DEX

## Problem statement

* True power of blockchain resides in its decentralization ability. As the field is growing and newer cryptocurrencies are emerging in the market with their own unique use cases, people are investing heap of money into them. An exchange allows people to trade their purchased cryptocurrencies or tokens, there are many centralized exchanges that provide this service however they have many flaws and due to which it is not very safe to trade cryptocurrencies using these centralized exchanges.

## Goals

* Our goal is to create a decentralized exchange operating where people can create their own liquidity pools or exchange their cryptocurrencies or tokens from existing liquidity pools.

## Stakeholders

* Users: These are the people who use the platform to buy and sell cryptocurrencies.
* Developers: These are the individuals or teams responsible for building and maintaining the decentralized exchange platform.
* Investors: These are the people who have invested money into the development and operation of the decentralized exchange.
* Community: This includes the broader group of users, developers, and other individuals who are interested in or actively involved in the decentralized exchange.
* Regulators: In some cases, regulatory bodies may also be stakeholders in a decentralized exchange, as they have a role in ensuring that the platform complies with relevant laws and regulations.

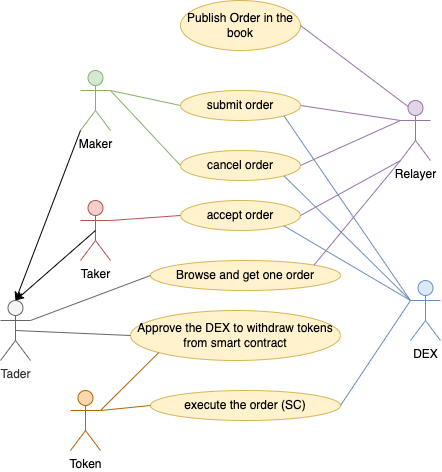
## Restrictions/Rules

* Regulatory: Some countries have strict regulations on cryptocurrency exchanges, and decentralized exchanges may be subject to these regulations as well. This could include restrictions on the types of cryptocurrencies that can be traded on the platform, as well as requirements for reporting and compliance.
* Security: To protect against security breaches and fraud, decentralized exchanges may impose restrictions on certain activities, such as withdrawing large amounts of funds or trading with suspicious accounts.
* User experience: In order to provide a smooth and user-friendly experience, decentralized exchanges may impose restrictions on certain features or functions, such as the number of trades that can be made in a given time period.
* Technical limitations: Decentralized exchanges may also be subject to technical limitations, such as the capacity of their servers or the speed of their network, which may restrict certain activities or functions.
* Fees: Liquidity providers earn a swap fee (0.3 % by default) on all trades proportional to their share of the pool. Fees are added to the pool, accrue in real time and can be claimed by withdrawing your liquidity.

## Data

* There is no need to store user data (user email, user name etc.) for a decentralized exchange.
* Liquidity pair’s data is stored on blockchain within factory smart contract, details of such pairs can easily be retrieved from there.
* Trading data is also stored on-chain, these transactions can also be retrieved using web3.js.
* Technical data include data about the operation of the decentralized exchange platform, such as the number of users, the volume of trades, and any technical issues that arise
* Market data include data about the broader cryptocurrency market, such as the prices of different cryptocurrencies, trading volumes, and market trends.

## User stories



Architecture

## Project description

## Functions - inputs/outputs

### Factory Contract

* createPair: Creates a pair for tokenA and tokenB if one doesn't exist already.
* setFeeTo: Takes an address as input which will receive the fees.
* getPair: Returns the address of the pair for tokenA and tokenB, if it has been created, else address(0).
* allPairs: Returns the address of the nth pair (0-indexed) created through the factory, or address(0) if not enough pairs have been created yet.
* allPairsLength: Returns the total number of pairs created through the factory so far.

### Router Contract

* addLiquidity: Adds liquidity to an ERC-20⇄ERC-20 pool.
* addLiquidityETH: Adds liquidity to an ERC-20⇄WETH pool with ETH.
* removeLiquidity: Removes liquidity from an ERC-20⇄ERC-20 pool.
* removeLiquidityETH: Removes liquidity from an ERC-20⇄WETH pool and receive ETH.

## Diagram (flow, system)

## Tech stack

* Programming languages: Solidity, Typescript
* Blockchain platform: Ethereum, BSC

Project Plan

## Breakdown of tasks and time per task:

* UI design for front-end: 3 days
* Backend-integration: 3 days
* Creating default liquidity pairs: 1 day
* Adding initial liquidity to pairs: 1 day
* Updating default token list: 1 day
* Improving UI: 3 weeks
* Adding support for more networks: 3 days/network
* Governance Token: 1 month
* Visualization tools for pairs and tokens: 2 months
* Enabling Bridge to connect multiple blockchains: 2 months
* Smart Contract Audit: 3 weeks

## Tasks assigned to roles/people

### Front-end:

* Nirav Parmar (Primary Contributor)
* Brijesh Kuvadiya & Jatin Jha (Secondary Contributors)

### Back-end:

* Jatin Jha (Primary Contributor)
* Nirav Parmar & Brijesh Kuvadiya (Secondary Contributors)

### Smart contract & Documentation:

* Brijesh Kuvadiya (Primary Contributor)
* Jatin Jha & Nirav Parmar (Secondary Contributors)

## Dependencies

* Yarn
* Node.js
* Typescript
* React

## Gantt chart – simple

Background pattern

Description automatically generated with medium confidence

Smart Contract

## OutletFactory.sol

* Allows users to create Liquidity Pairs

## OutletRouter.sol

* Allows users to manage their liquidity pools and trade their tokens

## OutletToken.sol

* Default token for the platform, will be used as a governance token in future.

How to Run Project

## Install dependencies:

Code: `yarn`

## Launch Platform:

Code: `yarn start`