
**Project Report : Special Topics On CSE (CSE-490)**
**Semester :Spring 2023**

# Title: NFT marketplace development using Hardhat, Solidity, Nodejs and React

**Submitted To: Dr.Rubaiyat Islam**
**E-mail: rubaiyatislam17002sets@iub.edu.bd**

**Submitted By**

| Name | ID | E-mail |
|---|---|---|
| Faisal Bhuiyan | 1921298 | 1921298@iub.edu.bd |
| Mustak Ali Virani | 1810656 | 1810656@iub.edu.bd |
| Showmit Roy Neel | 1811135 | 1811135@iub.edu.bd |
| Samiul Haque Sunny | 1820547 | 1820547@iub.edu.bd |
| Avishak Sikdar | 2022759 | 2022759@iub.edu.bd |

Date of Submission: 07/05/2023

# Table of Contents

# Abstract

Non-fungible tokens (NFTs) have gained significant attention in recent times due to their ability to represent unique digital assets such as artworks, music, and even tweets. As a result, the development of NFT marketplaces has become increasingly popular, providing a platform for creators and collectors to trade these assets. The NFT marketplace is designed to allow users to create, buy, and sell NFTs securely and efficiently. Users can create NFTs by uploading their digital assets and defining the characteristics of the asset, such as its name, description, and metadata. They can also set the price for the NFT and choose the type of auction that they prefer.

# Introduction

NFTs (non-fungible tokens) have exploded in popularity in recent years, thanks in large part to the growing interest in blockchain technology. As a result, the demand for NFT marketplaces has grown significantly, as artists, collectors, and investors look for a platform to buy, sell, and trade NFTs. If you're looking to develop an NFT marketplace, you'll need to have a solid understanding of the technology involved, including Hardhat, Solidity, Nodejs, and React. Hardhat is a popular development environment for Ethereum, designed to streamline the development process by providing a range of tools and services. Solidity is the programming language used to write smart contracts for the Ethereum blockchain, which are the backbone of any NFT marketplace. Nodejs is a popular server-side platform that can be used to develop the backend of your marketplace, while React is a widely-used front-end library for building user interfaces.

By combining these technologies, one can create a robust and reliable NFT marketplace that can handle a high volume of transactions securely and efficiently. In this report, we'll explore the key steps involved in developing an NFT marketplace using Hardhat, Solidity, Nodejs, and React, and provide you with the knowledge you need to get started on your own project.

# Why Build An NFT Marketplace?

The question of digital ownership and provenance is one issue that NFT marketplaces address in the real world. It can be challenging for creators to maintain ownership and control of their work given the widespread availability of digital content and the ease with which it can be reproduced and shared. NFTs give artists a mechanism to claim ownership of their digital assets and follow the provenance of those items across time.

NFTs have also given authors in the digital sphere a new source of income. Creators can monetize their work and forge close relationships with their followers and collectors by offering their digital assets as NFTs. Additionally, NFT markets have opened up a new market for collectors who want to acquire special digital assets and are prepared to pay a premium for them.

NFT marketplaces have the ability to fundamentally alter how we perceive digital ownership and asset value. They offer a higher level of security and transparency for digital transactions and open up new possibilities for both makers and collectors.

# Designing A Token

Designing a Non-Fungible Token (NFT) involves quite a few steps.  Here are few general steps which are followed to design a NFT:

## Step-1:

Choose the NFT platform: There are several NFT platforms available such as Ethereum, Binance Smart Chain, Open sea and others. We need to choose a platform that suits our requirements.

## Step-2:

Defining the NFT: We need to decide what our NFT will represent. It can be anything from artwork, music, video, or a digital asset.

## Step-3:

Defining our NFT: We have to decide what our NFT will represent. It can be anything from artwork, music, video, or a digital asset.

### Step-4:

Choose our NFT type: There are two types of NFTs: ERC-721 and ERC-1155. ERC-721 NFTs are unique and indivisible, whereas ERC-1155 NFTs can represent multiple tokens of the same type.

### Step-5:

Set our NFT price: We need to decide how much we want to sell our NFT for. We can set a fixed price or let the market decide the price through bidding.

### Step-6:

Launching our NFT: Once our NFT is created, we can launch it on our chosen platform. We can promote our NFT on social media platforms, NFT marketplaces, and other relevant communities to increase its visibility.

It's worth noting that designing an NFT involves more than just the technical aspects. We should also consider the value proposition of our NFT and ensure that it resonates with your target audience. Additionally, we should ensure that our NFT is unique and adds value to the NFT ecosystem.

## Designing The Market Place

Designing an NFT marketplace involves several steps and requires knowledge of different technologies. Here we've written down  the general steps of how to design an NFT marketplace using Hardhat, Solidity, Nodejs, and React:

### Step-1

We have to Set up the development environment by installing Node.js, Hardhat, and other necessary tools like React.

## Step-2

We need to write the smart contract: Use Solidity to write the smart contract that defines the NFTs, their ownership, and how they can be transferred. This contract will be deployed on the Ethereum blockchain.

## Step-3

We have to deploy the smart contract: Deploy the smart contract using Hardhat or any other Ethereum development tool.

