[4.1 TEST CASES 57](#_2nusc19)

[4.1.1 TC01: Connect Wallet 57](#_1302m92)

[4.1.2 TC02: Disconnect Wallet 57](#_3mzq4wv)

[4.1.3 TC03: Add Liquidity 58](#_2250f4o)

[4.1.4 TC04: Remove Liquidity 59](#_haapch)

[4.1.5 TC05: Swap Token 59](#_319y80a)

[4.1.6 TC06: Check Transaction History 60](#_1gf8i83)

[4.1.7 TC07: Add Liquidity to a Pool with Insufficient Funds 60](#_40ew0vw)

[4.1.8 TC08: Removing more liquidity than present 61](#_2fk6b3p)

[4.1.9 TC09: Swapping Unsupported Token 62](#_upglbi)

[4.1.10 TC10: Check GAS Fee 62](#_3ep43zb)

[4.1.11 TC11: Withdraw GAS fee 63](#_1tuee74)

[4.1.12 TC12: Check Price of a Token in a pool 63](#_4du1wux)

[4.2 Unit / Integration / Acceptance Testing 64](#_2szc72q)

[4.2.1 Unit Testing 64](#_184mhaj)

[4.2.2 Integration Testing 65](#_3s49zyc)

[4.2.3 System Testing 65](#_279ka65)

[4.2.4 Acceptance Testing 65](#_meukdy)

[**5 Application Front-End 65**](#_qpigxcomzj0k)

[5.1 Web-interface 66](#_t6hqfhi8md56)

[5.1.1 Landing Page: 66](#_9a6n0819k2ea)

[5.1.2 Swap Page: 66](#_kx3i58p63cj9)

[5.1.3 Create Pool Page: 67](#_u8obwe34e6l1)

[5.1.4 Pool Position Page: 68](#_6iwpgw4w2pl4)

[5.1.5 Check Liquidity Page: 69](#_f1rlijo0jzot)

[5.1.6 RemoveLiquidity Page: 70](#_uv4miybadpwf)

[5.1.7 Create Token Page: 70](#_gcwkw12ewgcn)

[5.1.8 Trending Tokens Page: 71](#_kffi4bnq58ys)

[5.1.9 Token Stats Page: 72](#_orkv0fhabsgr)

[5.1.10 Deposit Amount to Bridge: 73](#_suknormjb1d5)

[5.1.11 Withdraw Amount from the Bridge: 74](#_sqjv7g56nfxl)

[5.1.12 Add Liquidity to Bridge: 75](#_swa5jliiikqo)

[5.2 Mobile-interface 76](#_p3j69sjteod)

[5.2.1 Landing Page: 76](#_3sguubob2vg1)

[5.2.2 Swap Page: 77](#_h5pwx7cs1hvw)

[5.2.3 Create Pool Page: 78](#_ix919q9zzvoc)

[5.2.4 Pool Position Page: 79](#_rv8qfc9afpus)

[5.2.5 Check Liquidity Page: 80](#_gk0uc77rz4vd)

[5.2.6 Create Token Page: 81](#_vjuimxgmba53)

[5.2.7 Deposit Amount to Bridge: 82](#_jy0pnrd4gn9n)

[5.2.8 Withdraw Amount from Bridge: 83](#_11tzrsh5qvv2)

[5.2.9 Add Liquidity To Bridge: 84](#_32zgh07lhkjn)

[6 Conclusion 85](#_36ei31r)

[6.1 Problems faced and lessons learned 85](#_1ljsd9k)

[6.1.1 Problems faced: 85](#_45jfvxd)

[6.1.2 Lessons Learned: 85](#_2koq656)

[6.2 Project Summary 85](#_zu0gcz)

[6.3 Future Work 85](#_1yyy98l)

[7 References 86](#_4iylrwe)

**List Of Tables**

[Table 1 FR-01](#_41mghml) 18

[Table 2 FR-02](#_2grqrue) 18

[Table 3 FR-03](#_vx1227) 19

[Table 4 FR-04](#_3fwokq0) 19

[Table 5 FR-05](#_1v1yuxt) 19

[Table 6 FR-06](#_4f1mdlm) 20

[Table 7 FR-07](#_2u6wntf) 20

[Table 8 FR-08](#_19c6y18) 20

[Table 9 FR-09](#_3tbugp1) 21

[Table 10 FR-10](#_3tbugp1) 21

[Table 11 FR-11](#_3tbugp1) 21

[Table 12 FR-12](#_3tbugp1) 21

[Table 13 FR-13](#_3tbugp1) 22

[Table 14 FR-14](#_3tbugp1) 22

[Table 15 NFR-01](#_2y3w247) 22

[Table 16 NFR-02](#_2y3w247) 23

[Table 17 NFR-03](#_2y3w247) 23

[Table 18 NFR-04](#_2y3w247) 23

[Table 19 NFR-05](#_2y3w247) 24

[Table 20 NFR-06](#_2y3w247) 24

[Table 21 NFR-07](#_2y3w247) 24

[Table 22 NFR-08](#_2y3w247) 24

[Table 23 UC-01](#_1d96cc0) 25

[Table 24 UC-02](#_1d96cc0) 25

[Table 25 UC-03 2](#_1d96cc0)6

[Table 26 UC-04 2](#_1d96cc0)7

[Table 27 UC-05 2](#_1d96cc0)7

[Table 28 UC-06 2](#_1d96cc0)8

[Table 29 UC-07 2](#_1d96cc0)8

[Table 30 UC-08 2](#_1d96cc0)9

[Table 31 TC-01](#_1d96cc0) 58

[Table 32 TC-02](#_3x8tuzt) 58

[Table 33 TC-03](#_2ce457m) 59

[Table 34 TC-04](#_rjefff) 60

[Table 35 TC-05](#_3bj1y38) 60

[Table 36 TC-06](#_1qoc8b1) 61

[Table 37 TC-07](#_4anzqyu) 61

[Table 38 TC-08 6](#_2pta16n)2

[Table 39 TC-09 6](#_14ykbeg)3

[Table 40 TC-10 6](#_14ykbeg)3

[Table 41 TC-11 6](#_14ykbeg)4

[Table 42 TC-12 6](#_14ykbeg)4

**List of Figures**

[Figure 1 The Agile model](#_3oy7u29) 30

[Figure 2 WBS](#_3oy7u29) 31

[Figure 3 Connect wallet UC](#_3oy7u29) 32

[Figure 4 Swap tokens UC](#_3oy7u29) 33

[Figure 5 Create Token UC](#_3oy7u29) 34

[Figure 6 Create Pool UC](#_3oy7u29) 35

[Figure 7 Bridge UC](#_3oy7u29) 36

[Figure 8 Add Liquidity to a bridge UC](#_3oy7u29) 37

[Figure 9 Check position UC](#_3oy7u29) 38

[Figure 10 Trending tokens UC](#_3oy7u29) 39

[Figure 11 Trending pools UC](#_3oy7u29) 40

[Figure 12 Connect Wallet Activity](#_3oy7u29) 41

[Figure 13 Disconnect Wallet Activity 4](#_3oy7u29)2

[Figure 14 Swap Token Activity 4](#_3oy7u29)3

[Figure 15 Token Actvity 4](#_3oy7u29)4

[Figure 16 Create Token Activity](#_3oy7u29) 45

[Figure 17 Pool Activity](#_3oy7u29) 46

[Figure 18 Create Pool Activity](#_3oy7u29) 47

[Figure 19 Check Position Activity](#_3oy7u29) 48

[Figure 20 Bridge Activity](#_3oy7u29) 49

[Figure 21 Bridge Liquidity activity](#_3oy7u29) 50

[Figure 22 Connect Wallet Sequence Diagram](#_3oy7u29) 51

[Figure 23 Swap tokens Sequence Diagram](#_3oy7u29) 52

[Figure 24 Token Creation Sequence Diagram](#_3oy7u29) 53

[Figure 25 Pool Creation Sequence Diagram](#_3oy7u29) 54

[Figure 26 Check Position Sequence Diagram](#_3oy7u29) 55

[Figure 27 Bridge Swap Sequence Diagram 5](#_3oy7u29)6

[Figure 28 Bridge Pool Sequence Diagram](#_3oy7u29) 57

[Figure 29 Landing Page ScreenShot](#_3oy7u29) 67

[Figure 30 SwapPage ScreenShot](#_3oy7u29) 67

[Figure 31 Initialize Page ScreenShot](#_3oy7u29) 68

[Figure 32 Add Liquidity Page ScreenShot](#_3oy7u29) 69

[Figure 33 Pool Position Page ScreenShot](#_3oy7u29) 69

[Figure 34 Check Liquidity Page ScreenShot](#_3oy7u29) 70

[Figure 35 Remove Liquidity Page ScreenShot](#_3oy7u29) 71

[Figure 36 Create Your Token ScreenShot](#_3oy7u29) 71

[Figure 37 Trending Tokens ScreenShot](#_3oy7u29) 72

[Figure 38 Token stats ScreenShot](#_3oy7u29) 73

[Figure 39 Deposit Token To Bridge ScreenShot](#_3oy7u29) 74

[Figure 40 Withdraw Token From Bridge ScreenShot](#_3oy7u29) 75

[Figure 41 Provide Liquidity To Bridge ScreenShot](#_3oy7u29) 76

[Figure 42 Provide Liquidity To Bridge ScreenShot](#_3oy7u29) 77

[Figure 43 Provide Liquidity To Bridge ScreenShot](#_3oy7u29) 78

[Figure 44 Provide Liquidity To Bridge ScreenShot](#_3oy7u29) 79

[Figure 45 Provide Liquidity To Bridge ScreenShot](#_3oy7u29) 80

[Figure 46 Provide Liquidity To Bridge ScreenShot](#_3oy7u29) 81

[Figure 47 Provide Liquidity To Bridge ScreenShot](#_3oy7u29) 82

[Figure 48 Provide Liquidity To Bridge ScreenShot](#_3oy7u29) 83

[Figure 49 Provide Liquidity To Bridge ScreenShot](#_3oy7u29) 84

[Figure 50 Provide Liquidity To Bridge ScreenShot](#_3oy7u29) 85

# Introduction

## Introduction:

Before the invention of money, people used a barter system. Bartering refers to direct trade of goods or services. Let suppose person A has 10 apples, but he wants 10 bananas and person B wants 10 apples, but he has 10 bananas, so he would trade them for 10 apples. The bartering system had many limitations such as one cannot hold his/her wealth in bartering as one cannot store apples or bananas for a long amount of time. Moreover, bartering lacks the double coincidence of wants. In simple words, in bartering a person who is selling his/her goods has to find the right person that is not only willing to purchase what the other person is selling but also willing to sell what the other person requires.

The limitations in the bartering system led to the invention of standardized currency. The concept of currency and currency exchange has always been known to people for almost the last five thousand years. With the passing time the form of currency evolved but these concepts remained the same. Later on, the currency evolved into coins or notes and centralized banks were formed. With the changing needs of people, the concept of banking evolved. Today, most societies use a combination of physical currency (such as coins and paper bills) and electronic forms of payment (such as debit and credit cards) as their medium of exchange. These forms of currency are backed by central banks and governments, which helps to ensure their value and stability.

Centralized currency and exchanges have been our only medium for our trades and money exchange for many years .In the past we had no other efficient medium present that could help us in holding or exchanging our money or other valuables. This centralized finance system was proven helpful at that time but wasn’t cost effective. The centralized banks today charge their consumers yearly for providing their services. In Pakistan, banks are required to collect 0.6 percent of the amount withdrawn as withdrawal fees. Moreover, they can manipulate the prices and transaction fees as they please.

Binance, a famous centralized exchange all over the world and in Pakistan deducts 0.10% of the amount transacted, and in case of instant buy or sell, it deducts 0.50% of the total amount transacted. Their end users can not just go to the exchange and make an exchange. As for an exchange to happen in these centralized exchanges, a seller needs to have a buyer. Furthermore, to make an exchange in a centralized exchange you need to KYC (know your customer) i.e., you need to provide the exchange with your personal information in order for them to verify that you are the person that you are claiming to be. Whereas, a decentralized exchange does not require its users to sign up and provide their personal information. They can connect to the exchange by connecting their wallet through their wallet address.

A decentralized exchange works solely on the rules that are defined in its smart contracts and transactions are executed only when predetermined conditions in the smart contracts are met. Therefore, there is no need for an intermediary or a middleman between the buyers and sellers. In a DEX, we have liquidity pools of assets where liquidity providers invest their assets and the traders can make trades from these pools, so for a buyer, there does not have to be a seller. Traders can simply make trades from a liquidity pool. One of the main significance a decentralized exchange has over a centralized exchange is low transaction fee. Only a small amount i.e., 0.3% of the total amount of transaction goes to the liquidity providers according to the amount of liquidity they provide to the pool. This fee is important as without it there will be no incentive for the liquidity providers to stake their money.

