# NFT MARKETPLACE

# PROJECT-IBC

Submitted by:

**Isha R. Mehta (9919103136)**
**Priyanshu Kumar (9919103137)**
**Bhavya Varshney (9919103142)**

Under the supervision of**:**

**Dr. Himanshu Agrawal**


**Department of CSE/IT**

**Jaypee Institute of Information Technology University, Noida**

# INTRODUCTION

NFT, known as non-fungible tokens (NFTs), these cryptographic assets are based on blockchain technology similar to cryptocurrencies and have unique identification codes and metadata that set them apart from each other.Unlike cryptocurrencies, they cannot be traded or exchanged at equivalency. NFTs are tokens that we can use to represent ownership of unique items. They let us tokenize things like art, collectibles, even real estate. They can only have one official owner at a time and no one can modify the record of ownership or copy/paste a new NFT. In the year 2022, the NFT has gone wildly outrageous; the growth they have shown is substantive market growth. From being a market worth a few million dollars, the NFT market has grown up to a market of handling billions of dollars as revenue in one single month.If you want in on the NFT craze, an NFT marketplace is your gateway to participating in the purchase and sale of these digital assets -- from art to music to entire virtual worlds.

# ABSTRACT

The following project is the replica of the Opensea NFT Marketplace which is one of the largest Marketplaces for NTF.Just like Opensea website we have tried to create a marketplace for NFT from where we can create a NFT and can purchase a NFT using cryptocurrency.We have used the local blockchain network using Hard hat which provides platform where we can deploy our smart contracts and can execute our transactions efficiently.We are doing all these transactions through metamask wallet. We have also used IPFS for storing the files and getting the hash value of that file as a transaction in the blockchain node.

# SCOPE OF THE PROJECT

In this rapid evaluation of NFTs stands for non-fungible tokens which are based on the blockchain technology.NFTs can represent real-world items like artwork and real estate.NFTs can also function to represent individuals' identities, property rights, and more.The distinct construction of each NFT has the potential for several use cases. For example, they are an ideal vehicle to digitally represent physical assets like real estate and artwork. Because they are based on blockchains, NFTs can also work to remove intermediaries and connect artists with audiences or for identity management. NFTs can remove intermediaries, simplify transactions, and create new markets.Much of the current market for NFTs is centered around collectibles, such as digital artwork, sports cards, and rarities. Perhaps the most hyped space is NBA Top Shot, a place to collect non-fungible tokenized NBA moments in digital card form. Some of these cards have sold for millions of dollars.To Manage all the NFTs we need a simple yet efficient marketplace where users can sell or buy the NFTs.our current interface include Mylisteditems,my purchases,home and wallet connected.Once the NFTs are listed they come along there design,description,name and cost.As we have already stated we tried to build replica of Opensea so future scope includes other functionality of the same marketplace with increases user interactions such as ranking different NFTs,categorizing them as per there Genre and so on.

# TOOLS AND TECHNOLOGIES

- **Solidity (Writing Smart Contract)**:-Solidity is an object-oriented programming language for implementing smart contracts on various blockchain platforms, most notably, Ethereum.

- **Javascript (React & Testing)**:-JavaScript, often abbreviated JS, is a programming language that is one of the core technologies of the World Wide Web, alongside HTML and CSS.React is a free and open-source front-end JavaScript library for building user interfaces based on UI components. It is maintained by Meta and a community of individual developers and companies.JavaScript Unit Testing is a method where JavaScript test code is written for a web page or web application module.

- **Ethers (Blockchain Interaction)**:-Ethereum is a decentralized, open-source blockchain with smart contract functionality. Ether is the native cryptocurrency of the platform. Among cryptocurrencies, Ether is second only to Bitcoin in market capitalization.

