NFT Market place with custom currency token support using Polygon

Solution details

- The NFT Marketplace will be built using ReactJS for the user interface and interaction.
- The smart contract functionality will be implemented using Solidity and Hardhat for development and Chai for testing.
- The Ethereum network interaction will be handled using ethers.js and web3.js libraries.
- MetaMask will be used as the only form of connecting to a wallet.
    - The NFTs and custom tokens will be deployed and managed on the Polygon network using the Mumbai TestNet.

List of Technologies with version and functionality

1. ReactJS: Version ^18.2.0 - Used to create the user interface for the NFT Marketplace.
2. Tailwind CSS Version 3.0 - Used for the CSS of the user interface.
3. Typescript Version ^4.9.5 - Programming language for writing the user interface
4. Solidity: Version ^0.8.0 - Programming language for developing smart contracts.
5. Hardhat: Version ^2.16.1 - Development environment for compiling, testing, and deploying smart contracts.
6. ethers.js: Version ^6.6.3 - JavaScript library for interacting with Ethereum networks and contracts.
7. web3.js: Version ^4.0.3 - JavaScript library for interacting with Ethereum networks and contracts.
8. Chai: Version 4.3.7 - Testing framework for writing unit tests for Solidity smart contracts.
9. Polygon: Utilizing the Polygon network and Mumbai TestNet for deploying and testing the NFT Marketplace.
10. OpenZepplin: for utilizing useful contracts such as Counters for generating unique ID's, ERC721 for NFTs, Ownable contract which provides a basic access control mechanism

Development environment setup

- Install Node.js and NPM (Node Package Manager).
- Create a new project using create-react-app for the client
- Install the required dependencies, such as ethers.js, web3.js, and other necessary packages.
- Set up the Hardhat development environment for Solidity smart contract development and testing.
- Configure the connection to the Polygon Mumbai TestNet in the Hardhat network configuration.
- Deploy the contract using npx hardhat run ./scripts/deploy.js --network mumbai

Functionalities (Don’t include code snippets or function names)

1. User Login
    - The user will click on a “Connect to Wallet” button which will open up MetaMask, using window.ethereum to check for MetaMask existence and ethers library to connect to MetaMask.
    - If MetaMask does not exist in the browser, the user will receive an error message on the screen notifying the user that he should have MetaMask installed on the browser. using a try-and-catch statement to receive MetaMask error message.
    - The user will enter his MetaMask credentials, and MetaMask will deal with handling the verification and validation of the credentials.
    - If the credentials entered are correct, then the User’s MetaMask wallet will be connected to the marketplace. The user information like his address and other account details will be received from MetaMask by ethers. And information such as his account address will be saved in a Global state store in the frontend.
    - If the credentials are incorrect, then the user will receive an error message.
2. How will the NFT marketplace support custom token
    - First, we need to design a custom token (build on Polygon’s Mumbai TestNet). It should be based upon the ERC-20 contract as it’s the most used token standard on the polygon network.
    - Using solidity and hardhat to deploy it, of course, it should include the necessary functions to mint it, buy, sell, and withdraw and then deploy the contract, remember to save it’s ABI and address so that we can create an instance of it on the frontend and NFT Marketplace contract.
    - Upon deploying the contract, we pass a number of how many tokens we want to mint (initial supply). In the constructor, _mint is used (an ERC-20 function that creates new tokens, assigns them to an address, creates new tokens, and assigns them to an address and increases the total supply). The buy and sell functions currently don’t work as required as we don’t have the token in a liquidity pool, so it doesn’t have any price on it. But for testing purposes, we used 1 MATIC = 1 TOKEN. But it works by transferring funds to the contract and vice versa. And if the owner wants to withdraw funds from the contract, he can use the withdrawFunds function which withdraws money from the contract. The money form is not defined yet (no liquidity pool).
    - Update the NFT marketplace smart contract to include functionality for interacting with the custom token. Modify payment processing logic to use the custom token instead of Matic (Polygon's native token), for example in our NFT marketplace we have two methods in which the user must pay money. 1- MINTING: where the user must pay a minting fee and 2- Buying NFT: where the user must pay for the NFT. To integrate the custom token with these methods, we need to import the custom token address in our NFT marketplace and create an instance of it in the constructor. We will use this instance of the custom token to transfer the custom tokens from one user to another, since we are going to use the NFT marketplace contract to transfer funds from the custom token’s contract we must use the “approve” function provided by the ERC-20 so that the user allows the NFT marketplace to spend his tokens in order to mint, purchase,

etc.. Then we use the instance of the custom token and invoke the "transferFrom" method to spend his tokens.
- In the front end, we will create an instance of the custom token contract using the ethers library, we will use it for approving transactions made by the NFT marketplace, and to enable buy/sell operations on the frontend (need more design).

