There are many kind of decentralized exchanges on Ethereum:

- The more simple ones only allow to trade ERC20 tokens
- while the most sophisticated ones allow you to trade more assets, with more advanced trading options like margin trading.

For your first DEX, we will keep it simple. You will soon realize that even a "simple" DEX is not so "simple".

Let's see the different parts we need to build.

Architecture of DEX

A DEX has the same architecture as any other decentralized application on Ethereum:

- User wallet
- Frontend (Web)
- Smart contract (Blockchain)

The logic of the DEX is inside one or several smart contracts. In our case there will be just one smart contract.

A frontend is connected to the smart contract, so that end-user have a nice and easy-to-use interface to the smart contract. This frontend is most often a web frontend, but it can also be a mobile app.

The frontend connects to the user wallet. The user wallet will prompt the user for confirmation before sending any transaction to the smart contract. The user wallet can be any Ethereum wallet, like Metamask. It's better if this Wallet is ERC20-aware, but it doesn't have to.

When you build your DEX, to make it easier, I recommend to assume that users will have Metamask installed. In a production DEX, you can also support other wallets, like Ledger or Trust Wallet.

Full trading sequence

We have 2 traders, Bob and Alice:

- Bob wants to buy 1 ABC token, at a price of up to 2 Ethers
- Alice wants to sell 1 ABC token, for whatever price

This is the whole trading sequence:

- Bob sends 2 Ethers to the DEX smart contract.
- Bob creates a buy limit order (explained later) for a limit price of 2 Ethers, amount of 1 ABC token, and send it to DEX smart contract
- Alice sends 1 ABC token to the DEX smart contract
- Alice creates a sell market order (explained later) for an amount of 1 ABC token, and send it to DEX smart contract
- The smart contract matches Bob and Alice order, and carry out the trade. Bob now owns 1 ABC token and Alice 2 Ethers
- Bob withdraws his 1 ABC token from the DEX smart contract
- Alice withdraws her 2 Ethers from the DEX smart contract

Tada! The trade is finished!

Traded Tokens

It's possible to trade ERC20, ERC721, or even more advanced tokens like ERC1155. It does not impact much the logic of the contract. In all cases, we will only mostly need to interact with the functions to transfer tokens.

For your first DEX, it's better to keep it simple and stick to ERC20 tokens only.

Additionally, you also need to decide what is the quote currency, i.e which asset will be used to quote the price of ERC20. Example: if the quote currency is Ether and the token ABC trades for 2, that means that you need 2 Ethers to buy 1 token.

The first DEXes only offered Ether, but more recent exchanges also quote prices with the Dai stablecoin.

But for your first DEX, again let's keep it simple and use Ether.

Wallet

Before users can trade, they need to transfer their ERC20 tokens / Ether to the smart contract of the DEX.

They will:

- click on a button in the frontend that it will initiate the transfer,
- confirm the transaction with their wallet
- and the Ethers / tokens will be sent at the address of the smart contract

You will also need to have a functionality that allow users to withdraw their tokens / Ether from the DEX smart contract, once they have finished to trade.

In your DEX smart contract, you will need to implement your own ledger to track the ownership of Tokens & Ether. The ledger will be incremented when assets are added, and decremented when assets are withdrawn.

Order types

Traders express their intent to trade by creating an order. An order has several fields:

- Currency (the asset you want to trade)
- Amount: how much you want to trade
- Type: see below

There are 2 main types of orders:

- Limit order, which specify a max / min limit price for buy / sell order.
- Market order, where you agree to whatever price is on the market

Optionally, you can also implement an order cancellation feature. That means that after a trader send a limit order, it can be cancelled, provided it hasn't been executed before.

Orderbook

The orderbook is the core part of the DEX. It:

- Lists all limit orders
- Matches incoming market orders against existing limit orders
- Remove limit orders that were executed

Orderbooks follow a price-time algorithm. When an incoming market order arrive, the orderbook will try to match it with the market order that has the best price. If several limit orders have the same price, the one that was created first get matched in priority.

What if the amount of the market and limit order don't match? Actually, that's what will happen most of the time.

In this case, there are 2 possibilities:

- Case 1: Amount of market order < amount of limit order. In this case the market order will be fully excuted, but the limit order will only be partially executed, with the unexecuted part remaining in the orderbook.
- Case 2: Amount of market order > amount of limit order. In this case the limit order will be fully executed, and the orderbook will continue to try to match the remaining of the market order with other limit orders

Note 1: For case 2, it's actually possible that there is not enough liquidity to match the remaining of the market order. You could decide to disallow partial matching of market order by rejecting any market order that can't be matched entirely.

Note 2: To make your orderbook more simple, you can disallow limit orders that "cross the books", i.e for example a buy order with a limit price so high that it would be matched instantly against limit sell orders on the other side of the orderbook.

Settlement

After a market and a limit order have been matched, you need to complete the transaction by sending the assets to their new owners:

- Tokens need to be transferred from the seller to the buyer
- and Ether needs to be transferred from the buyer to the seller

The trick is that these transfers only take place symbolically. As we discussed before, all the assets are held in the wallet of the DEX smart contract at the moment of the trade. So we only have to update the internal ledger of the DEX smart contract to reflect the asset transfer.

After the trade, the 2 traders will have to use the withdraw function of the wallet of the DEX smart contract to get back their asset.