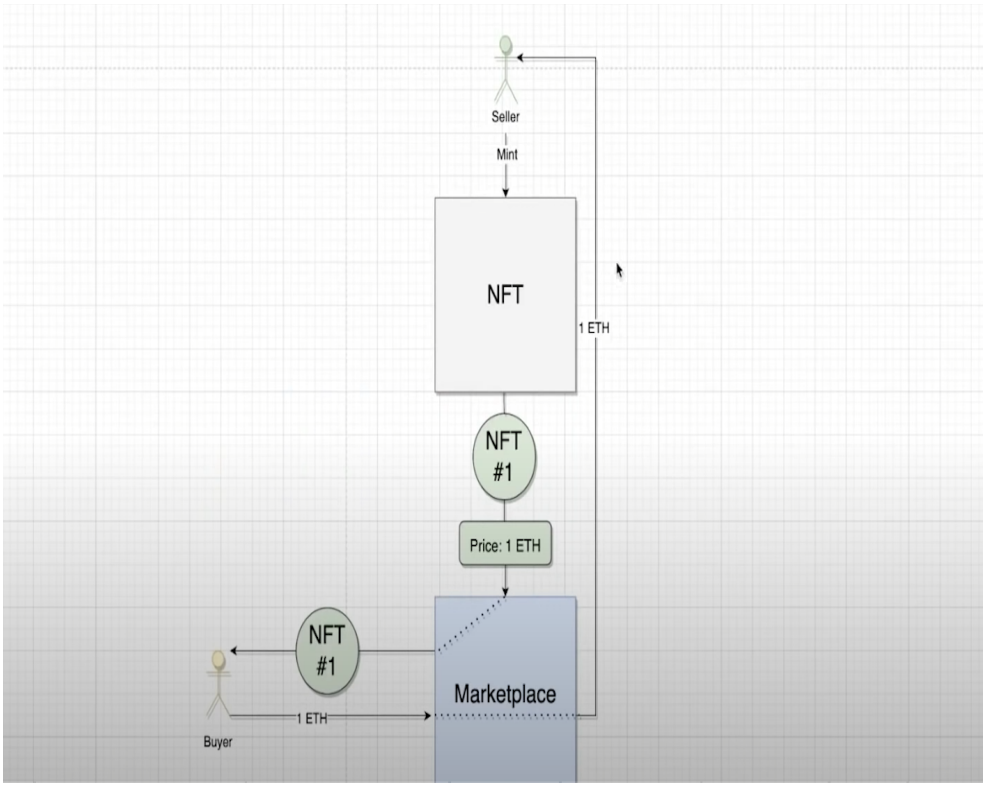
- **Hardhat (Development Framework)**:-Hardhat is a development environment to compile, deploy, test, and debug your Ethereum software. It helps developers manage and automate the recurring tasks that are inherent to the process of building smart contracts and dApps, as well as easily introducing more functionality around this workflow.

- **Ipfs (Metadata storage)**:-The InterPlanetary File System is a protocol and peer-to-peer network for storing and sharing data in a distributed file system. IPFS uses content-addressing to uniquely identify each file in a global namespace connecting all computing devices.

- **React routers (Navigational components)**:-React Router is the standard routing library for React. From the docs: “React Router keeps your UI in sync with the URL. It has a simple API with powerful features like lazy code loading, dynamic route matching, and location transition handling built right in.

