



Centurion  
UNIVERSITY  
*Shaping Lives...  
Empowering Communities...*

School: ..... Campus: .....

Academic Year: ..... Subject Name: ..... Subject Code: .....

Semester: ..... Program: ..... Branch: ..... Specialization: .....

Date: .....

## Applied and Action Learning

(Learning by Doing and Discovery)

**Name of the Experiment :** Mint it Yourself – NFT Creation and Deployment

### \* Coding Phase: Pseudo Code / Flow Chart / Algorithm

#### ALGORITHM:

- 1.Initialize environment with Remix IDE and MetaMask on the Sepolia Testnet.
- 2.Write an ERC-721 compliant NFT smart contract using OpenZeppelin libraries.
- 3.Compile the smart contract in Remix using Solidity 0.8.x compiler.
- 4.Deploy the smart contract to the Sepolia Testnet through MetaMask.
- 5.Call the mint function to create an NFT with a specific metadata URI.
- 6.Verify the NFT minting transaction on Etherscan and confirm token ownership.

### \* Software used

- 1.Remix IDE
- 2.MetaMask (Sepolia Testnet)
- 3.OpenZeppelin Contracts
- 4.IPFS / Pinata for storing NFT metadata
- 5.Etherscan Testnet for verification

## \* Testing Phase: Compilation of Code (error detection)

### 1. Compilation of Smart Contract

Open Remix IDE and paste the NFT smart contract code.

Select Solidity compiler version 0.8.20 or higher for compatibility with OpenZeppelin v5.x.

Click Compile MyNFT.sol and ensure no syntax errors occur.

Successful compilation indicates the contract is ready for deployment.

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.20; // use 0.8.20 or higher for OZ v5.x

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

contract MyNFT is ERC721URIStorage, Ownable {
    uint256 public tokenCounter;

    // Pass the deployer address to the Ownable constructor
    constructor() ERC721("MyFirstNFT", "MFN") Ownable(msg.sender) {
        tokenCounter = 0;
    }

