

# **Capstone Project Report**

## **Mp.3 NFTs**

**Submitted to**

**Advisor: Dr.**

**By**

Nutthapat Lohasawaroge 6222772285

Panyakorn Kraiteerawut 6222790139

Thanakorn Pumpongam 6222790261

## INTRODUCTION

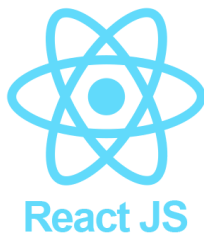
This Web3 project incorporates mp3 or music in the form of an NFT. The platforms we use to listen to music too are constantly changing. In the past, analog systems such as tapes and vinyl records were everywhere. Now everyone you know either has a Spotify or an Apple Music account. Web3 and mp3 NFTs might be the future of the industry. Artists can promote their content and earn money through our platform. In addition, only a few people have ever seen the mp3 NFTs which are NFTs that can be able to run music at the sametime. We also created the opportunity for NFT collectors, artist's fans, and investors to do what they want on our platform.

## TECHNICAL STRUCTURE

We have created a web3 application where everyone can make their transaction of mp3 NFTS in our platform by using smart contracts when users trigger the font end components. Below here, we are going to explain how we created our web3 application.

### For Frontend

We are using ReactJs to connect our frontend with backend. Our frontend is composed of jsx, css, javascript, and html files. We are using



### For Backend

We are using many things such as hardhat, IPFS, json, and solidity files.

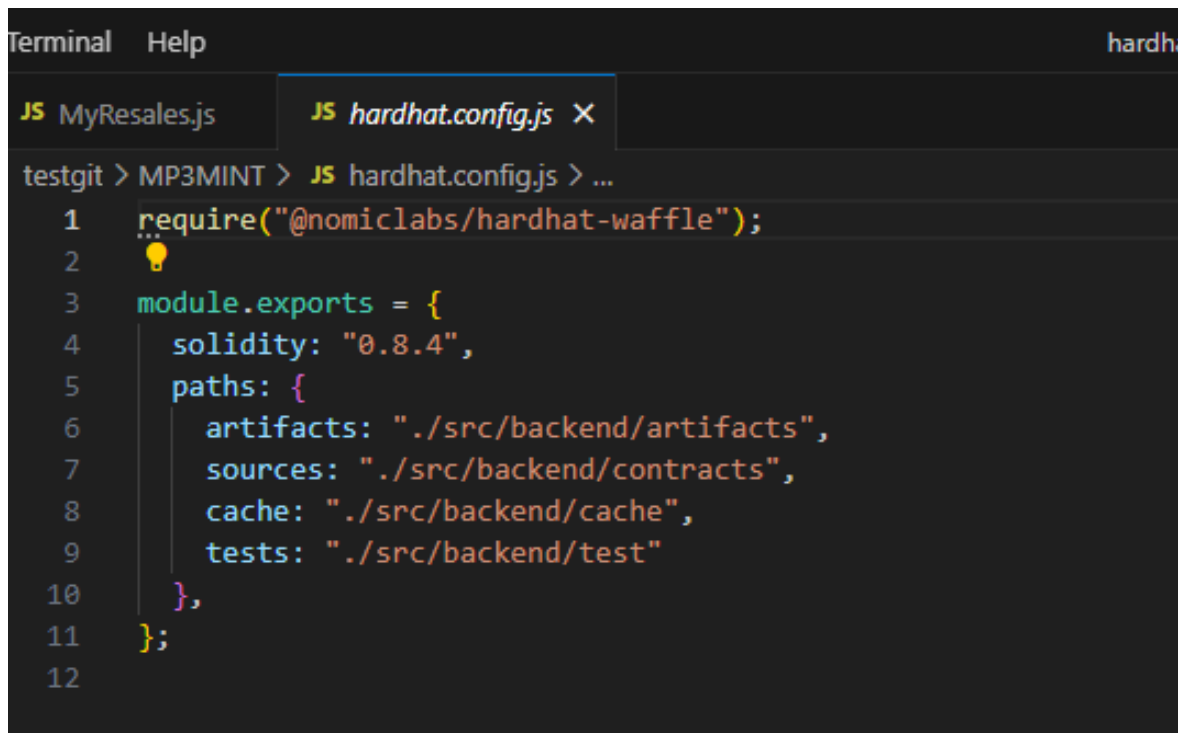
Hardhat file is used to interact with solidity to deploy a smart contract to the network.

IPFS is where our data is stored and we use it as a backend where we can retrieve the data by typing command.

Json file is where to keep the method and script including data information in arrays.

Solidity is simply called a smart contract file that defines how our contract should work. For example, we create the function buytoken() for the smart contract to create the sale of a music NFT listed on the marketplace, transfer the ownership, and make it interactive by connecting with the frontend by using javascript.