# DESIGN OF THE PROJECT

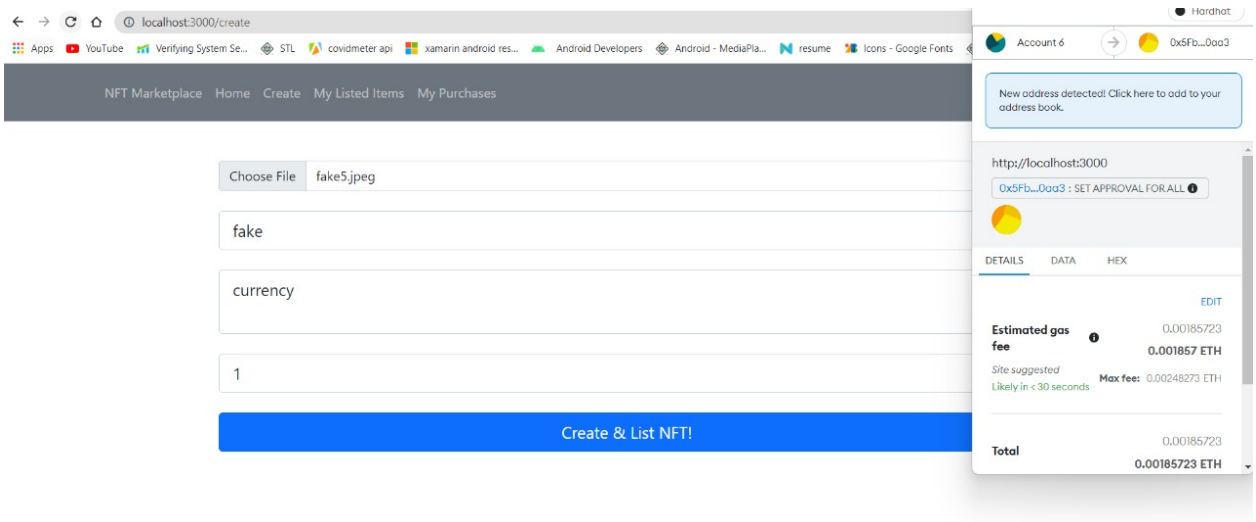
It is designed in such a way that sellers can create a NFT by uploading files which are stored on ipfs server and it returns the hash value of that file which is stored as a transaction in the node of the local blockchain which is supplied by the hardhat framework.After that buyer interacts with the market place and can buy the NFT directly through the seller.In this way this is completely a peer to peer network where one can generate his/her NFT and anyone can buy it directly through the seller.

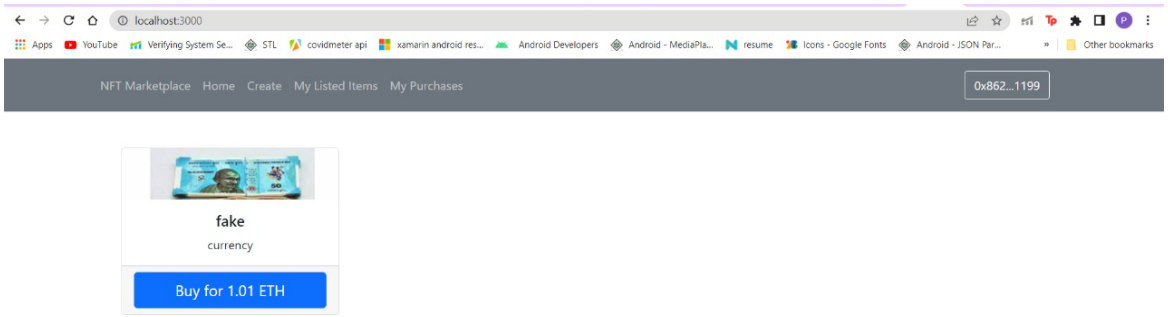


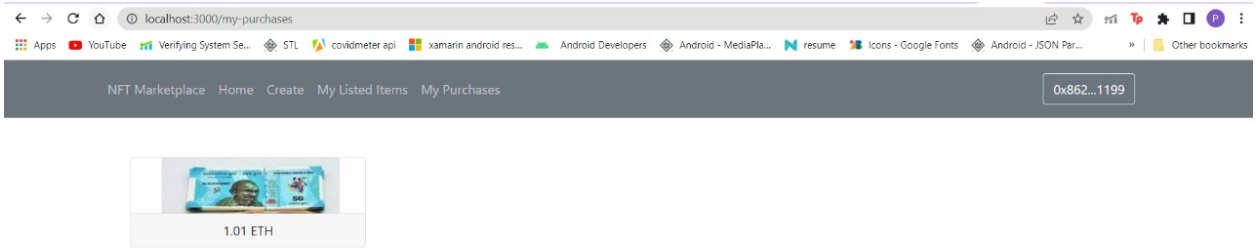
# IMPLEMENTATIONS DETAILS

Frontend is implemented using React JS in which we have divided the Webpage into four sections:-

- Home:-It displays all the NFTs which are created by a user and now available for sale .The listed NFTs have their art,name,description and price,also a button for buying that particular NFT.
- Create:-This section is used to upload digital files of which a user wants to create a NFT and he/she can also decide the value for that NFT.
- My Listed Items:- It shows all the items that are Listed by us for selling the NFT.
- My Purchases:-Is shows all the purchased NFTs by a user.
- Connect wallet:-For all the Metamask Transactions.






