NFT Platform Project Guide

Build Digital Ownership Infrastructure

BSc Blockchain, Crypto Economy & NFTs

FS2026

Learning Objectives

By completing this project, you will:

1. Understand ERC-721 and ERC-1155 standards
2. Implement proper metadata handling
3. Store NFT data on IPFS
4. Create marketplace functionality

Related Lessons: L18, L21, L22, L23, L25, L26

Problem: How do we create verifiable digital ownership and scarcity?

NFT Standards Comparison

Feature	ERC-721	ERC-1155
Token Type	Non-fungible only	Fungible + Non-fungible
Uniqueness	Each token unique	Multiple of same ID
Gas Efficiency	Higher per transfer	Lower (batch support)
Use Case	1/1 Art, PFPs	Game items, editions
Metadata	Per token URI	Shared URI template
Batch Transfer	Not native	Native support

Decision Rule:

- Use ERC-721 for unique collectibles
- Use ERC-1155 for gaming items or editions

Choose the standard based on your use case requirements

Basic ERC-721 Implementation

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.20;

import "@openzeppelin/contracts/token/ERC721/ERC721.sol";
import "@openzeppelin/contracts/token/ERC721/extensions/ERC721URIStorage.sol";
import "@openzeppelin/contracts/access/Ownable.sol";

contract MyNFT is ERC721, ERC721URIStorage, Ownable {
    uint256 private _nextTokenId;

    constructor() ERC721("My NFT", "MNFT") Ownable(msg.sender) {}

    function mint(address to, string memory uri)
        public onlyOwner returns (uint256) {
        uint256 tokenId = _nextTokenId++;
        _safeMint(to, tokenId);
        _setTokenURI(tokenId, uri);
        return tokenId;
    }
}
```

OpenZeppelin provides secure, audited base contracts

Metadata Standard

OpenSea-Compatible JSON:

```
{
  "name": "Artwork #1",
  "description": "A digital artwork",
  "image": "ipfs://Qm.../image.png",
  "attributes": [
    {
      "trait_type": "Background",
      "value": "Blue"
    },
    {
      "trait_type": "Rarity",
      "value": "Legendary"
    }
  ]
}
```

Attribute Types:

- String: Background, Rarity
- Number: Power, Level
- Boost: Speed +10%
- Date: Birthday

Required Fields:

- name
- description
- image

Following standards ensures compatibility with marketplaces

IPFS Storage

Why IPFS?

- **Content addressing:** Files identified by hash (CID)
- **Immutability:** Hash changes if content changes
- **Decentralization:** No single point of failure

Storage Flow:

1. Upload image to IPFS → Get image CID
2. Create metadata JSON with image CID
3. Upload metadata to IPFS → Get metadata CID
4. Set tokenURI to `ipfs://[metadata CID]`

Pinning Services:

- Pinata (recommended for beginners)
- Infura IPFS
- NFT.Storage (free for NFTs)

Always pin your content - unpinned data may disappear

Marketplace Contract

Core Functions:

- `listNFT(nftContract, tokenId, price)` - List for sale
- `buyNFT(listingId)` - Purchase listed NFT
- `cancelListing(listingId)` - Remove listing

Fee Structure:

- Platform fee: 2.5% (basis points: 250)
- Creator royalty: 5-10% (ERC-2981)
- Seller receives: Price - fees

Security Considerations:

- Reentrancy guards on buy function
- Check NFT approval before listing
- Verify ownership on cancel

Marketplace contracts handle significant value - security is critical

Royalty Standard (ERC-2981)

What It Does:

- Standardizes royalty information retrieval
- Works across all marketplaces
- Creator earns on every resale

Implementation:

```
function royaltyInfo(uint256 tokenId, uint256 salePrice)
    returns (address receiver, uint256 royaltyAmount)
{
    return (creator, salePrice * 500 / 10000);  // 5%
}
```

Note: ERC-2981 is informational only - marketplaces must voluntarily enforce it.

Royalties align creator incentives with long-term collection value

Utility NFTs

Beyond Speculation: Real Utility

Access Control

- Membership passes
- Event tickets
- Gated content

Example: Membership NFT

- NFT grants 1-year access
- Expiry stored on-chain
- Renewable via payment
- Transferable between users

Gaming

- In-game items
- Character skins
- Land ownership

Utility creates intrinsic value beyond speculation

Apply the 6 Questions:

Question	NFT Platform Answer
PROBLEM	Prove digital ownership and scarcity
INCENTIVES	Creators earn royalties, collectors speculate
BENEFITS/COSTS	Verifiable provenance; gas costs, metadata risk
FAILURE MODE	Metadata loss, rug pulls, smart contract bugs
DESIGN CHOICES	On-chain vs off-chain metadata
ALTERNATIVES	Traditional digital rights management

NFTs shift ownership proofs from legal to cryptographic

Implementation Checklist

Phase 1: Basic NFT

- ☐ Deploy ERC-721 contract
- ☐ Mint test NFTs
- ☐ Verify tokenURI returns correctly

Phase 2: IPFS Integration

- ☐ Upload images to IPFS
- ☐ Create metadata JSON files
- ☐ Link metadata to tokens

Phase 3: Marketplace

- ☐ Deploy marketplace contract
- ☐ List NFTs for sale
- ☐ Test buy/cancel flows

Phase 4: Advanced

- ☐ Add royalty support (ERC-2981)
- ☐ Implement reveal mechanism
- ☐ Create utility functions

Resources

Standards:

- EIP-721: `eips.ethereum.org/EIPS/eip-721`
- EIP-1155: `eips.ethereum.org/EIPS/eip-1155`
- EIP-2981: `eips.ethereum.org/EIPS/eip-2981`

Tools:

- Pinata: `pinata.cloud`
- NFT.Storage: `nft.storage`
- OpenSea Docs: `docs.opensea.io`

Project Materials:

- Notebook: `projects/notebooks/02_nft_platform.ipynb`
- Web: `.../projects/nft-platform/`