## Source code overview:



The screenshot shows a code editor with a dark theme. At the top, there are tabs for 'Terminal' and 'Help'. Below the tabs, there are two file tabs: 'JS MyResales.js' and 'JS hardhat.config.js' with a close button. The main editor area shows the content of 'hardhat.config.js'. The code is as follows:

```
testgit > MP3MINT > JS hardhat.config.js > ...
1  require("@nomiclabs/hardhat-waffle");
2  ⚡
3  module.exports = {
4    solidity: "0.8.4",
5    paths: {
6      artifacts: "./src/backend/artifacts",
7      sources: "./src/backend/contracts",
8      cache: "./src/backend/cache",
9      tests: "./src/backend/test"
10   },
11 };
12
```

### hardhat.config.js

- Set default module every single use of hardhat config by command **require('@nomiclabs/hardhat-waffle')**
- **module.exports** to specific setting path and directory of any source code needed to use in this project

# deploy.js

```
Terminal  Help  deployjs - Untitled (Workspace) - Visual Studio Code

JS deployjs x
testgit > MP3MINT > src > backend > scripts > JS deployjs > main
1  async function main() {
2    const toWei = (num) => ethers.utils.parseEther(num.toString())
3    let royaltyFee = toWei(0.01);
4    let prices = [toWei(1), toWei(2), toWei(3), toWei(4), toWei(5), toWei(6), toWei(7), toWei(8)]
5    let deploymentFees = toWei(prices.length * 0.01)
6    const [deployer, artist] = await ethers.getSigners();
7
8    console.log("Deploying contracts with the account:", deployer.address);
9    console.log("Account balance:", (await deployer.getBalance()).toString());
10
11    // deploy contracts here:
12    const NFTMarketplaceFactory = await ethers.getContractFactory("MusicNFTMarketplace");
13    nftMarketplace = await NFTMarketplaceFactory.deploy(
14      royaltyFee,
15      artist.address,
16      prices,
17      { value: deploymentFees }
18    );
19
20    console.log("Smart contract address:", nftMarketplace.address)
21
22    // For each contract, pass the deployed contract and name to this function to save a copy of the contract ABI and address to the front end.
23    saveFrontendFiles(nftMarketplace, "MusicNFTMarketplace");
24  }
25
```

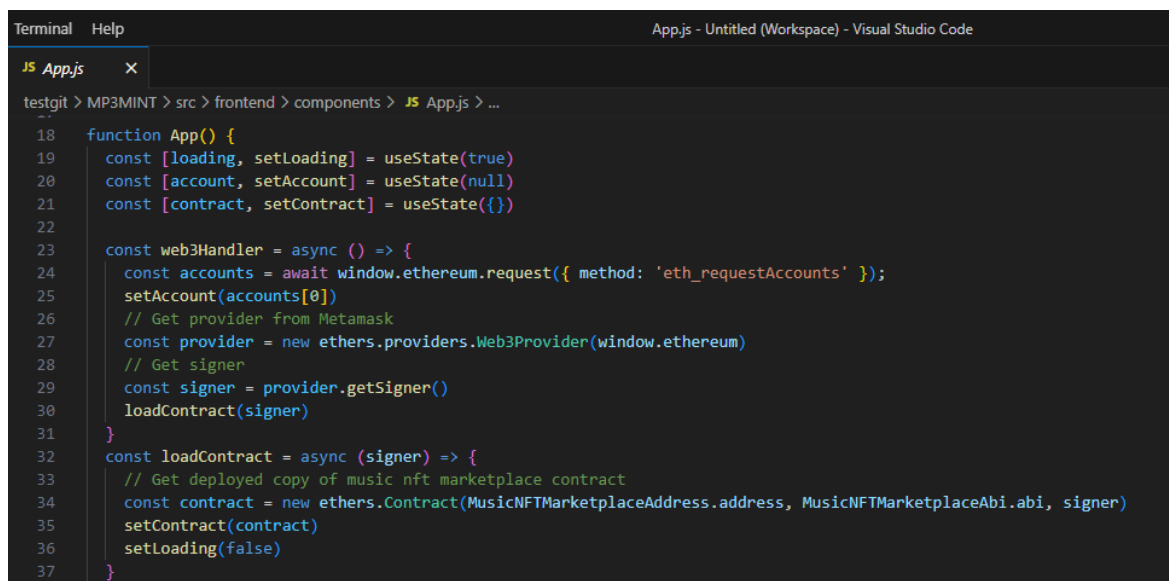
```
Terminal  Help  deployjs - Untitled (Workspace) - Visual Studio Code

JS deployjs x
testgit > MP3MINT > src > backend > scripts > JS deployjs > main
25
26  function saveFrontendFiles(contract, name) {
27    const fs = require("fs");
28    const contractsDir = __dirname + "/../../frontend/contractsData";
29
30    if (!fs.existsSync(contractsDir)) {
31      fs.mkdirSync(contractsDir);
32    }
33
34    fs.writeFileSync(
35      contractsDir + `/${name}-address.json`,
36      JSON.stringify({ address: contract.address }, undefined, 2)
37    );
38
39    const contractArtifact = artifacts.readArtifactSync(name);
40
41    fs.writeFileSync(
42      contractsDir + `/${name}.json`,
43      JSON.stringify(contractArtifact, null, 2)
44    );
45  }
46
47  main()
48    .then(() => process.exit(0))
49    .catch(error => {
50      console.error(error);
51      process.exit(1);
52    });
53
```