## Objective:

Our decentralized exchange has the potential to overcome all the shortcomings that remained unsolved centralized exchanges. We will create a decentralized exchange that will let its end users have full custody over their currency. It will work based on the rules that will be predefined on the smart contracts. Smart contract is a program that is either written on Rust or Solidity. As smart contracts are uploaded on the blockchain, therefore, they are unalterable. Which will make our DEX secure.

In addition to everything that is mentioned above, our decentralized exchange will also provide its users with the following features.

* Our decentralized exchange will have its own native currency/token which its holders can trade with other coins.
* We are also providing our users a feature of Create your own token. This feature lets them create their own token by simply providing the token's name, token’s symbol, number of decimals, and the total supply of tokens.
* They can later use their token in pools while creating their own pool.
* It also lets its liquidity providers or traders transfer their funds from one blockchain to another with the help of a bridge.
* Liquidity providers can also provide liquidity to other blockchains to get more transaction fees.
* We are also providing our users with the facility of taking flash loans.

A flash loan is a type of short-term, high-risk loan that is facilitated through smart contracts on a decentralized finance (DeFi) platform. Flash loans allow users to borrow a large amount of cryptocurrency for a very short period of time, typically just a few seconds or minutes. This means that users can borrow a large amount of cryptocurrency without putting up any of their own assets as collateral.

## Problem Statement

Centralized Exchanges like Binance or Coinbase have complete control over their end user’s money. If these Exchanges shut down, all their end users will get notified beforehand so that they have enough time to transfer money from their wallet into another wallet. On the other hand, in case if they get bankrupt due to some hack on their wallet storage, it is very likely that you would not be able to get your money back.

## Assumptions and Constraints

The following assumptions and constraints are defined for the users so that they have the necessary knowledge required to use our decentralized exchange.

### Assumptions

* **Decentralization:** DEXs are decentralized platforms that work on a distributed network, rather than having a central authority that can control it.
* **Volatility:** The value of cryptocurrency and other decentralized assets is volatile. It may decline or increase and can have a big impact on the trades made on DEX. This means that users may face significant losses if the value of an asset they hold declines sharply.

### Constraints

* **Understanding:** User understands how a decentralized exchange works.
* **Limited liquidity:** DEXs may not have the same level of liquidity as centralized exchanges, which can make it more difficult to buy and sell certain assets. This can result in wider spreads and longer wait times for trades to be executed.
* **Crypto wallet:** User has at least one crypto currency wallet to connect to the decentralized exchange.
* **Cryptocurrencies and digital assets**: DEXs typically only allow users to trade cryptocurrencies and other digital assets that are built on blockchain technology. This means that they do not offer the same range of assets that are on a centralized exchange.
* **Lack of central authority:** As decentralized exchanges do not have a controlling party therefore they do not provide the same level of protection and security as a centralized exchange does. Users are responsible for their own security. They can secure their assets by using secure wallets and keeping their private keys safe and secure.

## Motivation and Scope

As discussed above, centralization is one of the biggest problems in the industry of finance while using secure distributed ledger. The goal of our decentralized Exchange is to provide a decentralized exchange that is better than the centralized exchanges that are currently in the market and also provides solutions to the unsolved problems in the existing decentralized exchanges. Exchanges like Binance or Coinbase are centralized. This means they own the assets of their users. They can manipulate the prices as they see fit, they can manipulate transaction fees. To put it shortly, they act as an intermediary party between their end users that can control their assets. Whereas decentralized exchange eliminates the concept of and the intermediary from the roots. The motivation of our project is to break people free from the centralized system where they do not have control over their own assets to a system where they have complete control over their assets. They can simply trade their assets without providing their personal information and having the need to know their customer.

The scope of our project is to provide the society with a decentralized exchange that provides better solutions to problems like importing assets from one blockchain to another and lets them provide liquidity to other blockchain to gain more profit. It goes even further to providing our users to create their own tokens and create their pools. There are already millions of dollars invested in existing decentralized exchanges like Uniswap, Quickswap, Curve and Balancer. Anyone from anywhere in the world can use Decentralized Finance (DeFi). DeFi gives us more control over our money through personal wallets and trading services. The total monthly revenue in DeFi was $189 million in 2021 and now it's a $200 million industry. Despite the downturn of the market, the adoption of DeFi continues and the number of wallets has been increased to 4.3 million unique wallet addresses.

## Tools and Technologies

* **Frontend:** React JS, Java Script, CSS, Bootstrap, Figma.
* **Backend:** Node JS, Express JS, web3 JS,
* **Database:** MongoDB
* **Blockchain:** EVM Compatible
* **Smart Contract:** Solidity
* **Testing Libraries:** chai/mocha
* **Testing Frameworks:** Ganache, Truffle
* **Tools:** VS Code, Postman, MetaMask

# Requirement Analysis:

## Literature Review

### Uniswap:

Uniswap is a decentralized exchange that is the biggest exchange on the Ethereum network at the moment. They have deployed their 3rd version of it. In the 2nd version, they introduced a new system of token-to-token swap. In this version, every token pair has its pool. They have made the swapping of tokens way more easy by making internal swaps if the token pair is not available.

In Uniswap’s 3rd version they introduced a new mechanism of selecting 3 liquidity fee options i.e. 0.3, 0.1, and 0.05. There are now pools having the same tokens but different fee mechanisms. They have also introduced a new version of the router which works on an advanced smart order routing algorithm that finds the best path and max output tokens from the given pools while considering the gas cost.

### 1-inch:

1inch is also a decentralized exchange like Uniswap that provides more options for DEFI protocols different than the current. 1inch allows users to access liquidity from multiple DEXs in a single transaction. It uses a smart contract-based routing system to find the best prices and liquidity across different DEXs and executes trades on behalf of users.

1inch introduces a limit order buying and selling of the crypto assets. A limit order is creating an order for buying a token at a lower price and selling that particular token at a higher price than the market price. They have also introduced cross-chain swapping by providing the option of the bridge to send tokens to almost 10 different exchanges.

### Balancer:

Balancer is a decentralized finance (DeFi) protocol that allows users to create and manage customizable liquidity pools and automate the trade of tokens within those pools. It operates on the Ethereum blockchain and is designed to provide users with a decentralized, automated way to earn passive income by providing liquidity to other users.

To create a liquidity pool on Balancer, users must deposit a combination of two or more tokens into the pool. The value of each token in the pool is determined by its weight, which is expressed as a percentage of the total pool value. For example, if a pool contains 100 tokens and Token A has a weight of 50%, it represents half of the total value of the pool.

### MakerDAO:

MakerDAO is a decentralized autonomous organization (DAO) that operates on the Ethereum blockchain. It is designed to provide stable, low-risk cryptocurrency loans to users by using a decentralized collateral system.

The main component of MakerDAO is the Dai stablecoin, which is designed to maintain a stable value relative to a specific fiat currency, such as the US dollar. To create Dai, users must collateralize their assets, such as Ether (ETH) or other cryptocurrencies, in the MakerDAO system. The value of the collateral is then used to generate an equivalent value of Dai, which is issued to the user as a loan.

Users can use the Dai they receive to make purchases or payments, or they can hold it as a stable store of value. To repay the loan and reclaim their collateral, users must pay back the equivalent value of Dai plus a small stability fee.

MakerDAO is governed by a decentralized governance system in which MKR token holders can participate in voting on important decisions related to the system's operation. The MKR token is used to pay stability fees and can also be bought and sold on cryptocurrency exchanges.

### Curve:

Curve is a decentralized exchange (DEX) and liquidity pool protocol that allows users to trade stablecoins and other cryptocurrencies in a fast and efficient manner. One of the main features of Curve is its liquidity pool, which allows users to provide liquidity to the DEX in exchange for a share of the trading fees generated by the platform. The liquidity pool is designed to be highly efficient, with low trading fees and tight spreads, which makes it attractive to traders.

In addition to its liquidity pool, Curve also offers a number of other features that make it a popular choice for trading stablecoins. These features include support for multiple stablecoins, a user-friendly interface, and the ability to trade using a variety of order types.

## Stakeholders List

### Users:

These are individuals or entities who use our DEX to buy and sell cryptocurrencies.

### Token holders:

If the DEX is built on a blockchain that uses a native token, such as Ethereum, token holders may also be stakeholders in the DEX. They may benefit from the success of the DEX through the value of their tokens.

### Developers:

The team of developers who build and maintain the DEX are also stakeholders. They may be motivated to ensure the DEX is successful in order to receive rewards or compensation.

### Investors:

If the DEX has received funding from outside investors, they may also be considered stakeholders in the DEX. They may have a financial interest in the success of the DEX.

### Liquidity providers:

In a DEX, liquidity providers are individuals or entities that offer to buy or sell certain assets on the exchange in order to facilitate trades and maintain liquidity. They may be considered stakeholders in the DEX as their actions can influence the success of the exchange.

### Community members:

Some DEXs may have a strong community of users who are actively involved in the development and growth of the exchange. These community members may also be considered stakeholders in the DEX.

## Requirements Elicitations

### Functional Requirements

Functional requirements for our project are as follows:

### 2.3.1.1 FR01: Automated Market Maker

| **Req. No.** | **Functional Requirements** |
| --- | --- |
| **FR01-01** | Users can swap tokens anytime |
| **FR01-02** | User should not wait for any other user to buy his/her tokens |
| **FR01-03** | AMM will automatically update the prices of the tokens |

*Table 1 FR-01*

### 2.3.1.2 FR02: Smart Contracts

| **Req. No.** | **Functional Requirements** |
| --- | --- |
| **FR02-01** | The DEX should use smart contracts to facilitate the execution of trades. |
| **FR02-02** | Smart contracts are self-executing contracts with the terms of the agreement between buyer and seller being directly written into lines of code. |

*Table 2 FR-02*

### 2.3.1.3 FR03: Low Fees

| **Req. No.** | **Functional Requirements** |
| --- | --- |
| **FR03-01** | DEXs often have lower fees than centralized exchanges, as they do not need to cover the costs of running a central authority. |
| **FR03-02** | It is still important for the DEX to have reasonable fees to ensure that it is attractive to users. |

*Table 3 FR-03*

### 2.3.1.4 FR04: Cross-chain

| **Req. No.** | **Functional Requirements** |
| --- | --- |
| **FR04-01** | DEXs support trading between different blockchain networks |
| **FR04-02** | Allowing users to trade tokens from one blockchain for tokens on another |

*Table 4 FR-04*

### 2.3.1.5 FR05: Integration with Wallets

| **Req. No.** | **Functional Requirements** |
| --- | --- |
| **FR05-01** | The DEX should be able to integrate with different types of cryptocurrency wallets. |
| **FR05-02** | Allowing users to easily access and manage their funds |

*Table 5 FR-05*

### 2.3.1.6 FR06: Customers Support

| **Req. No.** | **Functional Requirements** |
| --- | --- |
| **FR06-01** | The DEX should have a customer support team available to answer questions and resolve issues for users. |
| **FR06-02** | This may include a knowledge base, a support email or chat service. |

*Table 6 FR-06*

#### 2.3.1.7 FR07: User Control

| **Req. No.** | **Functional Requirements** |
| --- | --- |
| **FR07-01** | Users of a DEX should have full control over their own assets. |
| **FR07-02** | Users are able to make trades and manage their accounts without the need for intermediaries. |

*Table 7 FR-07*

#### 2.3.1.8 FR08: Transparency

| **Req. No.** | **Functional Requirements** |
| --- | --- |
| **FR08-01** | A DEX should be transparent in its operations. |
| **FR08-02** | All transactions and trades being publicly available on the blockchain for anyone to see. |

*Table 8 FR-08*

#### 2.3.1.9 FR09: User-friendly interface

| **Req. No.** | **Functional Requirements** |
| --- | --- |
| **FR09-01** | A DEX should have a user-friendly interface that is easy for users to navigate and understand. |

*Table 9 FR-09*

#### FR10: Fast transaction times

| **Req. No.** | **Functional Requirements** |
| --- | --- |
| **FR10-01** | A DEX should be able to process transactions quickly to ensure that trades are completed efficiently. |

*Table 10 FR-10*

#### FR11: High liquidity

| **Req. No.** | **Functional Requirements** |
| --- | --- |
| **FR11-01** | A DEX should have a sufficient number of users. |
| **FR11-02** | Wide variety of assets available for trading to ensure that there is always a market for any given asset. |

*Table 11 FR-11*

#### FR12: Mobile Support

| **Req. No.** | **Functional Requirements** |
| --- | --- |
| **FR12-01** | The DEX should have a mobile-optimized version of its platform. |
| **FR12-02** | Allow users to access it on their phones and tablets. |

*Table 12 FR-12*

#### FR13: Customizable interface

| **Req. No.** | **Functional Requirements** |
| --- | --- |
| **FR13-01** | A DEX may offer customization options for the user interface |
| **FR13-02** | The ability to rearrange widgets or change the color scheme, to meet the preferences of different use |