3. How will the user interact with the NFT marketplace?
    - Users are able to specify the price of their tokens and cancel their listings. Using NFT marketplace contract instance to trigger "updatePrice" function in the contract to change/specify the price.
    - Technically speaking, a user can interact with the UI and mint a token, using ethers library to interact with the contract, the frontend will call upon the contract and assign the minted token to the user address which will be his own. The user needs to provide a picture, title, price and description to mint an NFT along with a minting fee of "0.01" TOKEN.
    - Again, from the front end the user will be able to change the price of the token, which will interact with the contract and change the price of the token using the ethers library.
    - The tokens the user has are his own and to use them in the UI we will use the ethers library to interact with the contract and receive these NFTs related to a specific user account. In the UI, the user has the ability to display them or cancel them. As they are stored in a "My NFTs" tab. Once they are displayed in the UI, they can be bought by other accounts on the site. The NFTs data are stored in the contract, upon starting the application, we will call the getNFTData to retrieve all NFTs and their related data, to display them on the UI.
    - The marketplace displays the listings and allows other users to search and purchase the listed custom tokens. The Marketplace will interact with the contract and retrieve all minted NFTs and display them (only listed NFTs) in a section on the UI. Users can interact with that section and look for the NFTs, then they can click on the NFT where they will be able to purchase the NFT.
    - Upon a successful purchase, the ownership of the custom token will be transferred to the buyer. The marketplace will call on the contract buyToken function with specific arguments like to (buyer's address) and price (buying price). To invoke a transaction in MetaMask, the user will accept or reject the transaction on MetaMask, on accepting the NFT ownership will be transferred to the buyer.
    - The marketplace processes the transactions using MetaMask securely. Using ethers library. Because the verification and validation are done by MetaMask, and any vulnerability in the front end itself does not affect the security as there's no functionality affecting the user wallet without MetaMask's own security.
4. What happens when a user adds a product to the cart?
    - Allow users to view and manage the items in their cart in the UI.
        - Any backend call involved in this? Technically No, since the NFTs would be fetched already and displayed on the screen, the user will just view the fetched NFTs data.
    - Users can purchase the item in the cart, invoking the buy token function from the contract and such transaction will be processed by MetaMask.
5. What happens when a user buys an item?

- The ownership of that item (token) will be transferred to him using the _transfer function in the contract provided by ERC-721 contracts, and he will be able to list it again and specify the price.
- This ownership transfer functionality is included in the buy token function inside the contract.

Following questions are related to Custom currency tokens

6. Given a scenario, I, a user has 1000 cts coins, out of which he/she decides to sell 100 and it is listed in an exchange (not our nft market, a proper exchange)
    a. Can the coins be listed for sale in multiple markets at the same time?
        - Yes, it's possible for a user to list their tokens (coins) for sale on multiple platforms simultaneously. Since the user's ownership of the tokens is managed through their wallet and your ERC-20 token smart contract, they can interact with various platforms independently to list their tokens for sale.

    b. If 100 coins are listed in a marketplace, will that be locked – means user cannot transfer them at the same time. - This happens in normal share trading, is it similar here
        - Listing tokens for sale on a marketplace generally doesn't lock the tokens. Users can list their tokens for sale while still retaining ownership and the ability to transfer them. Token transfers are controlled by the ERC-20 functions like transfer, transferFrom, and approve.
            - Who invokes this transfer function?
                - In this case, the exchange contract invokes the custom contract's built-in function "transferFrom", to transfer funds from user A to another user B.
            - How is the permission to invoke this function controlled?
                - The exchange cannot transfer funds without getting user A's approval to deal with the balance. using the "approve" function using the custom contract's address and user A's address. So "msg.sender' needs to be equal to user A when he approves.
        - ** When you list your tokens on multiple exchanges, it doesn't inherently lock your tokens. Your tokens remain accessible and transferable, **Unlike** traditional share trading, where listing shares on one exchange can restrict their trading on another exchange, tokens on blockchains are more flexible. You can list and trade tokens across multiple exchanges simultaneously without locking them to a specific exchange. **
        - If you only have 100 tokens and you list 100 tokens on both Exchange A and Exchange B simultaneously, you are essentially committing to sell a total of 200 tokens across both exchanges. This situation could lead to potential overcommitment of tokens. But it has solutions, each transaction happens independently, assuming people buy these tokens at the same time, the first one to execute transfers the tokens and the other one fails due to insufficient balance, but this depends on the mechanisms the exchange uses.
        - In CEXs, they have centralized control over their order books and trading infrastructure. They can manage situations of overselling by preventing users from

placing orders that exceed the available token supply. This is typically done through the exchange's order matching and execution mechanisms. So they can avoid this problem altogether and much more easily than the DEXs.