## deploy.js

- Declare a variable for importing the data in a smart contract by ethers dependency and returns console log to check the validation of data
- Declare **saveFrontendFiles(contract, name)** to write the received data and parse into json

## App.js



```
Terminal  Help
App.js - Untitled (Workspace) - Visual Studio Code

JS App.js x
testgit > MP3MINT > src > frontend > components > JS App.js > ...
18  function App() {
19    const [loading, setLoading] = useState(true)
20    const [account, setAccount] = useState(null)
21    const [contract, setContract] = useState({})
22
23    const web3Handler = async () => {
24      const accounts = await window.ethereum.request({ method: 'eth_requestAccounts' });
25      setAccount(accounts[0])
26      // Get provider from Metamask
27      const provider = new ethers.providers.Web3Provider(window.ethereum)
28      // Get signer
29      const signer = provider.getSigner()
30      loadContract(signer)
31    }
32    const loadContract = async (signer) => {
33      // Get deployed copy of music nft marketplace contract
34      const contract = new ethers.Contract(MusicNFTMarketplaceAddress.address, MusicNFTMarketplaceAbi.abi, signer)
35      setContract(contract)
36      setLoading(false)
37    }
}
```

```

return (
  <BrowserRouter>
    <div className="App">
      <>
        <Navbar expand="lg" style={{backgroundColor: '#ffffff'}}>
          <Container style={{backgroundColor: '#ffffff'}}>
            <Navbar.Brand
              href="/"
              style={{
                fontWeight: '500',
                color: 'black'}}
            >
              <img src={logo} width="100" height="100" className="" alt="" />
              {/ * &nbsp; MP3 MINT */}
            </Navbar.Brand>
            <Navbar.Toggle aria-controls="responsive-navbar-nav" />
            <Navbar.Collapse id="responsive-navbar-nav">
              <Nav className="me-auto">
                <Nav.Link as={Link} to="/" style={{color: 'black'}}>Home</Nav.Link>
                <Nav.Link as={Link} to="/my-tokens" style={{color: 'black'}}>My Tokens</Nav.Link>
                <Nav.Link as={Link} to="/my-resales" style={{color: 'black'}}>My Resales</Nav.Link>
              </Nav>
            </Navbar.Collapse>
          </Container>
        </Navbar>
      </>
    </div>
  </BrowserRouter>
)

```

Terminal Help App.js - Untitled (Workspace) - Visual

JS App.js X

testgit > MP3MINT > src > frontend > components > JS App.js > ...

```

60     <Nav>
61       {account ? (
62         <Nav.Link
63           href={`https://etherscan.io/address/${account}`}
64           target="_blank"
65           rel="noopener noreferrer"
66           className="button nav-button btn-sm mx-4">
67         <Button
68           variant="outline-light"
69           style={{
70             borderColor: 'black',
71             color: 'black'}}
72         >
73           {account.slice(0, 5) + '...' + account.slice(38, 42)}
74         </Button>
75       </Nav.Link>
76     ) : (
77       <Button
78         onClick={web3Handler}
79         variant="outline-light"
80         style={{
81           borderColor: 'black',
82           color: 'black'}}
83       >Connect Wallet</Button>
84     )}
85   </Nav>
86 </Navbar.Collapse>
87 </Navbar>

```

```
Terminal  Help
App.js - Untitled (Workspace) - Visual Studio Code

JS App.js x
testgit > MP3MINT > src > frontend > components > JS App.js > ...
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117

    </Navbar>
  </>
  <div>
    {loading ? (
      <div style={{ display: 'flex', justifyContent: 'center', alignItems: 'center', minHeight: '80vh' }}>
        <Spinner animation="border" style={{ display: 'flex' }} />
        <p className='mx-3 my-0'>Awaiting Metamask Connection...</p>
      </div>
    ) : (
      <Routes>
        <Route path="/" element={
          <Home contract={contract} />
        } />
        <Route path="/my-tokens" element={
          <MyTokens contract={contract} />
        } />
        <Route path="/my-resales" element={
          <MyResales contract={contract} account={account} />
        } />
      </Routes>
    )}
  </div>
</div>
</BrowserRouter>
);
}
export default App;
```