*Table 13 FR-13*

#### FR14: Market-making capabilities

| **Req. No.** | **Functional Requirements** |
| --- | --- |
| **FR14-01** | A DEX may offer market-making capabilities to help ensure that there is always liquidity available for certain assets. |
| **FR14-02** | If the liquidity goes below some level it will allow liquidity providers more chance to earn profits. |

*Table 14 FR-14*

### Non-Functional Requirements

#### NFR01: Performance

| **Req. No.** | **Functional Requirements** |
| --- | --- |
| **NFR 01-01** | The DEX should have a fast and responsive interface. |
| **NFR 01-02** | With low latency and high throughput to enable smooth and efficient trading. |

*Table 15 NFR-01*

#### NFR02: Scalability

| **Req. No.** | **Functional Requirements** |
| --- | --- |
| **NFR 02-01** | The DEX should be able to handle a large number of users. |
| **NFR 02-02** | Transactions without experiencing performance issues. |

*Table 16 NFR-02*

#### NFR03: Reliability

| **Req. No.** | **Functional Requirements** |
| --- | --- |
| **NFR 03-01** | The DEX should be reliable and consistently available |
| **NFR 03-02** | With minimal downtime or other disruptions. |

*Table 17 NFR-03*

#### NFR04: Security

| **Req. No.** | **Functional Requirements** |
| --- | --- |
| **NFR 04-01** | The DEX should have strong security measures in place to protect against threats such as hacks, fraud, and other types of cyber-attacks. |

*Table 18 NFR-04*

#### NFR05: Maintainability

| **Req. No.** | **Functional Requirements** |
| --- | --- |
| **NFR 05-01** | The DEX should be easy to maintain and update |
| **NFR 05-02** | With a well-documented codebase and a clear upgrade process. |

*Table 19 NFR-05*

#### NFR06: Integration with external liquidity providers

| **Req. No.** | **Functional Requirements** |
| --- | --- |
| **NFR 06-01** | A DEX may offer integration with external liquidity providers to help ensure that there is always sufficient liquidity available for trading |

*Table 20 NFR-06*

#### NFR07: API support

| **Req. No.** | **Functional Requirements** |
| --- | --- |
| **NFR 07-01** | The DEX should offer an API (Application Programming Interface) that allows developers to build custom applications that integrate with the exchange. |
| **NFR 07-02** | This can enable advanced trading functionality and enable integration with other systems. |

*Table 21 NFR-07*

#### NFR08: Automation

| **Req. No.** | **Functional Requirements** |
| --- | --- |
| **NFR 08-01** | A DEX may offer automation capabilities, such as trading bots |
| **NFR 08-02** | Automatic rebalancing, to allow users to streamline their trading processes. |

*Table 22 NFR-08*

## Use Case Description

#### Trading cryptocurrencies

| Id | UC-01 |
| --- | --- |
| Name | Trading cryptocurrencies |
| Actors | Trader |
| Summary | Users can trade various types of cryptocurrencies on a DEX, often with lower fees and faster transaction times compared to traditional exchanges. |
| Pre-conditions | Users should have a cold wallet.  Users should have ETH to pay for gas fees.  Users should have tokens what they want to swap. |
| Post-conditions | Users will receive their tokens in their wallet. |
| Normal Course of Flow | 1. The users will receive tokens if there was less slippage. |

*Table 23 UC-01*

#### Earning passive income

| Id | UC-02 |
| --- | --- |
| Name | Earning passive income |
| Actors | Liquidity Providers |
| Summary | Users can lend their cryptocurrencies on a DEX and earn interest from traders, or participate in liquidity provision to earn trading fees. |
| Pre-conditions | Users should have both tokens that they are providing liquidity for.  Users should have ETH to pay for gas fees.  Users should select fees they want from the trades.  Users should select a price range in which they want to provide liquidity. |
| Post-conditions | Users will now have a position open.  Users can collect fees any time they want.  Users now own some part of the pool's liquidity. |
| Normal Course of Flow | 1. Users will be able to get fees for the liquidity he/she provided. 2. He should get fees whenever a trade happens in the marked region. |

*Table 24 UC-02*

#### Connect wallet

| Id | UC-03 |
| --- | --- |
| Name | Connect wallet |
| Actors | Users |
| Summary | Users will connect their wallets with the exchange to use all the features of the app. |
| Pre-conditions | Users should have a wallet installed. |
| Post-conditions | Users will connect with the app after signing a message. |
| Normal Course of Flow | 1. Users can now use any feature in the app without any problem. |

*Table 25 UC-03*

#### Create Token

| Id | UC-04 |
| --- | --- |
| Name | Create Token |
| Actors | Users |
| Summary | Users can create custom ERC20 tokens without the need of code. |
| Pre-conditions | Users should have fees in ETH to pay for the transaction.  Users should provide the required parameters for the coin. |
| Post-conditions | Users can now have the address of their own deployed ERC20 token.  Users can now create pools for their own token. |
| Normal Course of Flow | 1. Users now have a token deployed. 2. Token is deployed without any error on the mainnet. |

*Table 26 UC-05*

#### Bridge

| Id | UC-05 |
| --- | --- |
| Name | Bridge |
| Actors | Users |
| Summary | Users can transfer their funds from one blockchain to other using the bridge. |
| Pre-conditions | Users should have fees and enough tokens to send on the other blockchain. |
| Post-conditions | Users receive their tokens on the other blockchain without any need of intermediary. |
| Normal Course of Flow | 1. User get tokens in their wallet |

*Table 27 UC-05*

#### Tokens Data

| Id | UC-06 |
| --- | --- |
| Name | Tokens Data |
| Actors | Users (Traders) |
| Summary | Users can watch tokens data and graphs to analyze the market to buy or sell the asset on their technical analysis. |
| Pre-conditions | Users don't have to require anything for viewing tokens charts and data. |
| Post-conditions | Users will have a great understanding of market movement on that particular token. |
| Normal Course of Flow | 1. The data and price graph of the token is perfectly synced. |

*Table 28 UC-06*

#### Position

| Id | UC-07 |
| --- | --- |
| Name | Position |
| Actors | Users (Liquidity Provides) |
| Summary | Users can manage their positions or open/remove their positions and collect fees easily. |
| Pre-conditions | User should have at least one open position to use this feature |
| Post-conditions | Users can collect fees.  Users can add liquidity  User can remove liquidity  User can create new position |
| Normal Course of Flow | 1. Users will be able to perform all the Post conditions mentioned above without any ambiguity. |

*Table 29 UC-07*

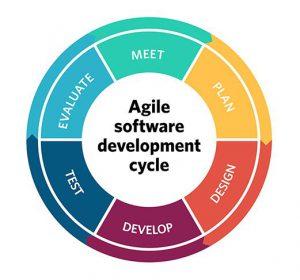
#### Pools data

| Id | UC-08 |
| --- | --- |
| Name | Pools data |
| Actors | Users (Traders) |
| Summary | Users can watch Pools data and graphs to analyze which pool will benefit him/her the most. |
| Pre-conditions | Users don't have to require anything for viewing tokens charts and data. |
| Post-conditions | Users will have a great understanding of pool movement and how many trades are happening in that particular pool. |
| Normal Course of Flow | 1. The data and price graph of the pool is perfectly synced. |

*Table 30 UC-08*

## Software Development Life Cycle (SDLC)

### Agile Model:

Agile is a project management approach that emphasizes flexibility and collaboration. It is based on the Agile Manifesto, a set of values and principles that guide the development of software and other products. Agile methodologies are designed to be adaptable to change, as they allow for frequent adjustments and modifications to the project plan. In an agile model, project work is divided into short iterations, called sprints, that usually last from one to four weeks. Each sprint begins with a planning meeting, where the team determines what work will be completed during the sprint. The team then works on completing that work, with daily check-ins (called stand-ups) to discuss progress and address any issues that have arisen. At the end of each sprint, the team reviews the completed work and adjusts the project plan as needed.

*Figure 1 Agile SDLC*

### Why Agile Model:

# There are several reasons why an agile model might be beneficial for developing a decentralized exchange:

* **Flexibility:** Agile allows for frequent adjustments to the project plan, which can be especially useful in a rapidly-evolving space like decentralized exchanges.
* **Continuous** **delivery:** Agile emphasizes the delivery of working software as soon as possible, which can help ensure that the decentralized exchange is delivered to users as quickly as possible.
* **Collaboration:** Agile promotes collaboration and communication between team members, which can be important when developing a decentralized exchange, as it may involve multiple teams working together to build different components of the exchange.
* **Risk management**: The short iteration cycle of agile allows for early identification and management of risks, which can be particularly important when building a decentralized exchange, as there may be many unknowns and uncertainties involved.