Backend is implemented using Solidity,Hardhet,IPFS and Matamask.Solidity is used to write the Smart Contracts named as Marketplace.sol and NFT.sol having functions for Transactions of NFTs.Now during creating a NFT the digital file is uploaded on ipfs server and it in response provides the hash value of that file which after sometimes is stored as a transaction in local blockchain node which is provided by Hardhat development framework.Also smart contracts are deployed on the local blockchain network which validates every transaction according to some condition which is value of that NFT in this section.Also all these transactions are done through Metamask for transfer of Ethers for the transaction to happen.

## TESTING DETAILS

```
const { expect } = require("chai");


const toWei = (num) => ethers.utils.parseEther(num.toString())
const fromWei = (num) => ethers.utils.formatEther(num)


describe("NFTMarketplace", function () {


  let NFT;
  let nft;
  let Marketplace;
  let marketplace
  let deployer;
  let addr1;
  let addr2;
  let addrs;
  let feePercent = 1;
  let URI = "sample URI"


  beforeEach(async function () {
```

- **Get the ContractFactories and Signers here.**

```
   NFT = await ethers.getContractFactory("NFT");
   Marketplace = await ethers.getContractFactory("Marketplace");
```

```
  [deployer, addr1, addr2, ...addrs] = await ethers.getSigners();
```

- **To deploy our contracts**

```
  nft = await NFT.deploy();
  marketplace = await Marketplace.deploy(feePercent);
});

describe("Deployment", function () {

  it("Should track name and symbol of the nft collection", async function () {
```

- **This test expects the owner variable stored in the contract to be equal to our Signer's owner.**

```
    const nftName = "DApp NFT"
    const nftSymbol = "DAPP"
    expect(await nft.name()).to.equal(nftName);
    expect(await nft.symbol()).to.equal(nftSymbol);
  });

  it("Should track feeAccount and feePercent of the marketplace", async function
() {
    expect(await marketplace.feeAccount()).to.equal(deployer.address);
    expect(await marketplace.feePercent()).to.equal(feePercent);
  });
```

```
});

describe("Minting NFTs", function () {

  it("Should track each minted NFT", async function () {
    • addr1 mints an nft
    await nft.connect(addr1).mint(URI)
    expect(await nft.tokenCount()).to.equal(1);
    expect(await nft.balanceOf(addr1.address)).to.equal(1);
    expect(await nft.tokenURI(1)).to.equal(URI);
    • addr2 mints an nft
    await nft.connect(addr2).mint(URI)
    expect(await nft.tokenCount()).to.equal(2);
    expect(await nft.balanceOf(addr2.address)).to.equal(1);
    expect(await nft.tokenURI(2)).to.equal(URI);
  });
})

describe("Making marketplace items", function () {
  let price = 1
  let result
  beforeEach(async function () {
    • addr1 mints an nft
```

```
    await nft.connect(addr1).mint(URI)
```

- **addr1 approves marketplace to spend nft**

```
    await nft.connect(addr1).setApprovalForAll(marketplace.address, true)
  })
```

```
  it("Should track newly created item, transfer NFT from seller to marketplace and emit Offered event", async function () {
```

- **addr1 offers their nft at a price of 1 ether**

```
    await expect(marketplace.connect(addr1).makeItem(nft.address, 1 , toWei(price)))
      .to.emit(marketplace, "Offered")
      .withArgs(
        1,
        nft.address,
        1,
        toWei(price),
        addr1.address
      )
```

- **Owner of NFT should now be the marketplace**

```
    expect(await nft.ownerOf(1)).to.equal(marketplace.address);
```

- **Item count should now equal 1**

```
    expect(await marketplace.itemCount()).to.equal(1)
```