## App.js

- After import module, declare variables to handle retrieved data by **web3Handler**(the request for sign in Metamask) and **loadContract**(retrieved solidity smart contract data and connect with the frontend)
- Setting **BrowserRouter** to create connection between page
- Using **<Navbar>** imported module from directory, styling beautiful navigator bar, and retrieved wallet data
- Condition if user already login or not, if user already login route the landing page's path with **<Home>** imported module that using props from App.js and user can connect to others path such as **<MyTokens>** and **<MyResales>**, if not the user will see only the message 'Awaiting Metamask Connection' until they're connect their Metamask wallet



# Solidity

```
Terminal Help MusicNFTMarketplace.sol - Untitled (Workspace) - Visual Studio Code

MusicNFTMarketplace.sol X
testgit > MP3MINT > src > backend > contracts > MusicNFTMarketplace.sol
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.4;
3
4 import "@openzeppelin/contracts/access/Ownable.sol";
5 import "@openzeppelin/contracts/token/ERC721/ERC721.sol";
6
7 contract MusicNFTMarketplace is ERC721("DAppFi", "DAPP"), Ownable {
8     string public baseURI =
9         "https://bafybeidhjjbjonyqcahuz1pt7s2nmh4xrlbspa3gstop5o47l6gsiaffee.ipfs.nftstorage.link/";
10    string public baseExtension = ".json";
11    address public artist;
12    uint256 public royaltyFee;
13
14    struct MarketItem {
15        uint256 tokenId;
16        address payable seller;
17        uint256 price;
18    }
19    MarketItem[] public marketItems;
20
21    event MarketItemBought(
22        uint256 indexed tokenId,
23        address indexed seller,
24        address buyer,
25        uint256 price
26    );
```

```
Terminal Help MusicNFTMarketplace.sol - Untitled (Workspace) - Visual Studio Code

MusicNFTMarketplace.sol X
testgit > MP3MINT > src > backend > contracts > MusicNFTMarketplace.sol
27     event MarketItemRelisted(
28         uint256 indexed tokenId,
29         address indexed seller,
30         uint256 price
31     );
32
33     /* In constructor we initialize royalty fee, artist address and prices of music nfts*/
34     constructor(
35         uint256 _royaltyFee,
36         address _artist,
37         uint256[] memory _prices
38     ) payable {
39         require(
40             _prices.length * _royaltyFee <= msg.value,
41             "Deployer must pay royalty fee for each token listed on the marketplace"
42         );
43         royaltyFee = _royaltyFee;
44         artist = _artist;
45         for (uint8 i = 0; i < _prices.length; i++) {
46             require(_prices[i] > 0, "Price must be greater than 0");
47             _mint(address(this), i);
48             marketItems.push(MarketItem(i, payable(msg.sender), _prices[i]));
49         }
50     }
51
52     /* Updates the royalty fee of the contract */
53     function updateRoyaltyFee(uint256 _royaltyFee) external onlyOwner {
54         royaltyFee = _royaltyFee;
55     }
56 }
```

```
Terminal Help MusicNFTMarketplace.sol - Untitled (Workspace)
MusicNFTMarketplace.sol X
testgit > MP3MINT > src > backend > contracts > MusicNFTMarketplace.sol
56
57     /* Creates the sale of a music nft listed on the marketplace */
58     /* Transfers ownership of the nft, as well as funds between parties */
59     function buyToken(uint256 _tokenId) external payable {
60         uint256 price = marketItems[_tokenId].price;
61         address seller = marketItems[_tokenId].seller;
62         require(
63             msg.value == price,
64             "Please send the asking price in order to complete the purchase"
65         );
66         marketItems[_tokenId].seller = payable(address(0));
67         _transfer(address(this), msg.sender, _tokenId);
68         payable(artist).transfer(royaltyFee);
69         payable(seller).transfer(msg.value);
70         emit MarketItemBought(_tokenId, seller, msg.sender, price);
71     }
72
73     /* Allows someone to resell their music nft */
74     function resellToken(uint256 _tokenId, uint256 _price) external payable {
75         require(msg.value == royaltyFee, "Must pay royalty");
76         require(_price > 0, "Price must be greater than zero");
77         marketItems[_tokenId].price = _price;
78         marketItems[_tokenId].seller = payable(msg.sender);
79
80         _transfer(msg.sender, address(this), _tokenId);
81         emit MarketItemRelisted(_tokenId, msg.sender, _price);
82     }
83
```

```
Terminal Help MusicNFTMarketplace.sol - Untitled (Workspace)
MusicNFTMarketplace.sol X
testgit > MP3MINT > src > backend > contracts > MusicNFTMarketplace.sol
85     function getAllUnsoldTokens() external view returns (MarketItem[] memory) {
86         uint256 unsoldCount = balanceOf(address(this));
87         MarketItem[] memory tokens = new MarketItem[](unsoldCount);
88         uint256 currentIndex;
89         for (uint256 i = 0; i < marketItems.length; i++) {
90             if (marketItems[i].seller != address(0)) {
91                 tokens[currentIndex] = marketItems[i];
92                 currentIndex++;
93             }
94         }
95         return (tokens);
96     }
97
98     /* Fetches all the tokens owned by the user */
99     function getMyTokens() external view returns (MarketItem[] memory) {
100         uint256 myTokenCount = balanceOf(msg.sender);
101         MarketItem[] memory tokens = new MarketItem[](myTokenCount);
102         uint256 currentIndex;
103         for (uint256 i = 0; i < marketItems.length; i++) {
104             if (ownerOf(i) == msg.sender) {
105                 tokens[currentIndex] = marketItems[i];
106                 currentIndex++;
107             }
108         }
109         return (tokens);
110     }
111
```

```

111
112     /* Internal function that gets the baseURI initialized in the constructor */
113     function _baseURI() internal view virtual override returns (string memory) {
114         return baseURI;
115     }
116 }
117

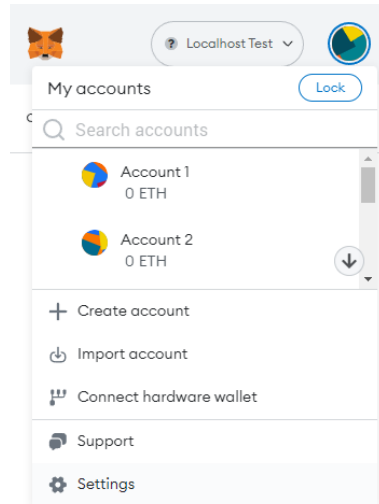
```