- But in DEXs, Overselling/Overcommitment can sometimes happen on DEXs due to the decentralized nature and the absence of a centralized authority. Since the DEX interacts directly with the ERC-20 smart contract on the blockchain and not in a centralized wallet. But there are mechanisms like real-time supply checking to mitigate this problem.

c. How listing coins for sale in a marketplace happens. What are the technicalities involved in it

- Listing coins for sale on a marketplace involves interacting with the marketplace's (exchange's) smart contract. Users will approve the marketplace contract to spend a certain amount of their tokens (using the approve function) and then call a function in the marketplace contract to list the tokens for sale. The marketplace contract holds the information about tokens listed for sale, their prices, and other relevant details.
- Users typically approve token transfers ("allowances") in advance, most of the exchanges invoke an approve transaction for the user to approve a certain allowance the exchange is allowed to use, to be able to list, if the user doesn't approve anything or just a little, he can't list above his allowance so he has to approve again when he tries to list, but most exchanges usually let users approve in advance so that they don't have to approve every transaction . before initiating specific transactions that involve the transfer of their tokens. This approval process is a security measure and helps users control how their tokens are used by third-party contracts, like marketplaces
- When users interact with a contract that needs to move their tokens (e.g., a marketplace), they first approve that contract to spend a certain amount of their tokens. This approval process is typically done using the approve function of the token's smart contract (It's an ERC-20 functions). Users specify the recipient (contract address) and the amount of tokens they approve to be spent.
- After approving an allowance, the user can interact with the marketplace contract (or any other contract they've approved) multiple times without needing to approve each individual transaction. As long as the amount being transferred remains within the **approved allowance**, the contract can initiate token transfers on behalf of the user.
- When the user initiates a transaction on the marketplace, the marketplace contract can utilize the approved allowance to transfer tokens directly from the user's address to another user's address (e.g., the buyer's address). The marketplace contract acts as an intermediary and follows the rules set by the approval.
- If the user wants to increase or decrease the approved allowance, they can do so at any time using the approve function. This allows them to grant additional access or revoke access as needed. By using the allowance mechanism, users maintain control over how their tokens are used by third-party contracts without needing to approve

every single transaction. This improves the user experience and reduces the need for frequent approvals.

d. Where is the information of 100 coins marked for sale stored in? If in coin smart contract, where in contract?
    - The information about coins marked for sale is stored within the marketplace's smart contract, not the ERC-20 token's smart contract. The marketplace/exchange contract maintains lists of tokens listed for sale, their prices, and the respective sellers. This allows users to interact with the marketplace and initiate trades.

e. As the coins are owned by the user, how can the market initiate the sale?
    - The marketplace initiates the sale by interacting with the user's **approved allowance**. When a buyer wishes to purchase tokens from the marketplace, they call a function in the marketplace contract. The marketplace contract uses the approved allowance from the user to transfer the tokens from the user's address to the buyer's address. The marketplace acts as an intermediary for facilitating trade.

7. What is the difference between a Centralized Exchange (CEX) and a Decentralized Exchange (DEX)?
    - Centralized Exchanges (CEXs) are platforms managed by a central entity where users trade within a controlled environment, often requiring users to deposit funds onto the exchange, complete KYC verification, and rely on the exchange's infrastructure for order matching and execution. In CEXs, users transfer funds to a centralized wallet in the CEX, in this manner the CEX has control of this wallet and can manage trades in an order book more efficiently unlike DEX, the downside is it has control over your wallet and funds, so it doesn’t differ much from a bank.
    - Decentralized Exchanges (DEXs), in contrast, utilize blockchain technology and smart contracts to enable peer-to-peer trading directly from users' wallets, offering increased user custody and privacy, as well as reduced vulnerability to hacks due to their trustless nature; however, DEXs can have slower transaction speeds and lower initial liquidity (lower trading volume due to DEXs having a lower userbase) compared to CEXs.