# SYSTEM DESIGN

## WORK BREAKDOWN STRUCTURE (WBS)Diagram Description automatically generated

*Figure 2 WBS*

## Use Case Diagrams

### Connect Wallet

**Chart, scatter chart, bubble chart

Description automatically generated**

*Figure 3 UC Diagram 1*

### Swap Tokens

**Chart, scatter chart, bubble chart

Description automatically generated**

*Figure 4 UC Diagram 2*

### Create Token

**Chart, scatter chart, bubble chart

Description automatically generated**

*Figure 5 UC Diagram 3*

### Create Pool

**Chart, scatter chart, bubble chart

Description automatically generated**

*Figure 6 UC Diagram 4*

### Bridge Chart, scatter chart, bubble chart Description automatically generated

*Figure 7 UC Diagram 5*

### Add Liquidity To Bridge

**Chart, scatter chart, bubble chart

Description automatically generated**

*Figure 8 UC Diagram 6*

### Check Position

**Chart, scatter chart, bubble chart

Description automatically generated**

*Figure 9 UC Diagram 7*

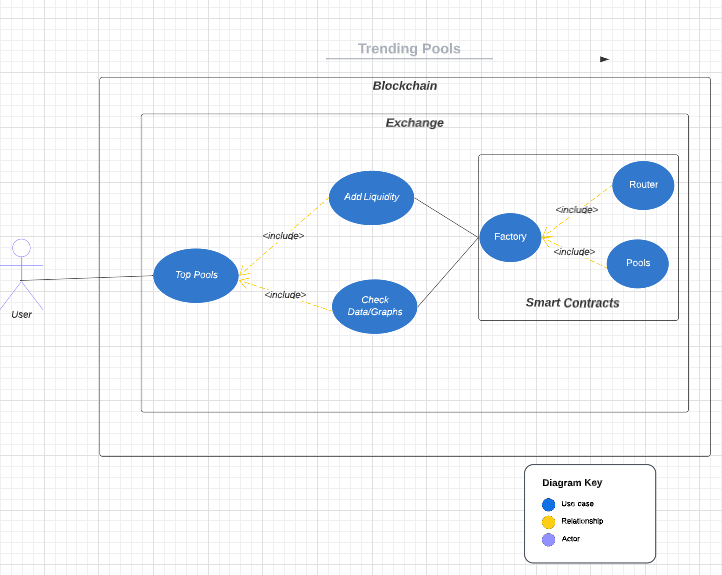
### Trending Tokens

Chart, scatter chart, bubble chart

Description automatically generated

*Figure 10 UC Diagram 8*

### Trending Pools



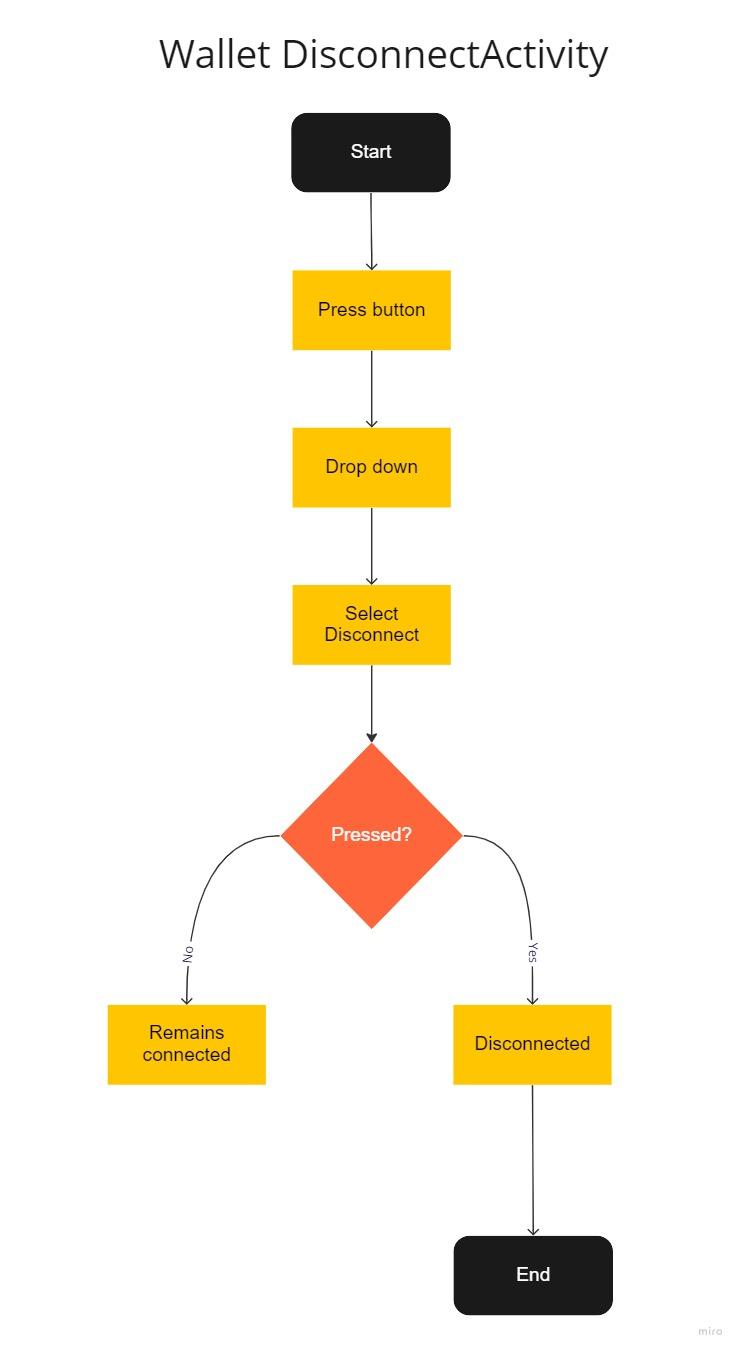
*Figure 11 UC Diagram 9*

## ACTIVITY DIAGRAMS

### Connect Wallet Activity

*Figure 12 Activity Diagram 1*

### Disconnect Wallet Activity



*Figure 13 Activity Diagram 2*

### Swap Token Activity

Diagram

Description automatically generated

*Figure 14 Activity Diagram 3*

### Token Activity

Diagram

Description automatically generated

*Figure 15 Activity Diagram 4*

### Create Token Activity

Diagram

Description automatically generated

*Figure 16 Activity Diagram 5*

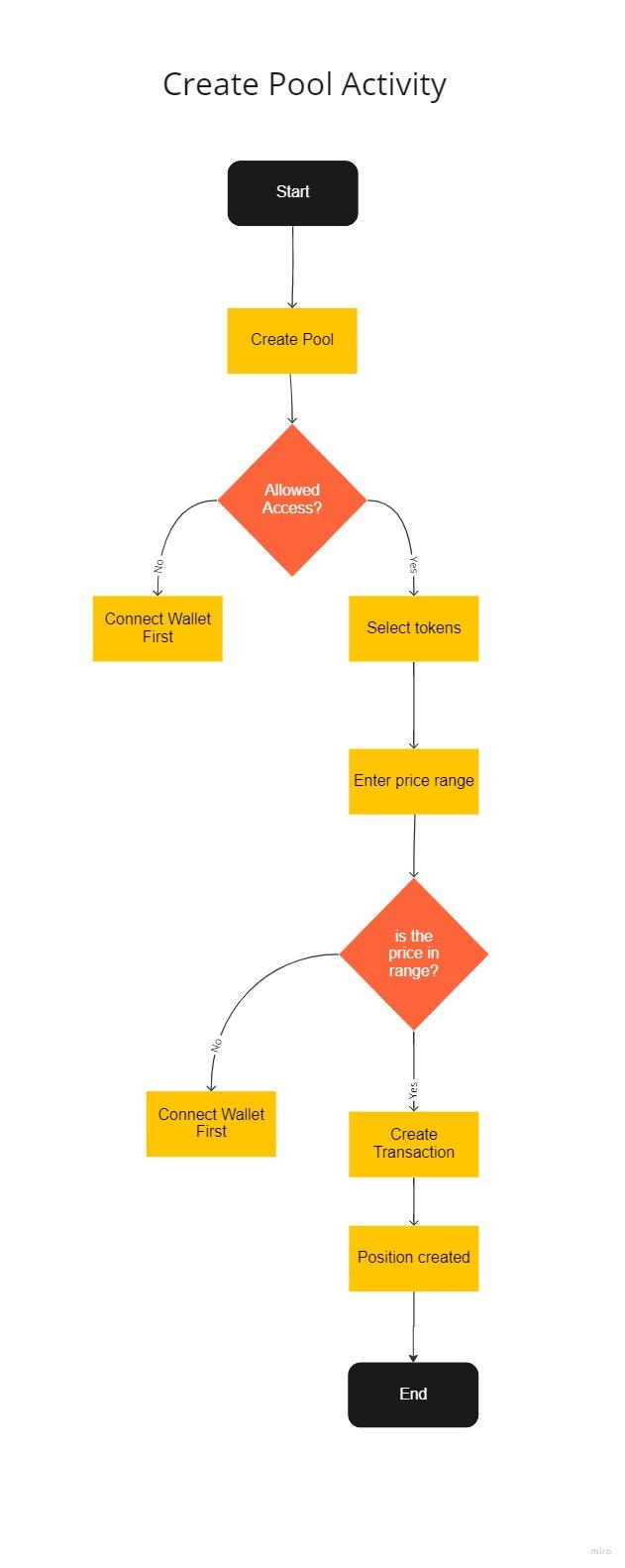
### Pool Activity

Diagram

Description automatically generated

*Figure 17 Activity Diagram 6*

### Create Pool Activity



*Figure 18 Activity Diagram 7*

### Check Position ActivityDiagram Description automatically generated

*Figure 19 Activity Diagram 8*

### Bridge Activity

Diagram

Description automatically generated

*Figure 20 Activity Diagram 9*

### Bridge Liquidity Activity

Diagram

Description automatically generated

*Figure 21 Activity Diagram 10*

## Sequence Diagrams:

### Connect Wallet

Diagram

Description automatically generated

*Figure 22 Sequence diagram 1*

### Swap Token

Diagram

Description automatically generated

*Figure 23 Sequence diagram 2*

### Token Creation

Diagram

Description automatically generated

*Figure 24 Sequence diagram 3*

### Pool creation

Diagram

Description automatically generated

*Figure 25 Sequence diagram 4*

### Check position

Diagram

Description automatically generated

*Figure 26 Sequence diagram 5*

### Bridge swap

Diagram

Description automatically generated

*Figure 27 Sequence diagram 6*

### Bridge pool

Diagram

Description automatically generated

*Figure 28 Sequence diagram 7*

# SYSTEM TESTING

## TEST CASES

### TC01: Connect Wallet

| **Test case**  **ID** | **Test Title** | **Test cases**  **Description** | **Pre-conditions** | **Test steps** | **Expected result** | **Actual result** | **Status** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| TC-01 | Connect wallet | Connecting crypto currency wallets to the decentralized exchange using private key. | 1. Users must have a crypto currency wallet to the exchange.  2. Wallets must be supported by the decentralized exchange. | 1. the user must add a correct and valid private key and password. | 1. Confirmation of wallet being connected. | 1. Confirmation of wallet being connected. | Pass |

*Table 31 TC-01*

### TC02: Disconnect Wallet

| **Test case**  **ID** | **Test Title** | **Test cases**  **Description** | **Pre-conditions** | **Test steps** | **Expected result** | **Actual result** | **Status** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| TC-02 | Disconnect wallet | Disconnecting your crypto currency wallet from the decentralized exchange. | 1. Users must have their wallet connected to the exchange. | 1. Users should click on their wallet address button.  2. User will then click the disconnect button. | 1. User’s wallet will be disconnected.  2. Connect wallet button will be visible again. | 1. User’s wallet will be disconnected.  2. Connect wallet button will be visible again. | Pass |

*Table 32 TC-02*

### TC03: Add Liquidity

| **Test case**  **ID** | **Test Title** | **Test cases**  **Description** | **Pre-conditions** | **Test steps** | **Expected result** | **Actual result** | **Status** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| TC-03 | Add liquidity | Add liquidity to an existing pool. | 1. Users should connect their wallet first.  2. Users should have balance in their wallet. | 1. user will select an existing pool.  2. Users will add the amount they want to provide as liquidity. | 1. Confirmation of liquidity being added to the pool will be shown.  2. liquidity will be added to the pool. | 1. Confirmation of liquidity being added to the pool will be shown.  2. liquidity will be added to the pool. | Pass |

*Table 33 TC-03*

### TC04: Remove Liquidity

| **Test case**  **ID** | **Test Title** | **Test cases**  **Description** | **Pre-conditions** | **Test steps** | **Expected result** | **Actual result** | **Status** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| TC-04 | Remove liquidity | Liquidity Providers can remove their liquidity from the pool. | 1. Users must have added liquidity to that pool in the first place.  2. Users connect their wallet to retrieve their liquidity. | 1. Users must select the correct pool .  2. users must click the remove liquidity button. | 1. Confirmation of liquidity being removed from the pool will be shown.  2. liquidity will be removed from the pool and the balance will be transferred to your wallet. | 1.Confirmation of liquidity being removed from the pool will be shown.  2. liquidity will be removed from the pool and the balance will be transferred to your wallet. | Pass |

*Table 34 TC-04*

### TC05: Swap Token

| **Test case**  **ID** | **Test Title** | **Test cases**  **Description** | **Pre-conditions** | **Test steps** | **Expected result** | **Actual result** | **Status** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| TC-05 | Swap Token | Swapping tokens means that you can swap your tokens with another token. | 1. Users must connect their wallet to the exchange.  2. Users should have balance or assets in their wallet.  3. The token must be supported by the exchange. | 1. User will add the amount of tokens he wants to swap.  2. user will select the tokens he wants to swap to. | 1. Confirmation of the token being swapped will be shown.  2. Swapped tokens will be transferred to the wallet. | 1. Confirmation of the token being swapped will be shown.  2. Swapped tokens will be transferred to the wallet. | Pass |

*Table 35 TC-05*

### TC06: Check Transaction History

| **Test case**  **ID** | **Test Title** | **Test cases**  **Description** | **Pre-conditions** | **Test steps** | **Expected result** | **Actual result** | **Status** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| TC-06 | Transaction history | Users can check the transaction history of pools in the decentralized exchange. | 1. Users must navigate to pools page. | 1. User will select the specific pool that he wants to check transaction history of. | 1. Top pools will be shown to the users. | 1. Top pools will be shown to the users. | Pass |

*Table 36 TC-06*

### TC07: Add Liquidity to a Pool with Insufficient Funds

| **Test case**  **ID** | **Test Title** | **Test cases**  **Description** | **Pre-conditions** | **Test steps** | **Expected result** | **Actual result** | **Status** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| TC-07 | Adding liquidity to a pool with insufficient funds | Add liquidity to an existing pool with insufficient funds. | 1. Users must have their wallet connected to the exchange. | 1. User will select the specific pool to which he wants to provide liquidity.  2. User will enter the amount of funds that he wants to provide as liquidity to a pool. | 1. Confirmation of liquidity being added to the pool will be shown.  2. liquidity will be added to the pool. | 1.Error message indicating insufficient funds will be shown to the user. | Fail |

*Table 37 TC-07*

### TC08: Removing more liquidity than present

| **Test case**  **ID** | **Test Title** | **Test cases**  **Description** | **Pre-conditions** | **Test steps** | **Expected result** | **Actual result** | **Status** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| TC-08 | Removing more liquidity | Attempt to remove more liquidity than is available in a pool. | 1. Users must have their wallet connected to the exchange.  2. Users must have added liquidity to that pool in the first place. | 1. Users must select the correct pool .  2. the user must click the remove liquidity button. | 1. Confirmation of liquidity being removed from the pool will be shown.  2. liquidity will be removed from the pool and the balance will be transferred to your wallet. | 1. Error message indicating insufficient liquidity in the pool will be shown to the user. | Fail |

*Table 38 TC-08*

### TC09: Swapping Unsupported Token

| **Test case**  **ID** | **Test Title** | **Test cases**  **Description** | **Pre-conditions** | **Test steps** | **Expected result** | **Actual result** | **Status** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| TC-09 | Unsupported token | An attempt to swap to a token that is not supported by the exchange. | 1. Users must have their wallet connected to the exchange.  2. users must have a balance or enough funds in their wallet to make the swap. | 1. User will add the amount of tokens he wants to swap.  2. user will select the tokens he wants to swap to. | 1. Confirmation of the token being swapped will be shown.  2. Swapped tokens will be transferred to the wallet. | 1. Error message indicating that the token is not supported on the exchange to the user. | Fail |

*Table 39 TC-09*

### TC10: Check GAS Fee

| **Test case**  **ID** | **Test Title** | **Test cases**  **Description** | **Pre-conditions** | **Test steps** | **Expected result** | **Actual result** | **Status** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| TC-10 | Check GAS fee | Checking GAS fee to make a particular transaction in the our decentralized exchange | 1. Users must have their wallet connected to the exchange. | 1. User must type the amount of the transaction.  2. Users will select the token they want to swap to. | 1. GAS fee will be calculated and shown to the user on run time. | 1. GAS fee will be calculated and shown to the user on run time. | Pass |

*Table 40 TC-10*

### TC11: Withdraw fee

| **Test case**  **ID** | **Test Title** | **Test cases**  **Description** | **Pre-conditions** | **Test steps** | **Expected result** | **Actual result** | **Status** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| TC-11 | Withdraw fee | Liquidity providers can withdraw the profit they made from their liquidity in a pool. | 1. Users must have their wallet connected to the exchange.  2. the user must have to be a liquidity provider. | 1. Users will select the pool from where they want to withdraw profit.  2. the user will press the withdraw button. | 1. Confirmation of profit being withdrawn will be provided.  2. The Withdrawal fee will be transferred to the wallet. | 1. Confirmation of profit being withdrawn will be provided.  2. The Withdrawal fee will be transferred to the wallet. | Pass |

*Table 41 TC-11*

### TC12: Check Price of a Token in a pool

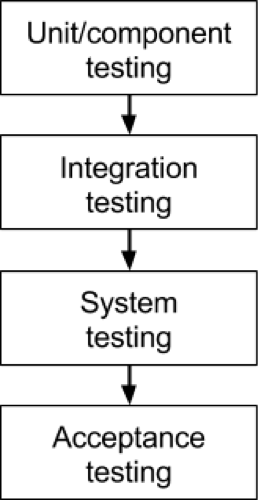
| **Test case**  **ID** | **Test Title** | **Test cases**  **Description** | **Pre-conditions** | **Test steps** | **Expected result** | **Actual result** | **Status** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| TC-12 | Check price of token in a pool | Users can check the price of a token in a particular pool. | 1. Users must navigate to the pool webpage. | 1. Users will select a particular pool. | 1. Prices of the token in the pool will be shown to the user. | 1. Prices of the token in the pool will be shown to the user. | Pass |

*Table 42 TC-12*

## Unit / Integration / Acceptance Testing

Following are the levels of testing that we will be implementing on our project.