### MusicNFTMarketplace.sol

- setting solidity version and import module from **@openzeppelin/contracts**
- Create contract **MusicNFTMarketplace** and declare variable **baseURI** to retrieve data from IPFS, **baseExtension** to specific retrieve data which in json file, **artist** to specific artist's address, and **royaltyFee** to set the royalty fee
- Declare struct **MarketItem** to set the NFTs id/can define seller address/set the price
- Declare event **MarketItemBought** to refer tokenId, seller address, buyer address, and bought price when NFTs get bought from marketplace
- Declare event **MarketItemRelisted** to refer tokenId, seller address, and resales price when NFTs are resale from owner
- Declare construct to mint NFTs into marketplace, set royalty fee, artist address when list the retrieved data from IPFS
- Declare function **updateRoyaltyFee** to update automatically the royalty fee when NFTs are resale
- Declare function **buyToken** to let users can buy NFTs from the marketplace, automatically transfer owner and pay royalty fee to the artist
- Declare function **resellToken** to let users who owned NFTs can resale their NFTs into the marketplace, automatically pay royalty fee, show NFTs, and updated price on the marketplace
- Declare function **getAllUnsoldTokens** to automatically delisted the sold NFTs and listed unsold NFTs in the marketplace
- Declare function **getMyTokens** to automatically list NFTs into users wallet when they're bought NFTs for the marketplace
- Declare function **\_baseURI** to get **baseURI** variable initialized in the constructor

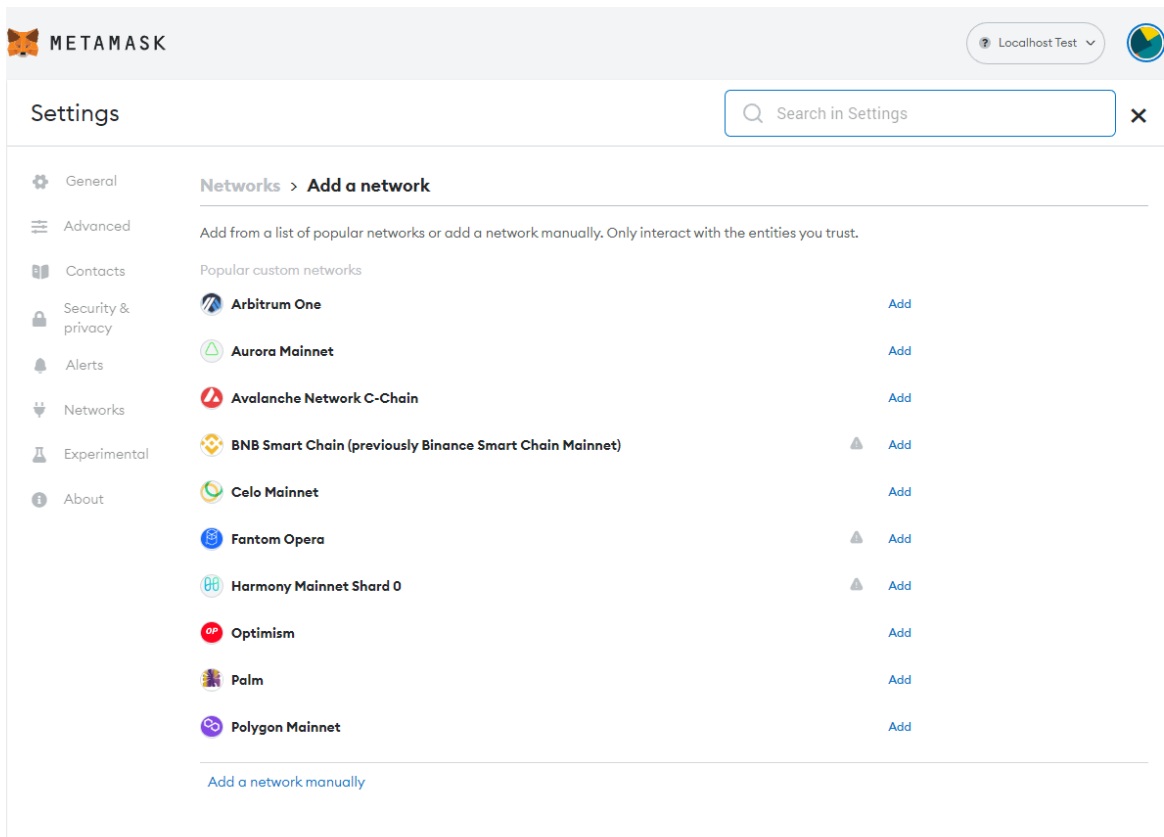
## Connect development blockchain accounts to Metamask:

- Copy the private key of the address and import it to Metamask
- Connect your Metamask to hardhat blockchain, network 127.0.0.1:8545
- If you have not added hardhat to the list of networks on your Metamask, open up a browser, click the fox icon, then click the dropdown button that lists all the available networks then click add networks.

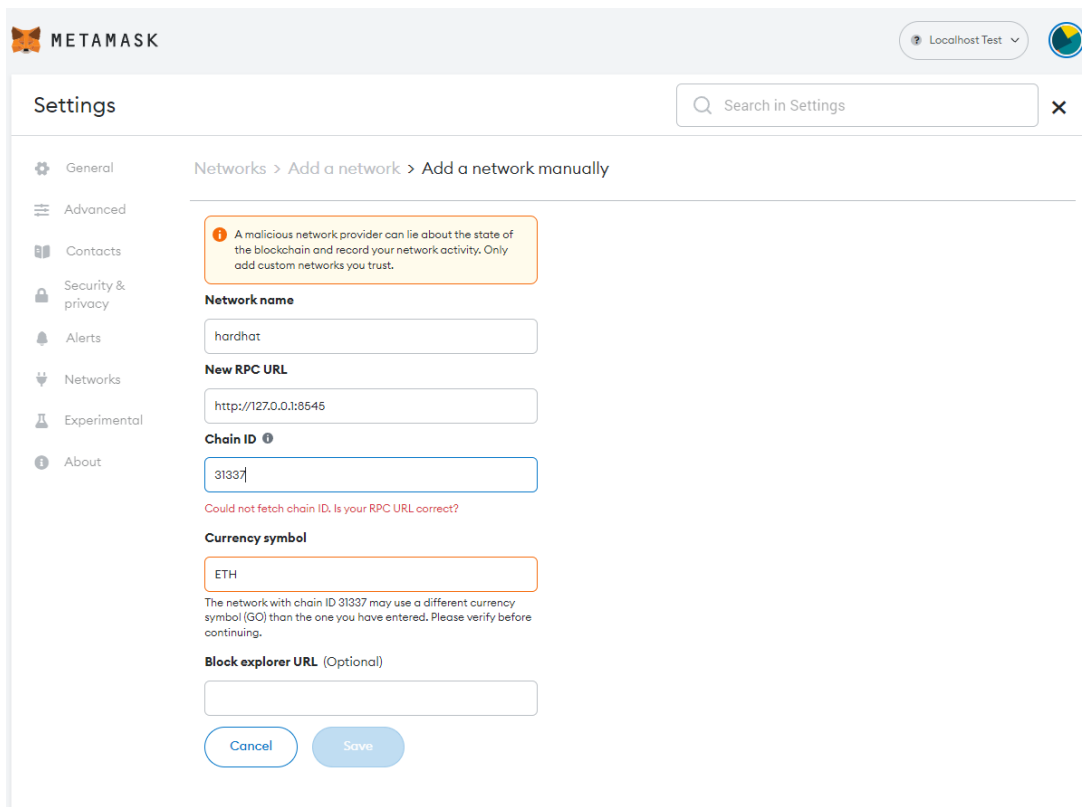


Need help? Contact [MetaMask support](#)

**click 'Settings'**



click ‘Add a network manually’



## Input Network Name, New RPC URL, Chain ID, and don't forget to input Currency symbol 'ETH'

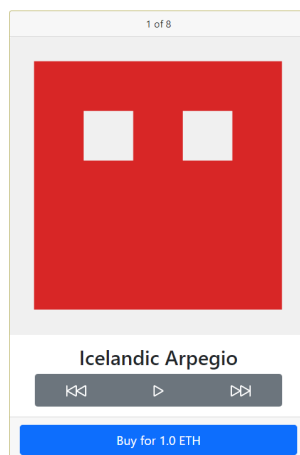
- A form should popup.
- For the 'Network Name' field enter 'hardhat'.
- For the 'New RPC URL' field the 'http://127.0.0.1:8545'.
- For the chainID enter '31337' then click save.