8. Gas fees of taker and maker?
    - In the taker and maker model, Makers are traders who place limit orders that add liquidity to the order book. They are often rewarded with lower trading fees because they contribute to the market by providing resting orders. Takers are traders who place orders that match existing orders on the order book and remove liquidity. Takers pay higher trading fees since they consume existing orders.
    - Makers provide liquidity whilst takers remove liquidity. Providing liquidity means adding new orders to the order book of an exchange. Removing liquidity means executing trades that consume or match with existing orders in the order book.
    - Liquidity refers to the ease with which an asset, such as a cryptocurrency or token, can be bought or sold in the market without causing significant price fluctuations

9. Who is going to pay the fee of the failing transaction by the overcommitment problem?
    - Users are responsible for paying gas fees for transactions on a DEX, including unsuccessful ones. If a transaction fails due to overselling or any other reason, the user would still incur the gas fees associated with that transaction.

        - So, when I list my tokens on both exchanges and over-commit, I only pay the 'listing gas' fees on both exchanges. But anyone who buys my tokens and fills my order will pay the gas fees related to that transaction. Even if an overselling error occurs, he still pays gas fees until that revert happens. NOTE: in Centralized exchanges, if a glitch or error happens and overcommitment happens, CEX is usually concerned with compensating users. But in DEXs it's totally up to the user.

10. What is better CEX or DEX?

- **Decentralization:** If your project's ethos aligns with decentralization and you want to maintain full control over your token, a DEX may be a better choice. DEXs operate without intermediaries and are built on blockchain technology, allowing users to trade directly from their wallets.
- **Ease of Use:** CEXs generally offer a more user-friendly experience for traders, making it easier for users to buy and sell tokens. DEXs, while gaining popularity, can still have a steeper learning curve for less tech-savvy users.
- **Liquidity:** CEXs often have higher liquidity due to their centralized nature and larger user base. This could lead to higher trading volumes for your token. DEXs might have lower initial liquidity but can benefit from network effects and community-driven growth over time.
- **Listing Costs and Requirements:** CEXs often charge substantial listing fees and may have specific requirements for listing tokens. DEXs, being more open and decentralized, might have lower listing fees and fewer requirements.
- **Control:** Listing on a DEX gives you more control over your token's listing and management. CEXs may have stricter rules and regulations, and you may have less control over the listing process.
- **Security and Trust:** CEXs typically have centralized security measures and customer support, which can provide a sense of trust to users. DEXs rely on smart contracts and decentralized protocols for security.

11. TOKENOMICS

In summary, if you prioritize decentralization, control, and alignment with the blockchain ethos, a DEX might be a better fit. If you're looking for higher liquidity, ease of use, and potentially a broader user base, a CEX could be more suitable.

High-level workflow of custom currency token

1. **Flow 1 : Listing of token in an exchange?**
    - Order Placement Initialization:
        - You access the exchange's platform using your account credentials, which typically include your username and password.
    - Selecting Token Pair:
        - Within the exchange, you choose the trading pair for your custom token, such as your token/MATIC. A trading pair consists of two tokens that can be

exchanged against each other, such as "your token/MATIC." In simple terms, it indicates the tokens you want to trade against each other.

- Order Creation:
  - You initiate the process of placing an order to list your custom token on the chosen trading pair.
- Specifying Order Details:
  - You specify the details of the order, including the type (e.g., market, limit), the quantity of tokens, and the desired price per token.
- Transaction Confirmation:
  - You confirm the order placement through the exchange's platform, either using a UI or an API.
- Gas Fees Payment:
  - You pay any relevant trading fees or gas fees required to place the order on the exchange.
- Order Submission:
  - The exchange processes your order submission and places it in the order book for the chosen trading pair.
- Order Execution:
  - If the order matches with a counterparty's order, a trade is executed, and the tokens are transferred accordingly.
- Order Listing Complete:
  - Your custom token is now officially listed on the exchange and available for trading against the chosen trading pair. This is the last step in the listing process. Once your order is executed and the tokens are transferred, your custom token is officially listed on the exchange. It is now available for trading against the chosen trading pair.

2. **Flow 2: Buying tokens in an exchange?**

- User Interaction:
  - Users access your NFT marketplace's ReactJS-based UI. The user refers to the buyer or the person who accessed the NFT marketplace website/app.
- Connecting to Metamask:
  - Users connect their Metamask wallet to the marketplace.
- Selecting Token Purchase:
  - Users navigate to the section of the marketplace where they can purchase custom tokens.
- Token Balance Check:
  - The UI queries ethers.js to check the user's balance of custom tokens.
- Viewing Token Price:
  - The UI displays the current price of your custom token in MATIC, or another chosen base currency.