* Unit testing
* Integration testing
* System testing
* Acceptance testing.



### Unit Testing

Unit testing in the context of our decentralized exchange will be done by testing small components of our project. For example., testing smart contracts that handle the functionalities like buy and sell funds, testing frontend of our exchange that our users interact with, and more. Unit testing can help ensure that the DEX is working correctly and efficiently, and can help catch bugs and other issues early on in the development process, saving time and resources in the long run.

### Integration Testing

Integration testing in the context of our decentralized exchange will be done at the time of integration of our frontend and backend to make sure that the components of backend work as desired when linked with the frontend.

### System Testing

In the context of developing a decentralized exchange (DEX), system testing will be used to test the DEX as a complete system. This could involve testing the DEX's functionality, performance, reliability, and other characteristics to ensure that it is working as intended. System testing can be an extensive and time-consuming process, but it is an important step in the software development process to ensure that the DEX is ready for deployment. It can help identify any issues or bugs that may not have been caught during earlier stages of testing, and can give developers confidence that the DEX is ready for use by the public.

### Acceptance Testing

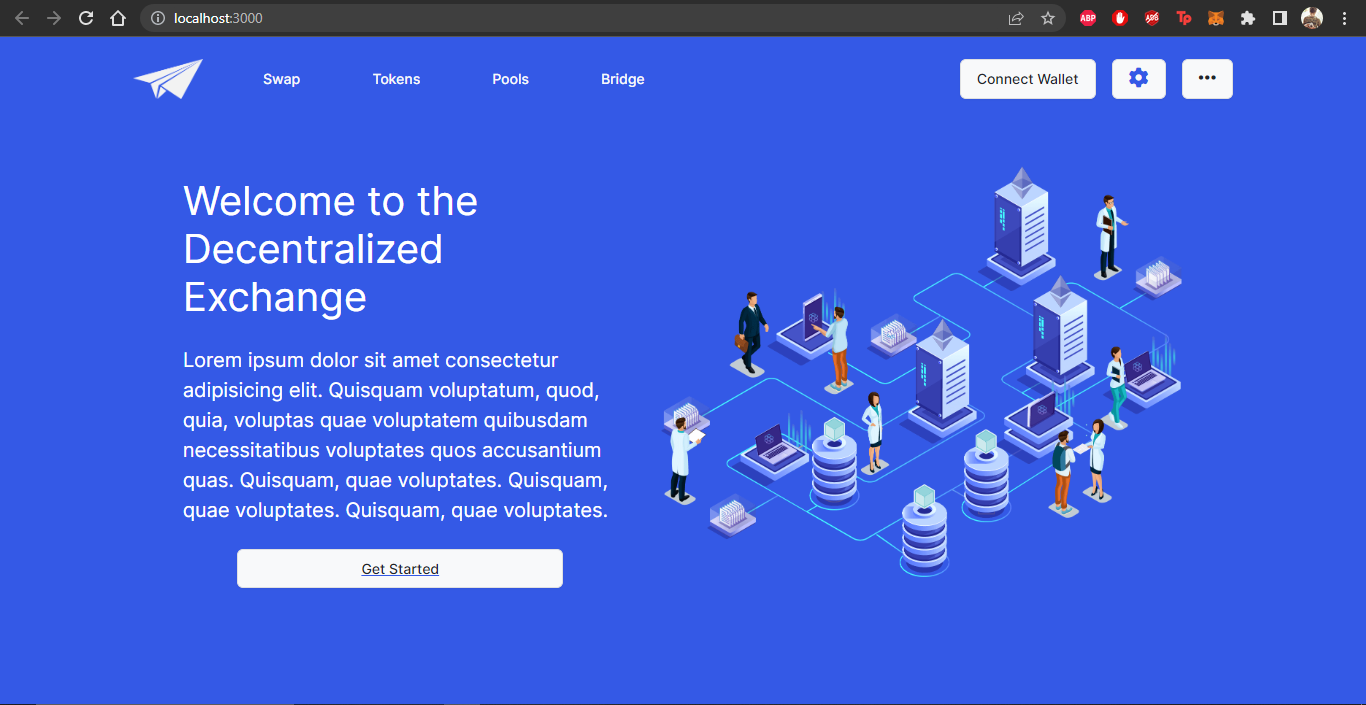
In the context of developing a decentralized exchange (DEX), acceptance testing can be used to evaluate the DEX from the perspective of the end user. This could involve testing the DEX's functionality, usability, and overall user experience to ensure that it meets the needs and expectations of the user. Acceptance testing can be particularly important for decentralized exchanges, as they often involve complex systems that need to be easy to use and understand for a wide range of users.

# Application Front-End

This section contains screenshots of all the working important features of decentralized exchange.

## Web-interface

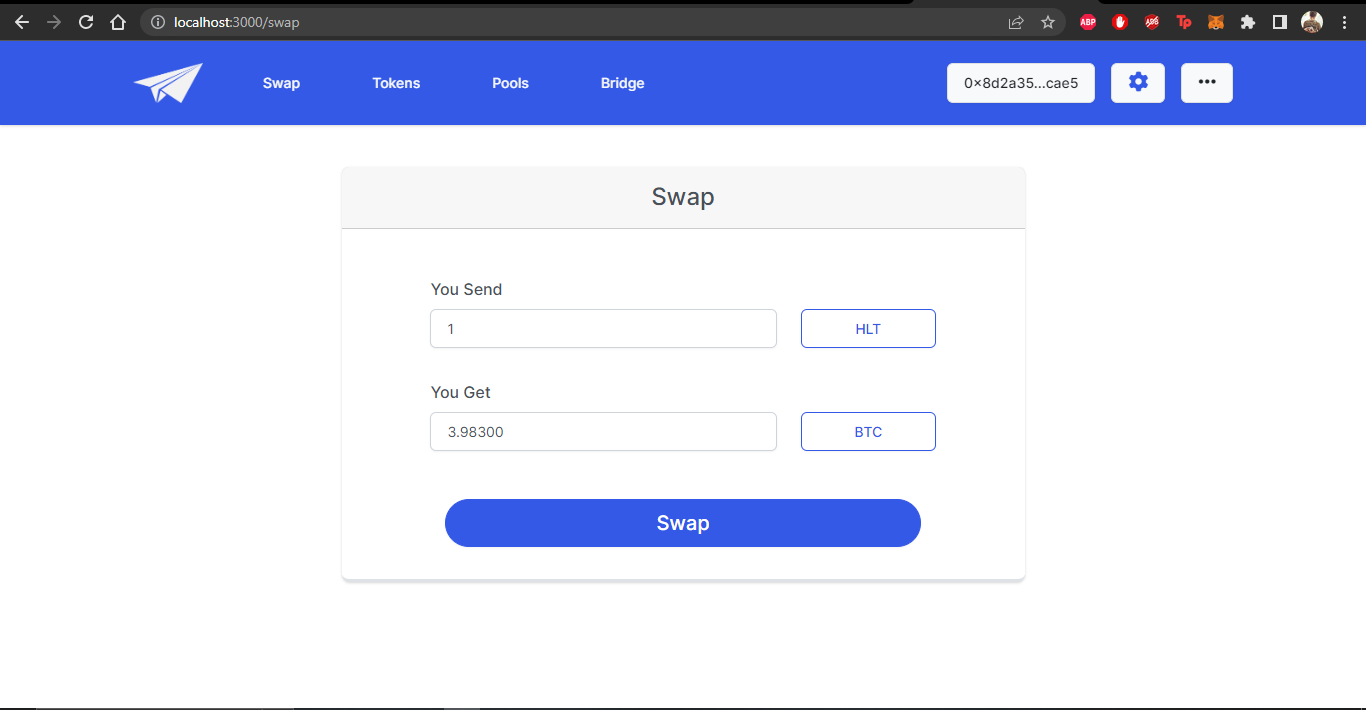
### Landing Page:



*Figure 29 Landing Page*

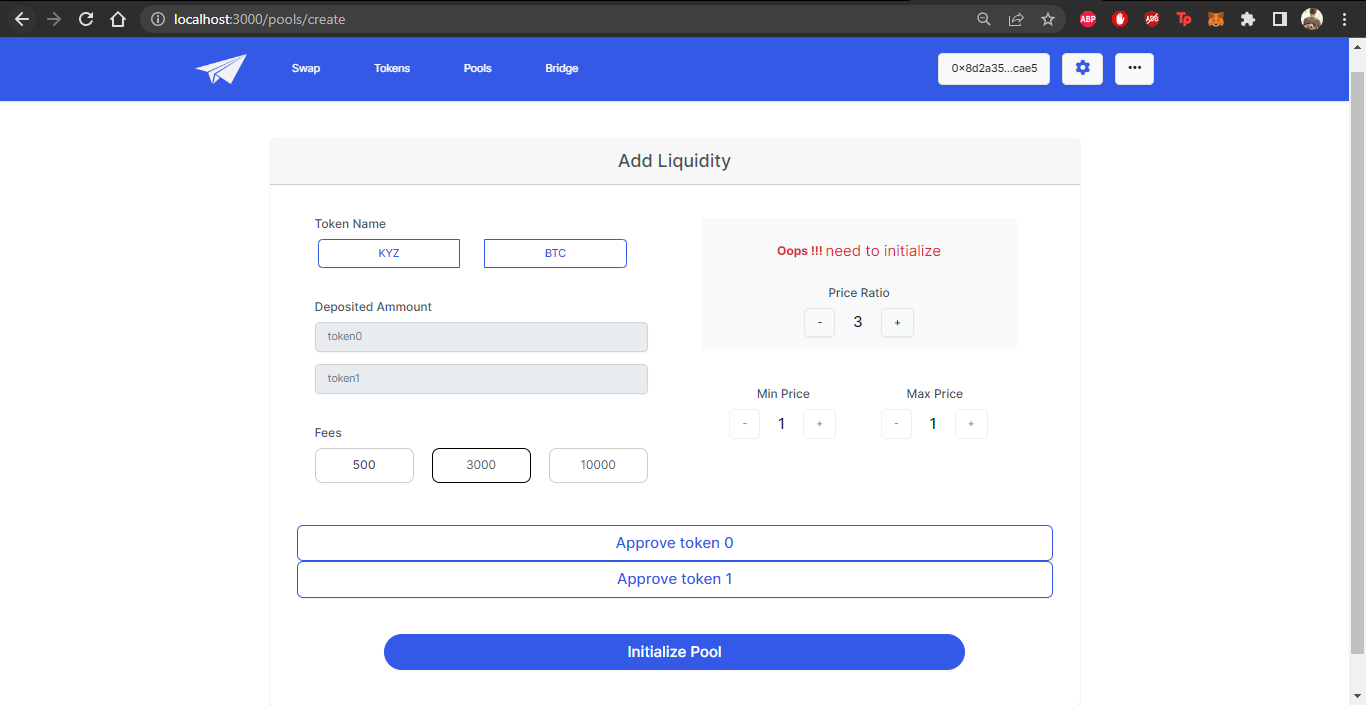
This is our landing page. It is simple and contains some information about our project. It has a function navigation bar through which you can navigate to any required page. Once you press the “Get Started” button. It Takes you to our next page.