## Application Overview:

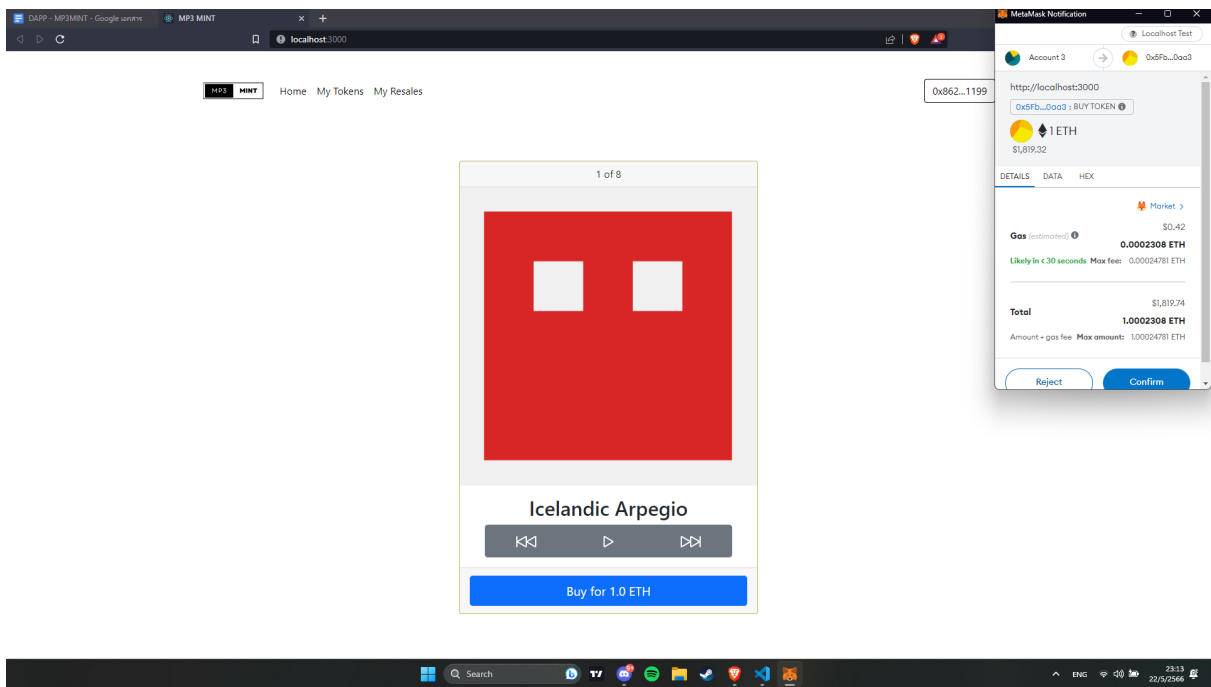


Awaiting Metamask Connection...

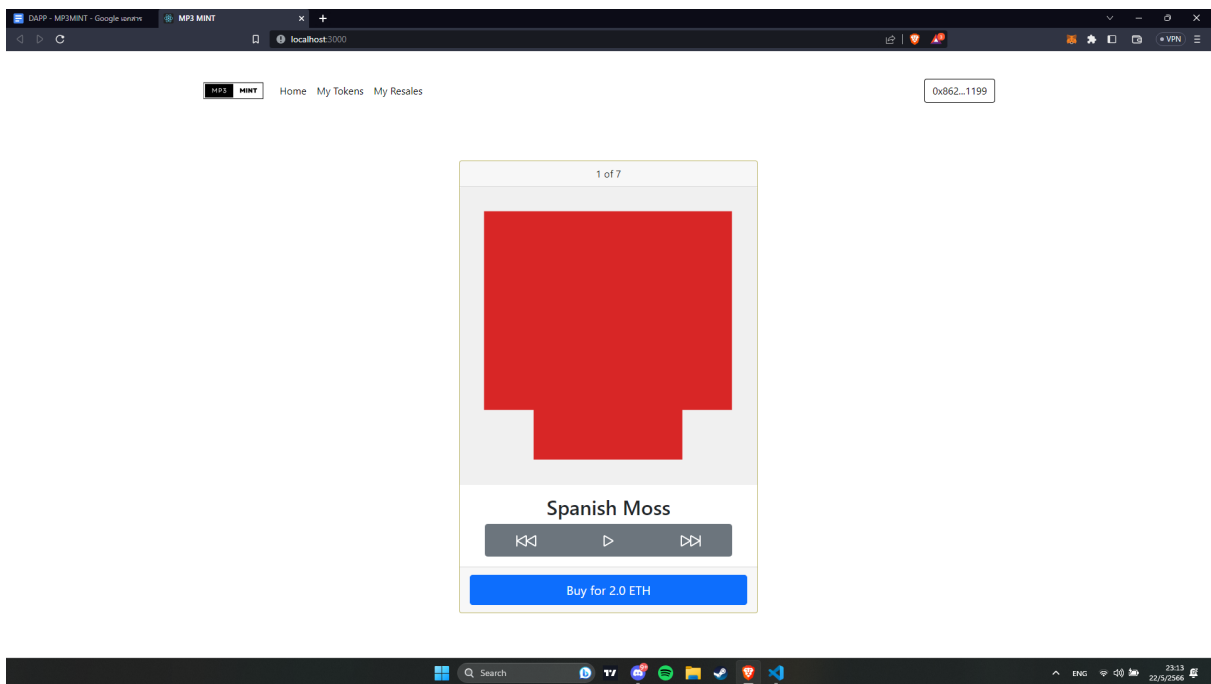
## Landing page(anonymous)