- Placing Token Order:
  - Users input the quantity of custom tokens they want to buy and confirm the order.

- Transaction Initialization:
  - The UI initiates a transaction using ethers.js, requesting approval from Metamask to interact with the custom token contract.
- Approval:
  - Metamask prompts the user to approve the custom token contract to spend tokens on their behalf.
- Contract Interaction - Approval:
- Ethers.js interacts with the custom token contract to gain approval for spending the required amount of tokens.
- Transaction Confirmation:
  - Metamask confirms the approval transaction, deducting the gas fee from the user's wallet.
- Token Purchase Transaction:
  - The UI initiates another transaction for the actual token purchase using ethers.js and Metamask.
- Contract Interaction - Token Purchase:
  - Ethers.js interacts with the custom token contract to transfer the purchased amount of tokens to the marketplace's contract address.
- Transaction Confirmation:
  - Metamask confirms the token purchase transaction, deducting the gas fee from the user's wallet.
- Confirmation and Feedback:
  - The UI displays a confirmation message, indicating that the token purchase was successful.
- Custom Token Balance Update:
  - The UI updates the user's custom token balance using ethers.js to reflect the tokens they purchased.

3. **Flow 3: Reverting(cancel) the listing of tokens from an exchange?**
   - Exchange Interaction:
     - You initiate the process to remove the listing of your custom token from the DEX.
   - Listing Removal Request:
     - Within the DEX interface, you trigger a transaction to interact with the DEX's smart contract for listing removal.
   - Transaction Initialization:
     - You specify the details of the listing removal, such as the token pair and other relevant information.
   - Gas Fees Calculation:
     - The DEX calculates the gas fees required for processing the listing removal transaction based on network conditions and complexity.
   - Gas Fees Payment:
     - You confirm the transaction and pay the calculated gas fees in the native cryptocurrency (e.g., ETH) of the blockchain network.
   - Transaction Execution:

        - Miners or validators on the blockchain network process the transaction and execute the listing removal request.
    - Confirmation and Feedback:
        - Once the transaction is confirmed, the DEX updates its order book and trading pairs to reflect the removal of your custom token's listing.
    - Listing Removal Complete:
        - The listing for your custom token is successfully removed from the DEX.

Testing

1. How to do the unit testing
    - Use Chai and Hardhat to write and execute unit tests for the Solidity smart contracts.
2. How to do the automation testing
    - (Not decided yet)
3. How to do the manual testing
    - Manually test the user interface and user flows to ensure a smooth user experience and for detecting unwanted behaviors and bugs.
4. Any set up instructions for testing
    - Installing hardhat will set up Chai as it's included inside of it and running 'npx hardhat test' will test the solidity code.

Questions

1. What is meant by mint token?
    - It means creating a new token on the blockchain.
2. Please elaborate. How it is different from other tokens e.g., NFT
    - NFT uses an ERC-721 contract which is designed for non-fungible tokens, contract but since we need to create a fungible token like a cryptocurrency, we use an ERC-20 contract.
3. Can the number of tokens be mentioned for minting?
    - On deploying the contract, the initial supply of tokens is zero. In our contract, we pass an initial number of tokens we want to mint. Then we use the mint function in the constructor to mint this passed number. Increasing our total supply to this number.
4. Can the price per token can be defined while minting, I.e., the initial price
    - Minting just takes care of creating new tokens, increasing the supply, and assigning to the address which we are minting for, it doesn't define any price. The price is defined in a liquidity pool when we add liquidity to the token using an exchange.
5. Is it required to have an exchange to put a price on our tokens? In our case we plan to sell it through our marketplace. There may not be a listing in exchange initially.

- We can add a static price for our token, I.e., we could say that the price of 1 CCS token is equal to 1 MATIC token. Temporarily at least.