### Swap Page:

*Figure 30 SwapPage*

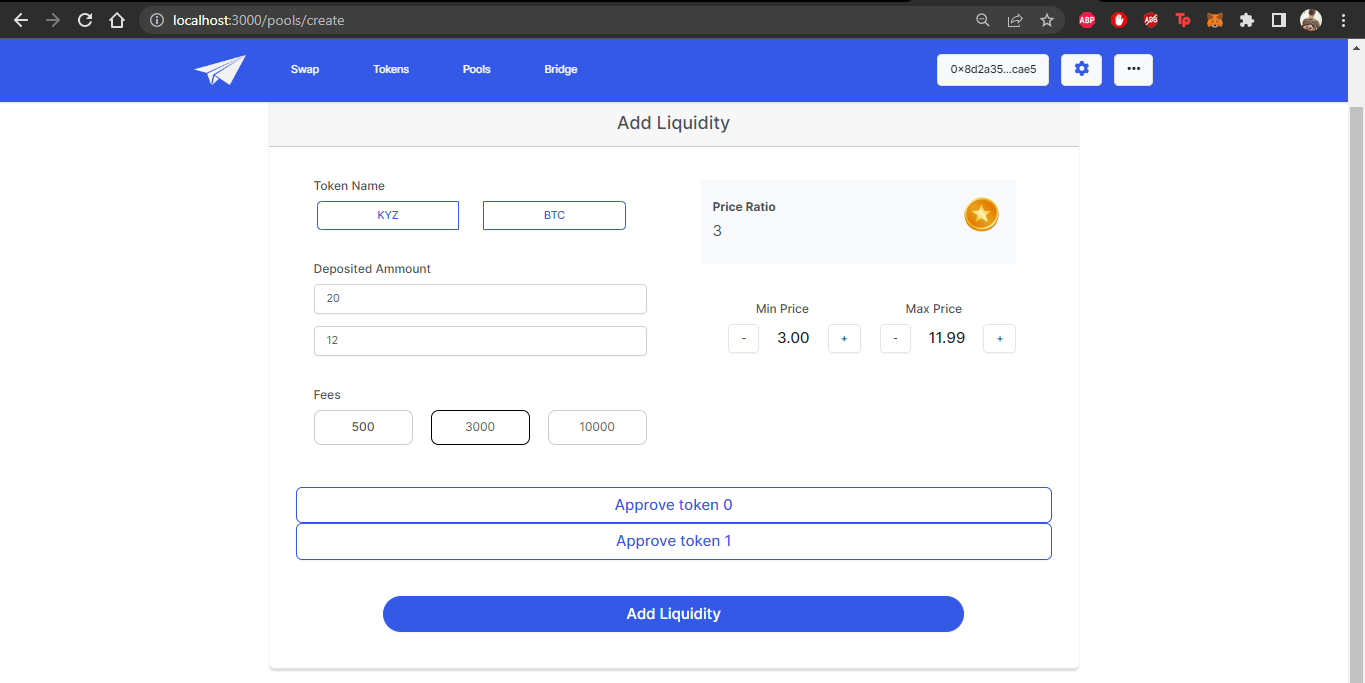
Now we are on the swap page. We can select the tokens from existing tokens and check if their pool exists. If the pool exists then after selecting the pool we can write either the amount we want to send or the amount we want to get in their respective fields and the Quoter smart contract will estimate the return amount of the other according to the liquidity in the pool. After checking the estimated price, we need to connect our wallet to swap the tokens. When we press the “Swap” button, the router contract will make the transaction on the pool.

### Create Pool Page:



*Figure 31 Initialize Pool Page*

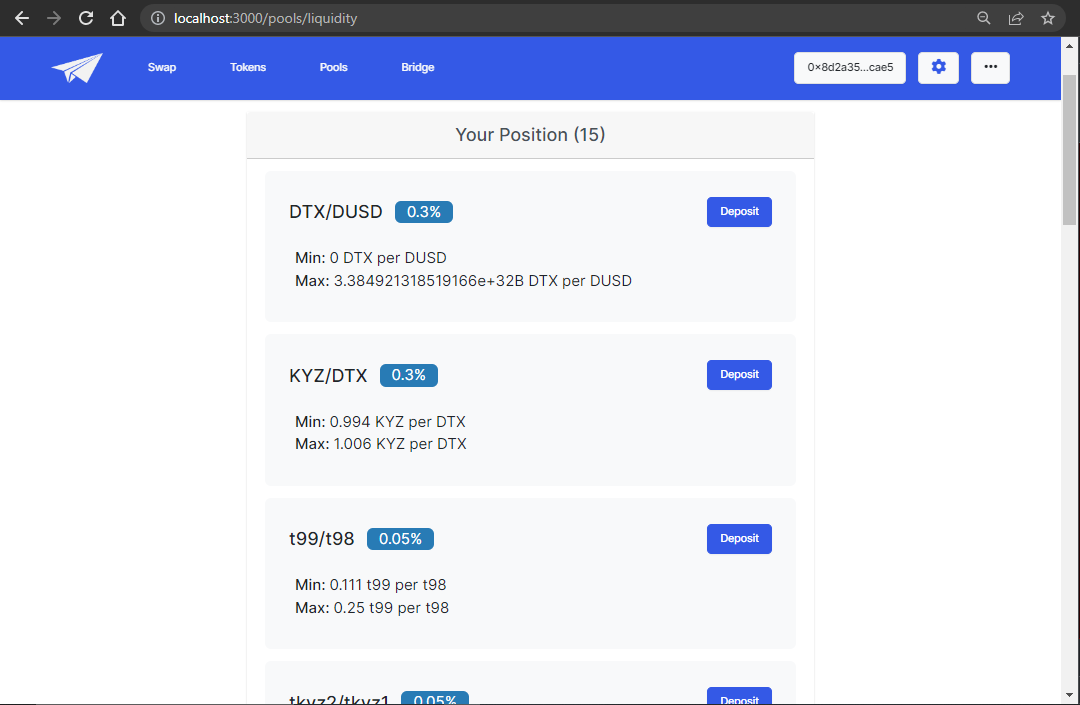
In order to make transactions on the Swap page, we need to have pools and liquidity. For that purpose, we have an Add Liquidity page in our decentralized exchange. In the first screenshot we deal with the case when the pool between the two selected tokens does not exist. It shows a warning of “need to initialize” in that case. Then in order to initialize the pool you must select two tokens such that no pool exists between them. Then you need to set the price ratio of token1 to token2. Then you need to set the fees of the pool. if you notice, when the pool is not initialized, all fields other than the mentioned ones are disabled.



*Figure 32 Add LiquidityPage*

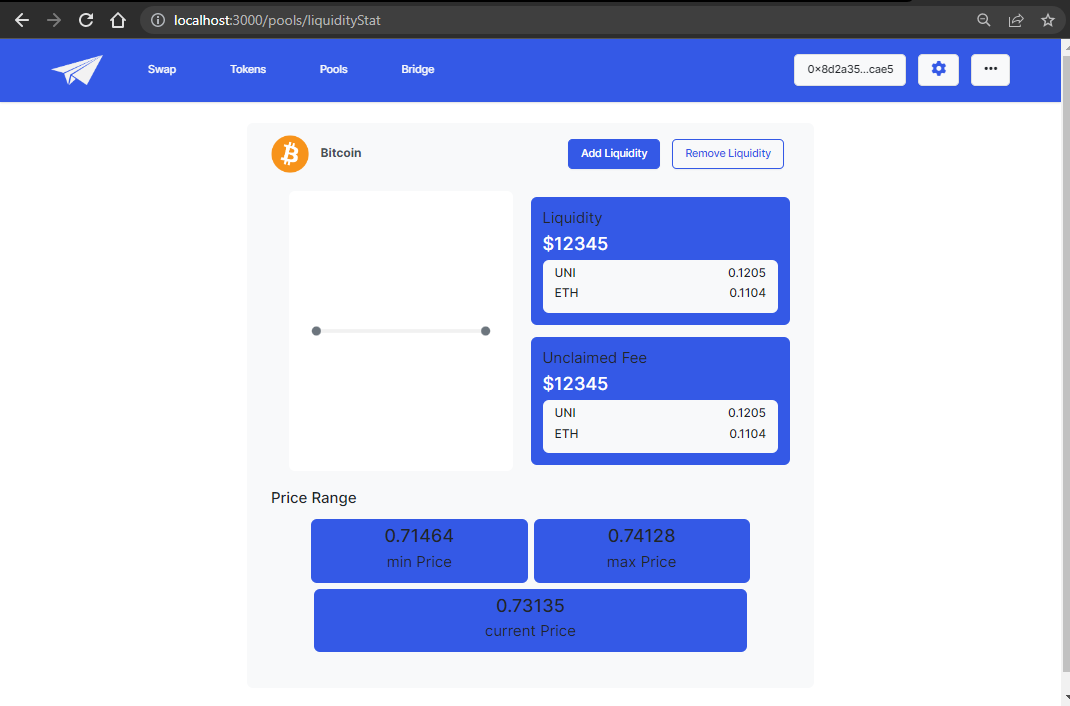
When you initialize the pool by clicking the “Initialize Pool”, all other fields are enabled. And the second case starts where you add liquidity to the pool. You select the amount for token1 and token2 you want to provide as liquidity and set tick spacing. After making sure that the tokens are approved to the “NFTManager” contract you press the “Add Liquidity” button and the liquidity is added and ticks are set to their possible near amounts to the added ones.

### Pool Position Page:

 *Figure 33 Positions Page*

This page inlists all the positions created in a pool. Each position shows the pool tokens and the fees that we set earlier and min and max ticks. You can get further details about the positions by clicking on the deposit button.

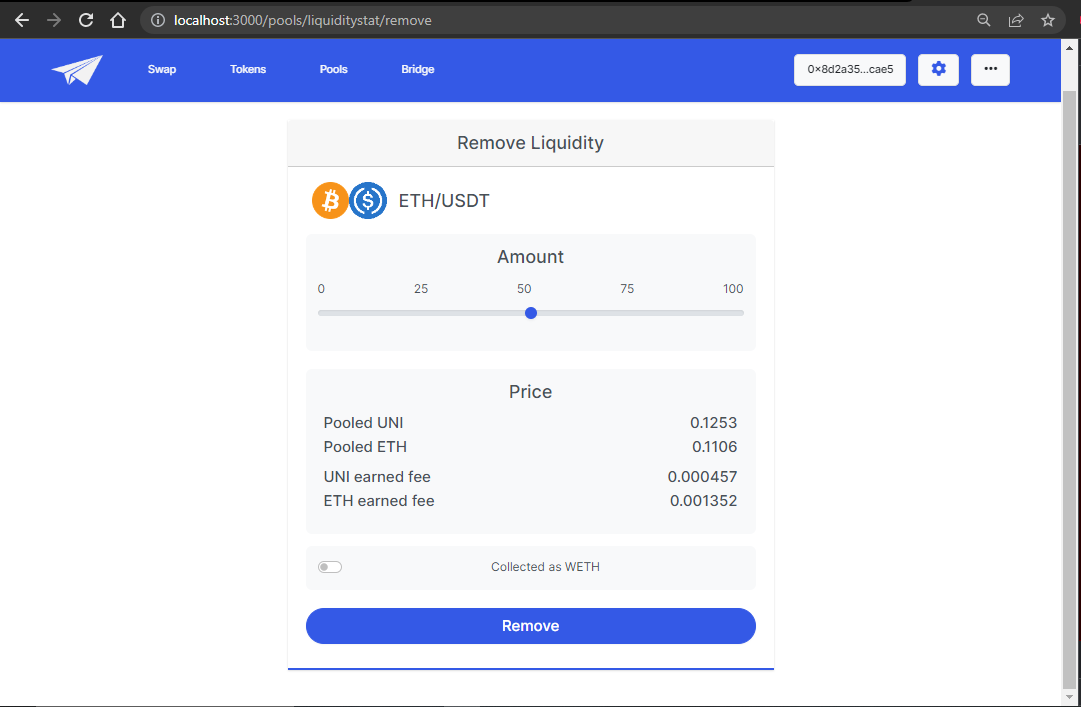
### Check Liquidity Page:



*Figure 34 Stats Page*

After clicking the “deposit” button, you get to the check position page. Here you can check the details of a position you created. You can press the “add liquidity” or “remove liquidity” button to add or remove liquidity respectively.