## Landing page(already login)



## Transaction




## Landing page(after buy nfts)



MP3 MINT Home My Tokens My Resales

0x862...1199



Icelandic Arpeggio

▶

1.0 ETH

[Resell](#)




## My Tokens page



MP3 MINT Home My Tokens My Resales

0x862...1199

Listed



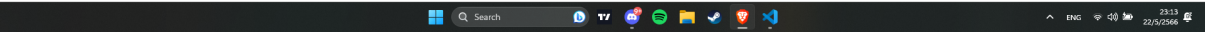
Icelandic Arpeggio

▶

12.0 ETH

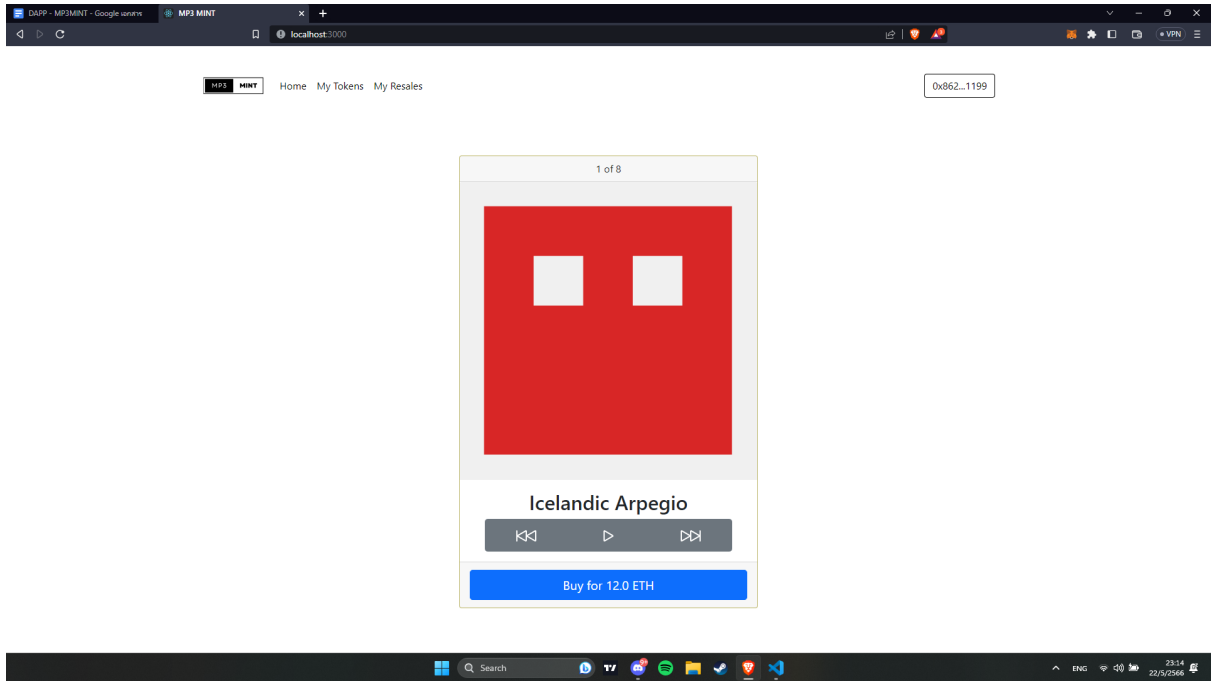
Sold

No sold assets



## My Resales page





**Landing page(after resale NFTs)**

## **Conclusion**

To conclude, this project outlines the design and implementation of our Web3 platform. NFTs that can play music are a fairly new concept that has not been seen much in the market. This is our strong point and what differentiates us from competitors. There is the potential for artists to sell the license for their music to be used in the NFTs and make additional income. Moreover, the NFTs on our platform have that special aspect of being able to play music in addition to the artwork which adds value to it. Overall, with this service couple with our passion to keep improving our service make me think that there is promising future for it.