- **Get item from items mapping then check fields to ensure they are correct**

```
    const item = await marketplace.items(1)
    expect(item.itemId).to.equal(1)
    expect(item.nft).to.equal(nft.address)
    expect(item.tokenId).to.equal(1)
    expect(item.price).to.equal(toWei(price))
    expect(item.sold).to.equal(false)
  });

  it("Should fail if price is set to zero", async function () {
    await expect(
      marketplace.connect(addr1).makeItem(nft.address, 1, 0)
    ).to.be.revertedWith("Price must be greater than zero");
  });

});
describe("Purchasing marketplace items", function () {
  let price = 2
  let fee = (feePercent/100)*price
  let totalPriceInWei
  beforeEach(async function () {
```

- **addr1 mints an nft**

```
    await nft.connect(addr1).mint(URI)
```

- **addr1 approves marketplace to spend tokens**

```
    await nft.connect(addr1).setApprovalForAll(marketplace.address, true)
```

- **addr1 makes their nft a marketplace item.**

```
    await marketplace.connect(addr1).makeItem(nft.address, 1 , toWei(price))
  })
  it("Should update item as sold, pay seller, transfer NFT to buyer, charge fees and emit a Bought event", async function () {
    const sellerInitalEthBal = await addr1.getBalance()
    const feeAccountInitialEthBal = await deployer.getBalance()
```

- **fetch items total price (market fees + item price)**

```
    totalPriceInWei = await marketplace.getTotalPrice(1);
```

- **addr 2 purchased items.**

```
    await expect(marketplace.connect(addr2).purchaseItem(1, {value: totalPriceInWei}))
    .to.emit(marketplace, "Bought")
      .withArgs(
        1,
        nft.address,
        1,
        toWei(price),
        addr1.address,
        addr2.address
      )
```

```
const sellerFinalEthBal = await addr1.getBalance()

const feeAccountFinalEthBal = await deployer.getBalance()
```

- **Item should be marked as sold**

```
expect((await marketplace.items(1)).sold).to.equal(true)
```

- **Seller should receive payment for the price of the NFT sold.**

```
expect(+fromWei(sellerFinalEthBal)).to.equal(+price +
+fromWei(sellerInitalEthBal))
```

- **feeAccount should receive fee**

```
expect(+fromWei(feeAccountFinalEthBal)).to.equal(+fee +
+fromWei(feeAccountInitialEthBal))
```

- **The buyer should now own the nft**

```
expect(await nft.ownerOf(1)).to.equal(addr2.address);

})

it("Should fail for invalid item ids, sold items and when not enough ether is
paid", async function () {
```

- **fails for invalid item ids**

```
await expect(

  marketplace.connect(addr2).purchaseItem(2, {value: totalPriceInWei})

).to.be.revertedWith("item doesn't exist");

await expect(

  marketplace.connect(addr2).purchaseItem(0, {value: totalPriceInWei})

).to.be.revertedWith("item doesn't exist");
```

- **Fails when not enough ether is paid with the transaction.**
- **In this instance, fails when the buyer only sends enough ether to cover the price of the nft not the additional market fee.**

```
    await expect(
      marketplace.connect(addr2).purchaseItem(1, {value: toWei(price)})
    ).to.be.revertedWith("not enough ether to cover item price and market fee");
```

- **addr2 purchases item 1**

```
    await marketplace.connect(addr2).purchaseItem(1, {value: totalPriceInWei})
```

- **addr3 tries purchasing item 1 after its been sold**

```
    const addr3 = addrs[0]
    await expect(
      marketplace.connect(addr3).purchaseItem(1, {value: totalPriceInWei})
    ).to.be.revertedWith("item already sold");
  });
 })
})
```

# REFERENCES


- https://opensea.io/
- https://ethereum.org/en/nft/
- https://hardhat.org/
- https://docs.polygon.technology/docs/develop/hardhat
- https://ipfs.io/
- https://docs.ipfs.io/how-to/mint-nfts-with-ipfs/#minty
- https://v5.reactrouter.com/web/guides/quick-start
- https://www.npmjs.com/package/react-router-dom
- https://sites.google.com/metamskloginx.com/metamask-wallet/home
- https://docs.ethers.io/v5/
- https://docs.soliditylang.org/en/v0.8.14/