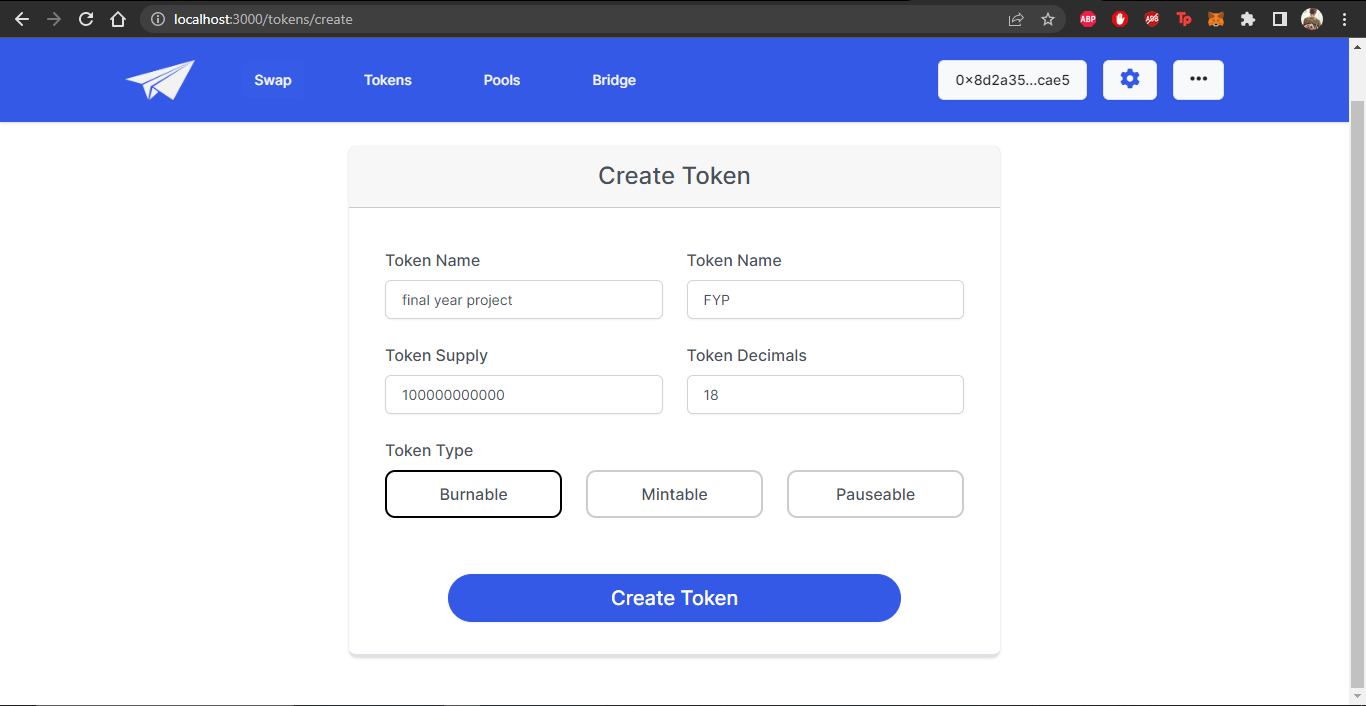
### RemoveLiquidity Page:



*Figure 35 Remove Liquidity Page*

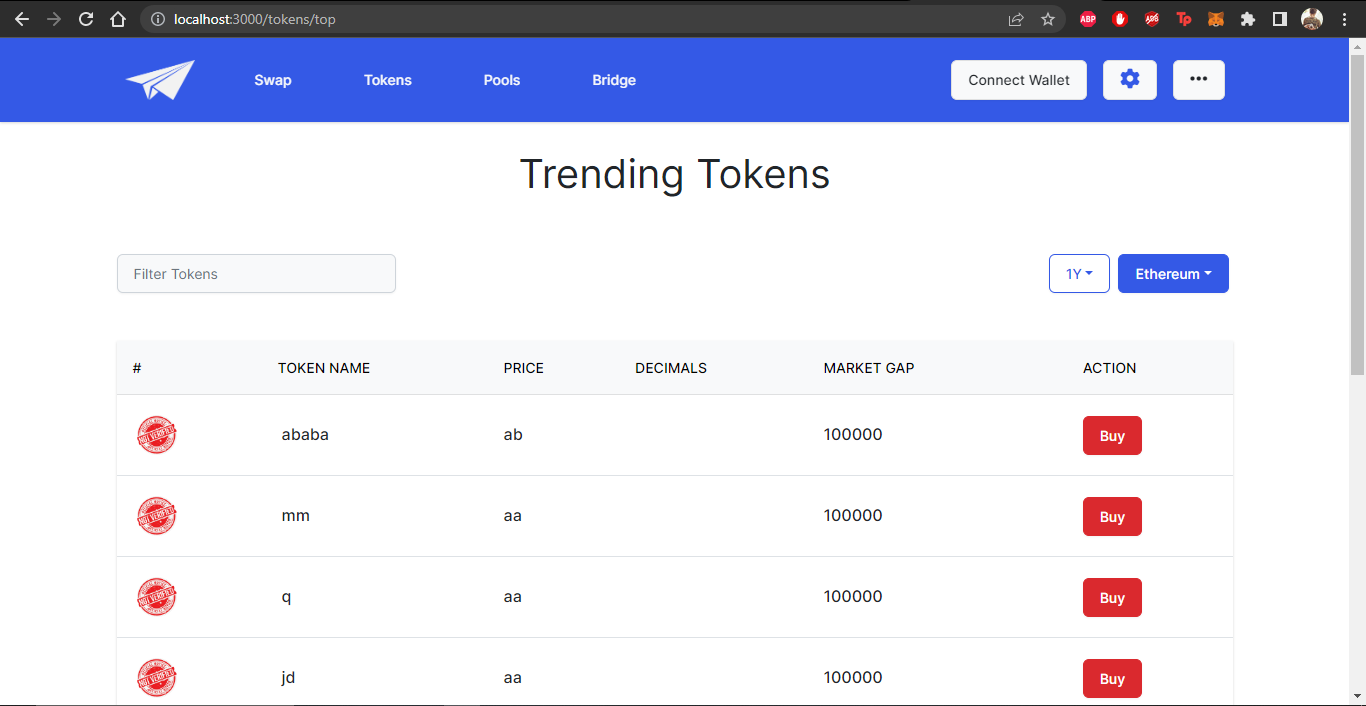
Once you click the “Remove Liquidity” button present on the previous page. you are navigated to this page. Here by pressing the Remove button, you can take back the amount you provided as liquidity.

### Create Token Page:

*Figure 36 Create Token Page*

Our DEX provides its end users with the feature of creating their own tokens. They just need to specify the name, symbol, total supply, number of decimals and the type of the token. Then by clicking the Create token button, they will be provided with the address to the created token which can later be used in the liquidity pools.

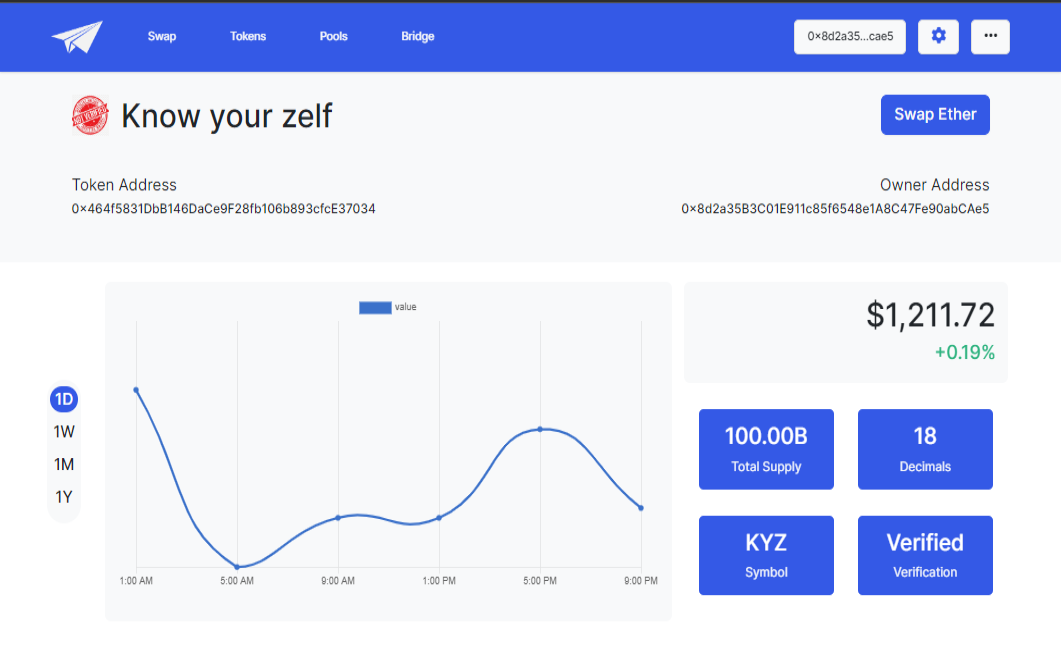
### Trending Tokens Page:



*Figure 37 Top Token Page*

If you want to discover other newly created tokens or the token you created yourself and other famous tokens in order to buy them, you can head to the trending tokens page.

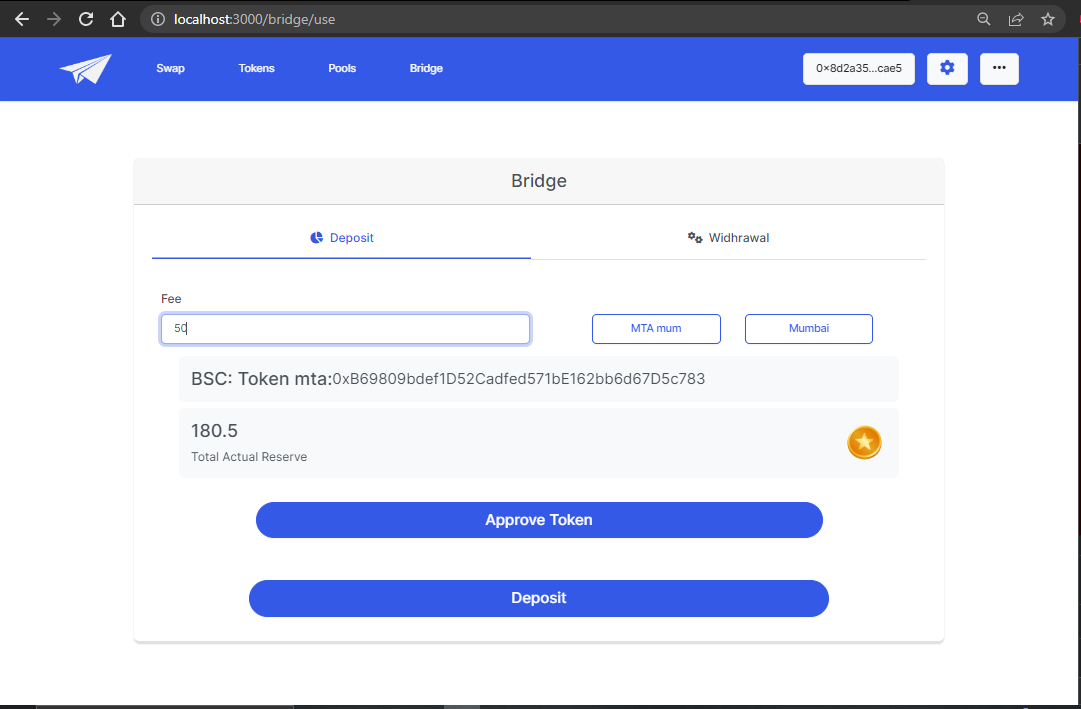
### Token Stats Page:



*Figure 38 Token Stats Page*

You can check the stats such as total supply, current price, number of decimals and their symbol and check their verification status. Moreover you can also get the transaction history of each token on our DEX without having the need to go to the blockchain scans.

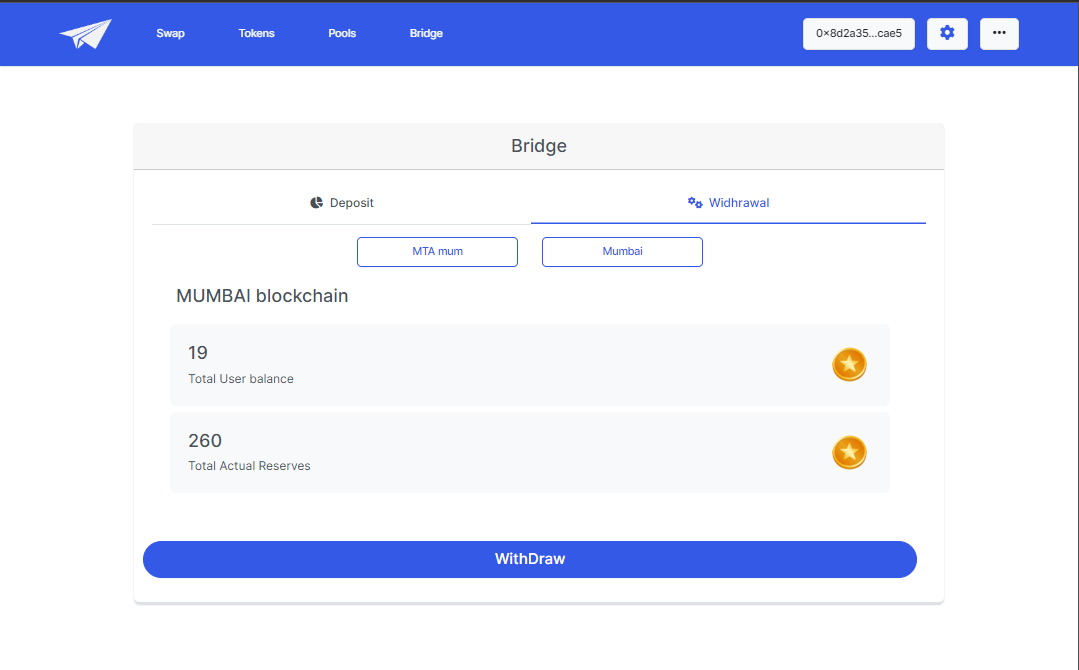
### Deposit Amount to Bridge:



*Figure 39 Deposit Token Page*

The bridge page has two phases, one to deposit the token to the blockchain. the user must select the token and the blockchain that the token is created on. Once the user presses the “deposit” button, the token is then locked inside the bridge smart contract. The total amount of reserves of the token created in the other blockchain are shown to the user .

