NFT Marketplace Using Blockchain & IPFS

1. Introduction

The rapid growth of blockchain technology has enabled decentralized applications that offer transparency, immutability, and security. One popular use case is the creation and trading of Non-Fungible Tokens (NFTs). NFTs represent unique digital assets stored on a blockchain and can include images, videos, music, and other digital collectibles.

This project is an **NFT Marketplace** built with **React (frontend)**, **Solidity (smart contract)**, **Ether.js**, and **IPFS** for decentralized storage.
The platform allows users to **mint NFTs**, **upload metadata to IPFS**, and **display created NFTs** on a marketplace dashboard.

2. Objective

The main objectives of the project are:

- To design and deploy a **smart contract** for minting NFTs.
- To enable users to **upload files (images)** to IPFS for decentralized storage.
- To create a **React-based web application** allowing users to mint NFTs.
- To develop a dashboard to **display NFTs** stored on the blockchain.
- To ensure **full integration** between frontend, backend, IPFS, and smart contract.

3. Technologies Used

Component	Technology
Frontend	React.js, Vite
Smart Contract	Solidity (ERC-721 standard)
Blockchain Interaction	Ether.js
Storage	IPFS using `ipfs-http-client`

Blockchain Network	Ethereum / Polygon Testnet
Development Environment	Hardhat
Wallet Integration	MetaMask
Node Package Manager	NPM

4. System Architecture

The architecture consists of:

1. **User Interface (React + Vite)**
   Users upload images, enter NFT details, and mint the NFT.

2. **IPFS Storage**
   Image files and metadata JSON are uploaded to IPFS using the IPFS HTTP client.

3. **Smart Contract (Solidity)**
   The ERC-721 NFT contract handles:

   - Minting NFTs
   - Storing tokenURI
   - Returning user-owned NFTs

4. **Blockchain (Polygon/Ethereum Testnet)**
   Contract deployed to a test network.

5. **MetaMask**
   Used for user authentication and transaction signing.

5. Project Modules

5.1 NFT Minting Module

- Users upload an image.
- Enter NFT name & description.

- Metadata JSON is generated and uploaded to IPFS.
- Smart contract function `mintNFT()` is called.
- NFT is created on blockchain.

5.2 IPFS Upload Module

- Takes the file and stores it on IPFS.
- Returns a permanent CID link (e.g., ipfs://Qm...).
- Metadata JSON stored similarly.

5.3 Blockchain Interaction Module

- Uses `ethers.js` for:
    - Connecting to MetaMask
    - Fetching account
    - Executing contract methods
    - Reading token URIs

5.4 Dashboard / NFT Viewer

- Fetches all minted NFTs.
- Displays them with image + title + description.

6. Smart Contract Overview

- Contract Type: **ERC-721 NFT**
- Key Functions:
    - `mintNFT(address to, string memory tokenURI)`
    - `tokenURI(uint256 tokenId)`
    - `totalSupply()`
    - `getMyNFTs()`

The smart contract ensures each NFT is unique and owned by a specific wallet.

7. Frontend Overview

- React UI with:
    - NFT Mint Form
    - Image Uploader
    - NFT Preview
    - Dashboard Display
- MetaMask integration
- Responsive UI

Pages:

1. **Home Page**
2. **Mint NFT Page**
3. **My NFTs Dashboard**

8. Working Process / Flowchart

Step 1: Connect MetaMask

User connects wallet using Ether.js.

Step 2: Upload Image

User selects an image and uploads it to IPFS.

Step 3: Generate Metadata

Metadata JSON:

```
{
  "name": "NFT Name",
  "description": "Description",
  "image": "ipfs://CID"
}
```

Step 4: Mint NFT

Smart contract stores:

- Wallet address
- Token ID
- tokenURI (IPFS metadata link)

Step 5: Display NFTs

Frontend retrieves tokenURIs → fetches metadata → displays NFT cards.

9. Features Implemented

- Decentralized NFT minting
- File upload to IPFS
- ERC-721 Smart Contract
- Wallet-based authentication
- Modern React UI
- Blockchain-based ownership
- Real-time dashboard

10. Advantages of the System

- **Decentralized storage** using IPFS
- **Complete transparency** using blockchain
- **Immutable ownership** of NFTs
- **Secure marketplace** for digital assets
- **Low cost** (testnet) deployment

11. Challenges Faced

- Handling IPFS upload delays
- Integrating React with MetaMask
- Smart contract debugging
- Network and gas fee issues
- Metadata validation errors

12. Applications

- Digital art marketplace
- Music and video tokenization
- Gaming assets trading
- Certificate/ID system
- Collectible trading platform

13. Future Enhancements

- Add NFT selling/buying feature
- Enable auctions
- Add multi-wallet support
- Deploy on Polygon mainnet
- Add user profiles
- Add search & filter system

- Improve UI/UX

14. Conclusion

This project successfully demonstrates how blockchain, IPFS, and smart contracts can be integrated to build a functional **NFT Marketplace**. It provides secure minting, decentralized storage, and NFT visualization. The system is scalable, transparent, and suitable for real-world decentralized applications.