## Step-4

Setting up the backend: Build the backend using Node.js to handle requests and communicate with the Ethereum blockchain.

## Step-5

Build the frontend: Use React to build the frontend of the marketplace, including user interfaces for buying and selling NFTs, displaying NFT metadata, and managing user accounts.

## Step-6

Test and deploy: By testing the marketplace thoroughly and deploying it to a production environment.

# NFT And NFT Marketplace Mechanism (How Does It Work?)

Non-fungible tokens (NFT) , are unique digital assets that are stored on a blockchain. In an NFT marketplace, users can buy and sell these digital assets using cryptocurrencies. Here are steps that explain the mechanism of NFTs and its marketplace.

## Step-1

The creator of an NFT needs to create a unique digital asset and upload it to the marketplace. This can be anything from a piece of digital art to a tweet.

## Step-2

The NFT is stored on a blockchain, which serves as a public ledger that records ownership and transaction history.

## Step-3

Interested buyers can browse the marketplace for NFTs they want to purchase. Each NFT has a unique identifier that distinguishes it from all other NFTs.

## Step-4

When a buyer decides to purchase an NFT, they send a cryptocurrency payment to the seller's wallet address. This payment is typically made in Ethereum or another cryptocurrency that is compatible with the blockchain used by the marketplace.

## Step-5

Once the payment is confirmed, the ownership of the NFT is transferred to the buyer's wallet address. This transfer is recorded on the blockchain, which ensures that the ownership of the NFT is transparent and verifiable.

## Step-6

The buyer can then hold onto the NFT as a digital asset, or they can sell it on the marketplace to another interested buyer.

Overall, NFT marketplaces provide a platform for creators and collectors to exchange unique digital assets in a secure and transparent way. These files now have a unique digital identifier

that cannot be copied, substituted, or subdivided, and that is used to certify authenticity and ownership.

# Smart Contracts Used In The Project

This is a Solidity smart contract code that creates an NFT (Non-Fungible Token) smart contract. Here is a brief explanation of the code:

```
pragma solidity ^0.8.4;
```
This particular line indicates the kind of the Solidity programming language that the code has been written in.
```
import "@openzeppelin/contracts/token/ERC721/extensions/ERC721URIStorage.sol";
```
This line overrides the ERC721URIStorage contract from the OpenZeppelin library. ERC721URIStorage is a Solidity smart contract that has been pre-written and provides functionality for NFTs(Non-Fungible Tokens).

```
contract NFT is ERC721URIStorage {
   uint public tokenCount;
   constructor() ERC721("DApp NFT", "DAPP"){}
   function mint(string memory _tokenURI) external returns(uint) {
      tokenCount ++;
      _safeMint(msg.sender, tokenCount);
      _setTokenURI(tokenCount, _tokenURI);
      return(tokenCount);
   }
}
```
This is the main portion of the contract. The `NFT` contract takes over from the `ERC721URIStorage` contract, that provides NFT functionality.

The total number of NFTs that have been issued is kept track of by the 'tokenCount' variable.

The NFT is initialized using the "constructor()" function with the names "DApp NFT" and "DAPP."

By supplying the '_tokenURI' parameter, which is a special token identification, to the'mint()' method, users can mint new NFTs.

The 'tokenCount' variable is increased inside the'mint()' function, and the '_safeMint()' function is called to generate a new NFT and assign it to the user who ran the function. The token URI, a special identifier for the NFT that may be used to get its metadata, is set using the '_setTokenURI()' method. The function then returns the value "tokenCount," which is the special ID of the recently created NFT.

==========

```
contract Marketplace is ReentrancyGuard {

    // Variables
    address payable public immutable feeAccount; // the account that
receives fees
    uint public immutable feePercent; // the fee percentage on sales
    uint public itemCount;

    struct Item {
        uint itemId;
        IERC721 nft;
        uint tokenId;
        uint price;
        address payable seller;
        bool sold;
    }

    // itemId -> Item
    mapping(uint => Item) public items;

    event Offered(
        uint itemId,
        address indexed nft,
        uint tokenId,
        uint price,
        address indexed seller
    );
    event Bought(
        uint itemId,
        address indexed nft,
```

```
        uint tokenId,
        uint price,
        address indexed seller,
        address indexed buyer
    );

    constructor(uint _feePercent) {
        feeAccount = payable(msg.sender);
        feePercent = _feePercent;
    }

    // Make item to offer on the marketplace
    function makeItem(IERC721 _nft, uint _tokenId, uint _price) external
nonReentrant {
        require(_price > 0, "Price must be greater than zero");
        // increment itemCount
        itemCount ++;
        // transfer nft
        _nft.transferFrom(msg.sender, address(this), _tokenId);
        // add new item to items mapping
        items[itemCount] = Item (
            itemCount,
            _nft,
            _tokenId,
            _price,
            payable(msg.sender),
            false
        );
        // emit Offered event
        emit Offered(
            itemCount,
            address(_nft),
            _tokenId,
            _price,
            msg.sender
        );
    }

    function purchaseItem(uint _itemId) external payable nonReentrant {
```

```
        uint _totalPrice = getTotalPrice(_itemId);
        Item storage item = items[_itemId];
        require(_itemId > 0 && _itemId <= itemCount, "item doesn't
exist");
        require(msg.value >= _totalPrice, "not enough ether to cover item
price and market fee");
        require(!item.sold, "item already sold");
        // pay seller and feeAccount
        item.seller.transfer(item.price);
        feeAccount.transfer(_totalPrice - item.price);
        // update item to sold
        item.sold = true;
        // transfer nft to buyer
        item.nft.transferFrom(address(this), msg.sender, item.tokenId);
        // emit Bought event
        emit Bought(
            _itemId,
            address(item.nft),
            item.tokenId,
            item.price,
            item.seller,
            msg.sender
        );
    }
    function getTotalPrice(uint _itemId) view public returns(uint){
        return((items[_itemId].price*(100 + feePercent))/100);
    }
}
```