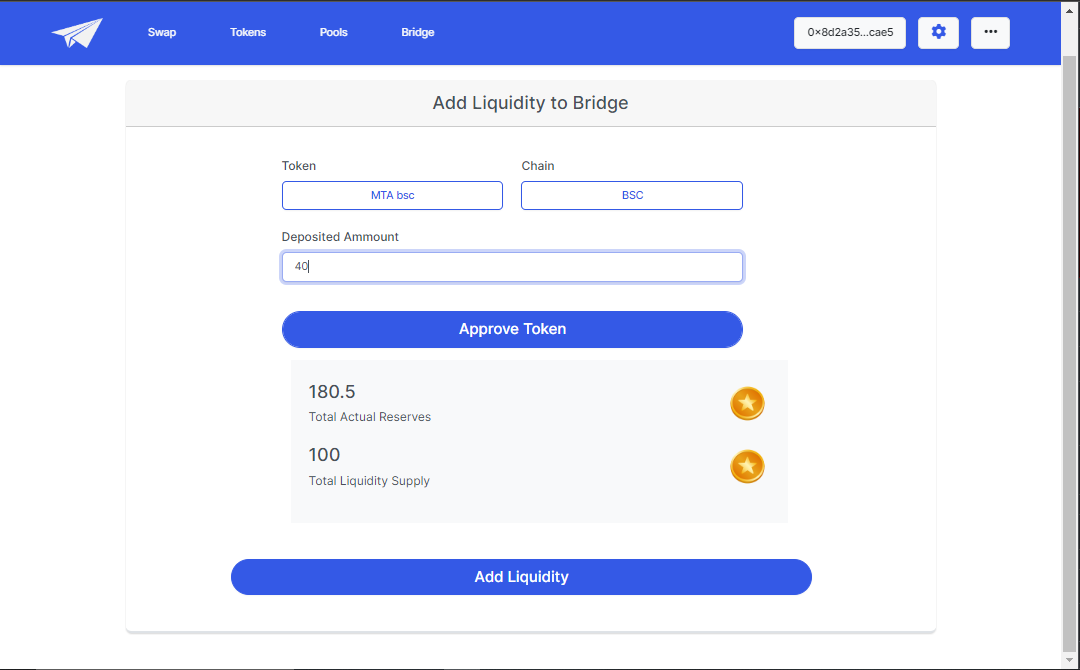
### Withdraw Amount from the Bridge:



*Figure 40 Withdraw Token Page*

In the other phase, the user specifies the token and the blockchain on which the token was created. When the person presses the “withDraw” button, The Bridge smart contract verifies if the person making the withdrawal request is actually the one that deposited the token by comparing their wallet’s private key hash. When the verification is done the user is asked to sign the transaction. And that is how the transaction from one blockchain to another blockchain is made through a bridge.

### Add Liquidity to Bridge:

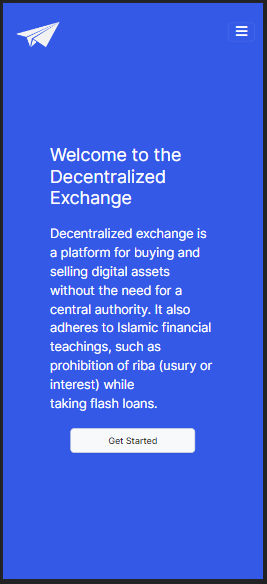


*Figure 41 Liquidity to Bridge Page*

Our DEX also provides its users with the feature of adding liquidity to a bridge. Moreover, just like liquidity pools, users can also check their bridge liquidity positions.

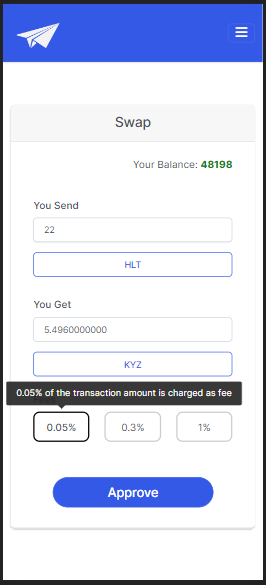
## Mobile-interface

### Landing Page:

****

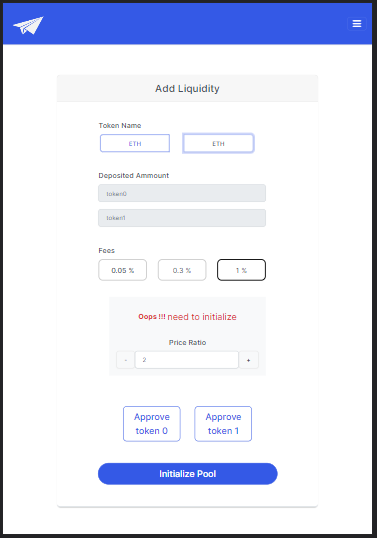
*Figure 42 Landing Page*

### Swap Page:

****

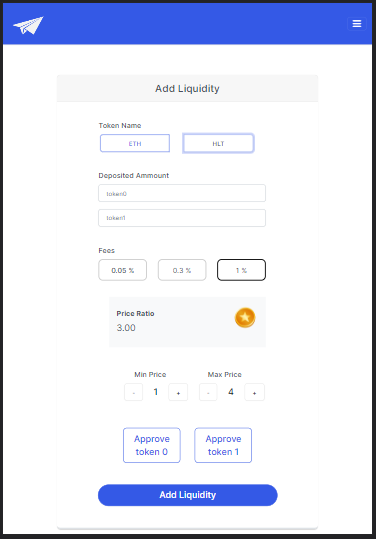
*Figure 43 SwapPage*

### Create Pool Page:

****

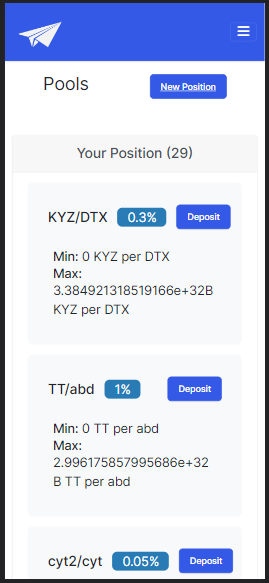
*Figure 44 Initialize Pool Page*

### Pool Position Page:

****

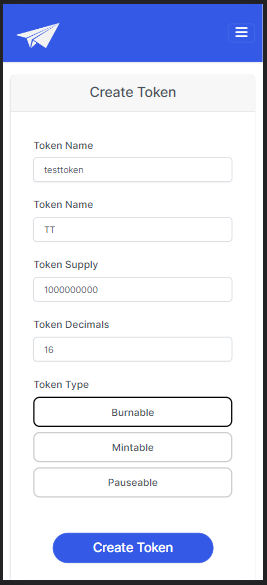
*Figure 45 Add LiquidityPage*

### Check Liquidity Page:

****

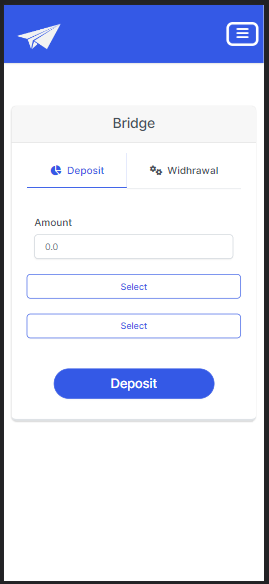
*Figure 46 Positions Page*

### Create Token Page:

****

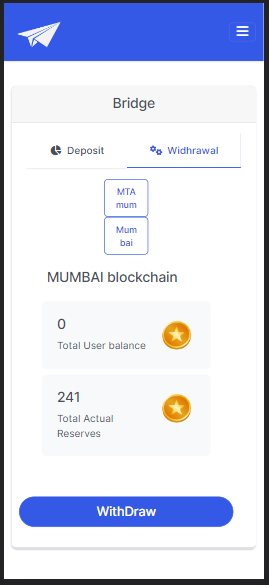
*Figure 47 Create Token Page*

### Deposit Amount to Bridge:

****

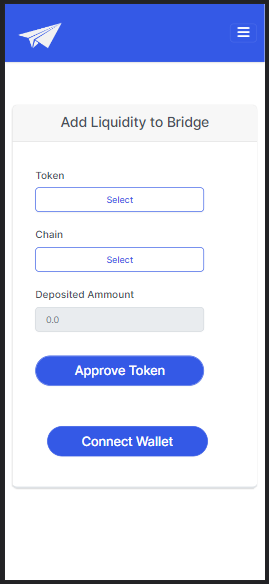
*Figure 48 Deposit Token Page*

### Withdraw Amount from Bridge:

****

*Figure 49 Withdraw Token Page*

### Add Liquidity To Bridge:

****

*Figure 50 Liquidity to Bridge Page*

# Conclusion

## Problems faced and lessons learned

### Problems faced:

As blockchain and cryptocurrencies are new emerging fields in Pakistan, it was difficult for us to find learning resources and learn new tools and technologies in a short amount of time. During the FYP-2 it got difficult for us to do research, development, and writing FYP-2 report along with our studies.

Following are some of the problems we faced during our FYP-2:

* Learning new tools and technologies for blockchain.
* Communication within group members.
* Managing regular classes and their homework with FYP.
* Finding authentic learning resources.
* Creating an intuitive and user-friendly interface for seamless user interaction.
* Managing price fluctuations and minimizing slippage in trades.
* Building logics for the bridge and implementing it on our project. It was a feature that was new to us, and we faced many difficulties coding and integrating it with our exchange.

### Lessons Learned:

Following are some of the major lessons we learned during our FYP-2:

* Time management.
* Explored a lot of new tools and technologies related to blockchain development.
* Learned the concepts related to blockchain and cryptocurrency.
* We understood the working of a decentralized system and a decentralized exchange.

## Project Summary

It is a decentralized exchange that allows its user to trade cryptocurrency and provide liquidity in liquidity pools. It is built on blockchain and uses Automated Market Maker (AMM) technology to make transactions for its users. It instead relies on liquidity pools to facilitate trades. It also provides its users to create their own tokens and enables them to import their assets from other blockchains or export them to other blockchains using bridge. It does not have a central authority or an intermediary that can interfere with the transactions or user’s assets.

## Future Work

In the future we plan on providing new features like customer support, providing excellent customer support is crucial for any exchange, and this is especially true for DEXs. Furthermore, we will be implementing DAO in our decentralized Exchange. DAO stands for Decentralized Autonomous Organization. It is a type of organization that operates based on smart contracts and blockchain technology, allowing for decentralized decision-making and governance by its community members, without the need for a centralized authority. We will be implementing many new and innovative features on our DEX in order to stay ahead of the competition. This could include new trading tools, unique offerings, or other features that set your DEX apart from others on the market.

# References

[1]. Uniswap. (n.d.). Uniswap Documentation. Retrieved from

<https://docs.uniswap.org>

[2]. SushiSwap. (n.d.). SushiSwap Documentation. Retrieved from

<https://dev.sushiswap.fi>

[3]. PancakeSwap. (n.d.). PancakeSwap Documentation. Retrieved from

<https://docs.pancakeswap.finance>

[4]. 1inch. (n.d.). 1inch Documentation. Retrieved from

<https://docs.1inch.io>

[5]. Curve Finance. (n.d.). Curve Finance Documentation. Retrieved from

<https://curve.readthedocs.io>

[6]. Balancer. (n.d.). Balancer Documentation. Retrieved from

<https://dev.balancer.fi>

[7]. Quickswap. (n.d.). Quickswap Documentation. Retrieved from

<https://docs.quickswap.exchange>

[9]. Aave. (n.d.). Aave Documentation. Retrieved from

<https://docs.aave.com>

[10]. dYdX. (n.d.). dYdX Documentation. Retrieved from

<https://docs.dydx.exchange>

[11]. MakerDAO. (n.d.). MakerDAO Documentation. Retrieved from

<https://docs.makerdao.com>

[12]. ApeSwap. (n.d.). ApeSwap Documentation. Retrieved from

<https://apeswap.finance/docs>