    // Function to mint a new NFT
    function mintNFT(address recipient, string memory tokenURI) {
        public
        onlyOwner
        returns (uint256)
    {
        uint256 newTokenId = tokenCounter;
        _safeMint(recipient, newTokenId);
        _setTokenURI(newTokenId, tokenURI);
        tokenCounter = tokenCounter + 1;
        return newTokenId;
    }
}
```

### 2. Deployment on Sepolia Testnet

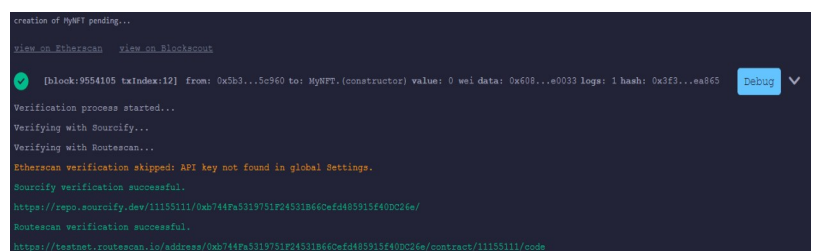
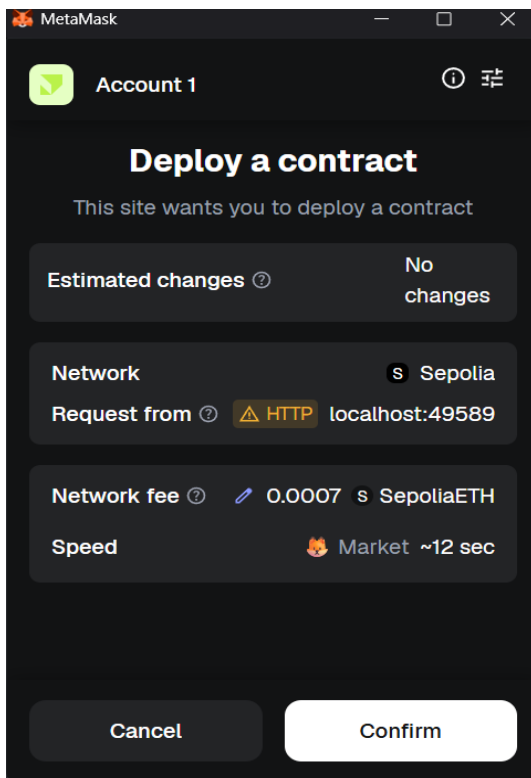
Connect MetaMask to the Sepolia Test Network.

In Remix, choose Injected Provider – MetaMask as the environment.

Click Deploy and approve the transaction in MetaMask.

Once confirmed, Remix displays the contract address under “Deployed Contracts.”

This confirms that the NFT contract is live on the blockchain.



## \* Testing Phase: Compilation of Code (error detection)

### 3. Linking Metadata and Minting NFT

Expand the deployed contract in Remix under Deployed Contracts.

Locate and execute the mintNFT() function.

Input:

Recipient Address: your MetaMask wallet address.

Token URI: the IPFS link to the NFT's metadata JSON file.

Click Transact and confirm the mint transaction in MetaMask.

Once mined, the NFT is minted and assigned to your wallet.

The first screenshot shows the Remix IDE interface. Under the 'DEPLOY & RUN TRANSACTIONS' tab, the 'MINTNFT' function is selected. The 'recipient' field is set to '0x5b329bad3f32ecf4dfedee03580' and the 'tokenURI' field is set to 'uQv19rWa4mcy1Q5KwNSxmyasBu'. The 'transact' button is highlighted. A blue arrow points from this screenshot to the second one.