A smart contract for a market where users can purchase and sell NFTs is contained in this code. Let's go step-by-step through the code:

1.Importing "@openzeppelin/contracts/token/ERC721/IERC721.sol": This imports the OpenZeppelin library's IERC721 interface, which is used to communicate with ERC721 tokens.

2. "miport "@openzeppelin/contracts/security/ReentrancyGuard.sol";" This imports the ReentrancyGuard contract, which guards against reentrancy attacks, from the OpenZeppelin library.

3. Importing "hardhat/console.sol"; This imports the Hardhat console module, which is useful for debugging.

4. The statement "contract Marketplace is ReentrancyGuard" creates a new contract with the name Marketplace that derives from the ReentrancyGuard contract.

5. 'address payable public immutable feeAccount;' declares an immutable payable public variable named feeAccount that will contain the address of the account that will be responsible for handling marketplace fees.

6. the phrase "uint public immutable feePercent;" This declares the percentage of the fee that will be charged on each sale as an immutable public variable named feePercent.

7. `uint public itemCount;`:  Declares a public variable called itemCount, it holds the number of items that exist in the marketplace.

8. "struct Item{...}"" This specifies the creation of a new struct called object, which will store the details of each object being sold.

9. public items; mapping(uint => Item); This declares the public item ID to Item struct mapping, or items, for short.

10. `event Offered(...)`: It declares an event called Offered, it’s emitted when a new item is added to the marketplace.

11. `event Bought(...)`: It declares an event called Bought, that’ll be emitted when an item is bought from the marketplace.

12. `constructor(uint _feePercent)`: It is the constructor function for the Marketplace contract. It takes in one argument, which is mainly the fee percentage to be charged on each sale.

13. "function makeItem(IERC721_nft, uint_tokenId, uint_price) external nonReentrant... " Using this feature, a new product can be added to the marketplace. The ERC721 token contract, the token ID, and the item's price are the three parameters it accepts. The itemCount variable is increased after first determining whether the price is more than zero. The NFT is subsequently transferred from the seller to the contract for the marketplace, the new item is added to the items mapping, and the Offered event is then emitted.

14. "function purchaseItem(uint _itemId) external payable nonReentrant": Using this feature, one can purchase something from the market. The ID of the object to be purchased is the only argument it accepts. It initially verifies the existence of the item, the buyer's ability to pay the price and the marketplace charge, and whether the item has not already been sold. It then

identifies the item as sold, transfers the NFT from the marketplace contract to the buyer, transfers the item's price to the seller, transfers the marketplace fee to the feeAccount, and emits the Bought event.

15. function "getTotalPrice(uint _itemId) view public returns(uint)" The overall price of an item, including the marketplace fee, is determined using this function. The item's ID is the only argument it accepts. It provides an uint number representing the final cost.

# Conclusion

NFT or Non-Fungible tokens are records on the blockchain that store the information about the digital file, who created it, who linked it to the blockchain, and who bought it and who has owned it through its history.  In the real world NFT marketplace addresses the issue of digital ownership and provenance  It can be challenging for creators to maintain ownership and control of their work given the widespread availability of digital content and how easily it can be shared and reproduced.

NFTs give artists a mechanism to claim ownership of their digital assets and follow the provenance of those items across time. For collectors who want to buy unique digital assets and are willing to pay a premium for them, NFT exchanges have created a new market.

NFT marketplaces have the power to radically change our understanding of digital asset ownership and value. For digital transactions, they provide a better level of security and transparency and present fresh opportunities for both creators and collectors.

# References


https://en.wikipedia.org/wiki/Non-fungible_token

https://justcreative.com/why-do-people-buy-nfts/

https://projectswatches.com/blogs/news/what-are-nfts-and-why-should-the-art-world-take-note?utm_campaign=gs-2021-05-08&utm_source=google&utm_medium=smart_campaign&gclid=Cj0

KCQjw9deiBhC1ARIsAHLjR2AqNGW8HnHXy2yNEH5BdINeUu_JYsXqcQ2-YAjr4QJ4rYcIztXqPRQaAhINEALw_wcB

https://www.investopedia.com/non-fungible-tokens-nft-5115211

https://nftplazas.com/learn-about-nfts/why-do-people-buy-nfts/