6. What if we want to sell it in our marketplace, where will the price be defined? - related to another question
    - We have two methods, we either use the exchange API in order to retrieve the market price of the coin, or we can set a fixed price by implementing a smart contract function using our own logic to set the pricing, etc.
    - When we list our custom token on a centralized or decentralized exchange, the price is determined by the market dynamics and trading activities.
    - The custom token's smart contract can support different functions with different prices. When users interact with your smart contract, we can implement specific logic in the contract to handle the pricing for those transactions separately from the exchange. So, we could implement methods for users to buy the COIN on the website for a different price than the exchange
7. What is a liquidity pool and why is it necessary?
    - In the context of decentralized finance (DeFi) and decentralized exchanges (DEXs), a liquidity pool refers to a smart contract that holds funds from users who want to provide liquidity to the exchange. They are used to facilitate trading between tokens. When you add your custom token to a liquidity pool, it means you are providing a certain amount of your custom token and an equivalent value of another token (usually a stablecoin or another popular ERC-20 token like MATIC) to the pool. This action enables users to trade your custom token against the other token in the pool.
    - The liquidity pool smart contract stores information such as Reserve Balances (the quantities of both tokens held in the pool.), Token Addresses (The addresses of the tokens involved in the trading pair), Liquidity Provider Information (People who contributed to the pool, the information includes their addresses, the amount of liquidity they provided, and their share of the pool.) and other functions and data related to enable trading.
8. In the case of marketplace will the price be defined when we allocate a liquidity pool?
    - Refer to question 6.
9. Anything special to be done to add (list) a token in an exchange?
    - the contract should be deployed to the blockchain.
    - We should find an exchange to list the token. The exchange should support our network.
    - Add the token to a liquidity pool to give it a price on the exchange.
    - The exchange will list our token if we meet its listing requirements (different exchanges different requirements), the requirements typically involve submitting an application and providing information about the token, the project, and its legitimacy.
        - The above-mentioned are mostly legal steps, any technical specification to be met?
10. Is liquidity pool the initial set of tokens minted?
    - It's the initial set of tokens minted and the amount of another cryptocurrency (Matic, BNB, whatever) poured in. To give initial value to the token. Xtokens = YMATIC

- What if we want to sell tokens through our NFTmarketplace and the exchange at the same time? Once we list our token to an exchange anyone will be able to buy it through that exchange, then we can use an API tool to get data from the exchange to sell it to the user through our site.

11. The price in the NFT marketplace may be in an actual currency like $, but the price in exchange will be volatile (in the morning it may be 2 Matic, evening it may be 2.5 Matic, etc.). How will we manage these differences?
    - **Dynamic Pricing**: Implement a dynamic pricing mechanism on the NFT marketplace that calculates the price of NFTs based on the real-time exchange rate of the custom token against a stablecoin or fiat currency. This way, buyers pay an amount equivalent to the current market value of the custom token at the time of the transaction.
    - **Integration with DEX**: If the NFT marketplace and the exchange are decentralized, explore the possibility of integrating the marketplace directly with the exchange's liquidity pool. This integration could enable users to make purchases on the NFT marketplace using the current exchange rate.
    - **Automatic Price Adjustment**: Implement a mechanism that periodically adjusts the price of NFTs on the marketplace based on the token's price on the exchange. This way, the NFT prices can be kept in sync with the current market value of the custom token.
    - **User Notifications**: Inform users on the NFT marketplace about the potential price fluctuations of the custom token on the exchange. Provide real-time notifications or warnings that the actual token price may vary due to market volatility and encourage users to check the current token price before making a purchase.
    - **Transaction Confirmation Time**: Implement a short expiration period for transactions on the NFT marketplace. Since token prices can change rapidly, allowing users a limited timeframe to complete the transaction can reduce the risk of price discrepancies during the payment process.
12. Does it mean that when the token is listed in marketplace, then the only way to sell it is though the marketplace? Can't we directly sell it to a user without calling any exchange API because it's our token. We want full control of it
    - refer to Question 6.
    - When you list your token on an exchange, you typically do not give any direct privileges to the exchange over your smart contract. Instead, the exchange integrates your token into its own trading platform, and users can trade your token on the exchange's platform without the exchange having direct control over your token's smart contract. So, in other words, we are in control of any action in the contract, listing it on an exchange just allows users to trade our coin.
13. How is the price calculated in the standard marketplace/exchange (not ours)? How this will impact the price of our tokens
    - Initially, it depends on the amount of the Stablecoin or the ERC-20 coin we provide (MATIC, BNB, etc.), when we add our coin to a liquidity pool, then it depends on the trade dynamics.

Zaid's questions:

1. I can’t find any exchange to add our token to a liquidity pool since we are working on Mumbai testnet. There was quickswap but it’s temporarily not working for Mumbai testnet. I am currently researching more about it, but I am kind of stuck on that part.

Ans - At this point using a test net, just make sure that our tokens support the necessary requirements to be listed in an exchange. If you don’t find a test exchange that’s fine.

2. Will we use a CEX or a DEX?

Please add your findings in this document