1. **Introduction**

In this project, an NFT marketplace is built for users to buy and sell NFTs. This project can be separated into three parts: smart contracts, querying and the front end.

In the application website, the user needs to connect with their wallets first before any action. Before connecting their wallet, the page will only notify the user to connect.

After connecting, in the index page, they will see the NFT listings. Here they can see all the NFTs on sale with their price listed. They can click on the NFT which will direct them to the individual NFT listing page.

In the individual listing page, users can buy the listed NFT with the price of the listed price plus the gas fee. If the user is the owner of the NFT, they can update the sale price which costs them gas fee as well.

1. **Development**

The following are the implementation code and brief introduction to them. Please note that even though the frontend occupies the majority of the source code, most of the time is spent into both Hardhat and the Graph on system configuration.

* 1. Smart Contract

We need to deploy contracts related to the marketplace and the NFTs that would be sold. For simplicity, the same NFT will be minted 4 times for the owner as 4 tokens for sale. The tool in this section includes Hardhat, a development environment for Ethereum to create, compiling and deploying smart contracts. To work with NFTs in the project smart contracts, OpenZeppelin is also used as a library for creating our NFTs (ERC721), so we only need to set the IPFS file of the NFT before deploying the contract.

2 contracts are deployed using Hardhat: NFTMarket.sol as the market contract and CourseNFT.sol as our hardcoded NFT contract (it can be changed to other names or IPFS files, this is just an example).

For the NFT market contract, it contains a mapping of NFT listings. It also contains functions that handle the NFT listing, including creating, update, remove, read, and purchase listings. It also has functions for checking, including if the NFT is listed or not, and the price is valid whether the user is the owner of the NFT.

//market solidity code

The other file is the NFT contract. The ERC721 must be deployed before entering the market. Just for simplicity, 5 NFTs of the same IPFS file will be created and minted to the user that deployed that contract. The only function in the contract is to get the token URI, which returns IPFS file which is uploaded there previously.

To interact with our smart contracts in the below sections, we also need to export the contract ABI to them as a JSON file.

* 1. The Graph

The Graph is a decentralized querying protocol, primarily used for querying and indexing blockchain data. Using this protocol, we can create our own subgraph which stores our NFT data like the buyers and sellers. It would be especially helpful if we needed to query a large amount of NFT data. We will store the 4 NFT details as a subgraph, which acts like a normal database, but decentralized. We need to use GraphQL to query the data. The Graph uses TypeScript (AssemblyScript in actual terms), which will be compiled into WebAssembly when a script runs.

In this (relatively simple) project, we only need to configurate 3 files: the schema file (schema.graphql), the subgraph metadata file (subgraph.yaml), and the function handler of the subgraph (src/nft-market.ts).

The schema of the NFT is:

//schema

The configuration file is:

//yaml

The function handler of the NFT is:

//src

* 1. Frontend

Next.js is a frontend JavaScript framework, which is a meta framework of React with server-side rendering (SSR). We will import multiple libraries to support the queries and transactions. To connect the wallet with the browser, RainbowKit is used to provide the “Connect Wallet” component with its default widget display. Wagmi is needed for fetching public data on the blockchain and any interactions with smart contracts (except querying our own NFT market). We also need to connect to a public IPFS gateway to retrieve the NFT data, primarily the image, so we use the Pinata Cloud public gateway as the link.