The second screenshot shows the MetaMask 'Transaction request' dialog. It displays 'Estimated changes' as 'You receive + #0 0xb744F...DC26e'. The network is 'Sepolia', the request is from 'localhost:49589', and it is interacting with 'Alert > 0xb744F...DC26e'. The network fee is '0.0002 SepoliaETH' and the speed is 'Market ~12 sec'. The 'Confirm' button is highlighted. A blue arrow points from this screenshot to the third one.

The third screenshot shows the Remix IDE console. It displays the transaction confirmation: 'transact to MyNFT.mintNFT pending ...'. Below this, it shows the transaction details: '[block:9554125 txIndex:10] from: 0x5b3...5c960 to: MyNFT.mintNFT(address,string) 0xb74...dc26e value: 0 wei data: 0xeac...00000 logs: 2 hash: 0x3fd...47e94'. A green checkmark icon is visible on the left, and a 'Debug' button is on the right.

### 4. Verification of Mint Transaction

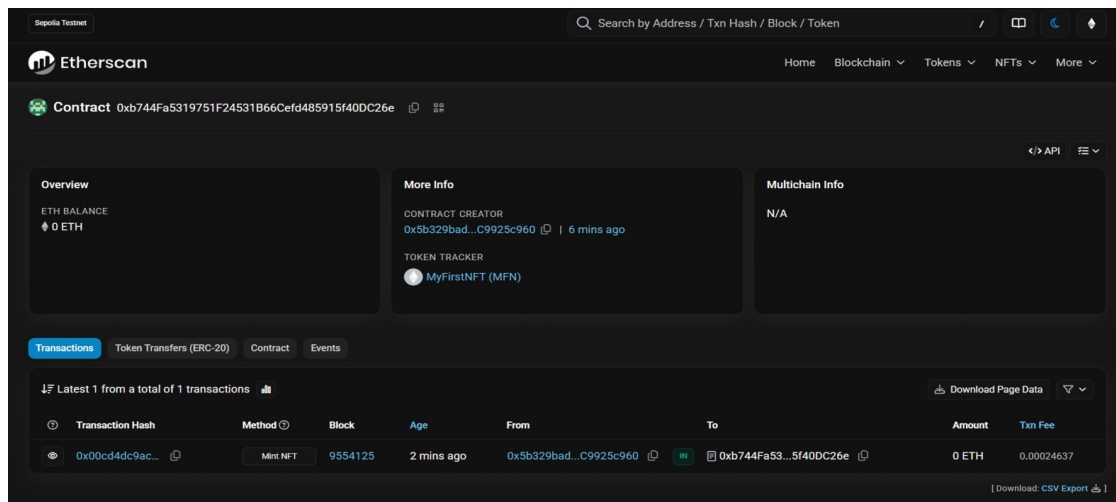
Copy the transaction hash from Remix.

Visit <https://sepolia.etherscan.io> and paste the hash in the search bar.

Check transaction details, including contract interaction and confirmation.

This step verifies successful on-chain recording of the NFT minting.

## \* Testing Phase: Compilation of Code (error detection)



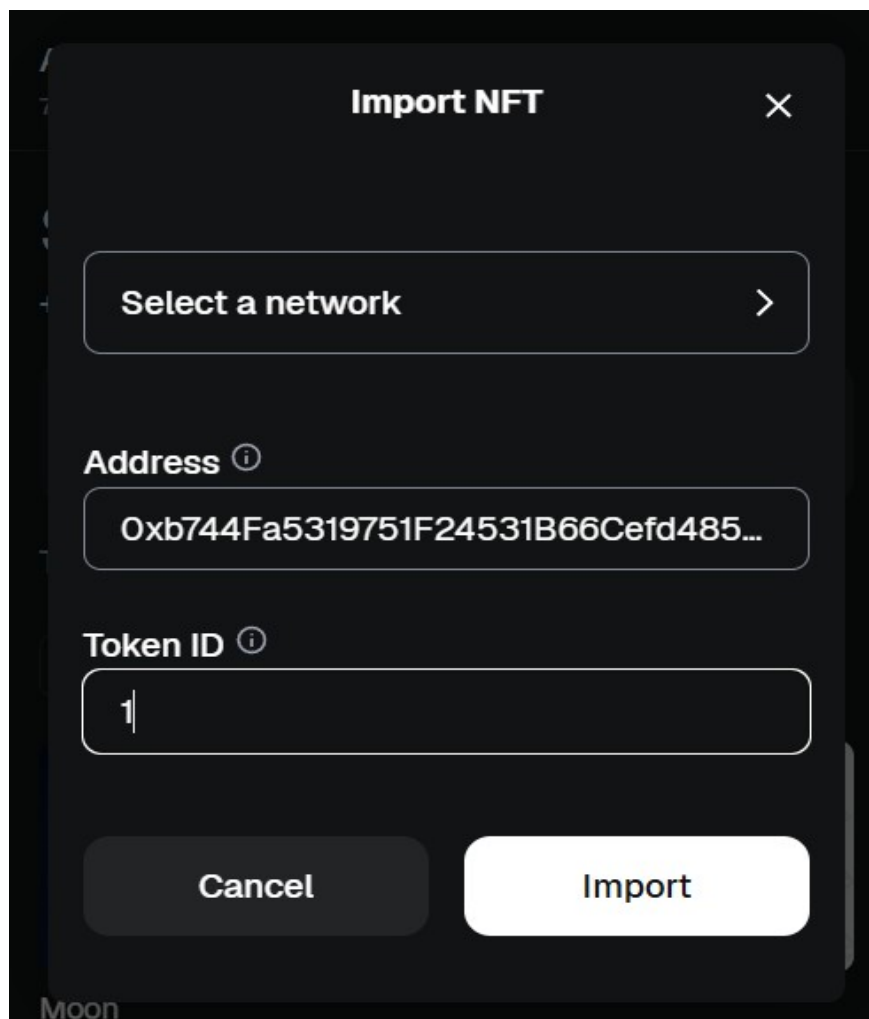
### 5. Viewing the NFT in MetaMask

Open MetaMask → NFTs Tab → Import NFTs.

Enter the contract address and Token ID (e.g., 0).

Your NFT will appear in MetaMask, showing that it's owned by your wallet.

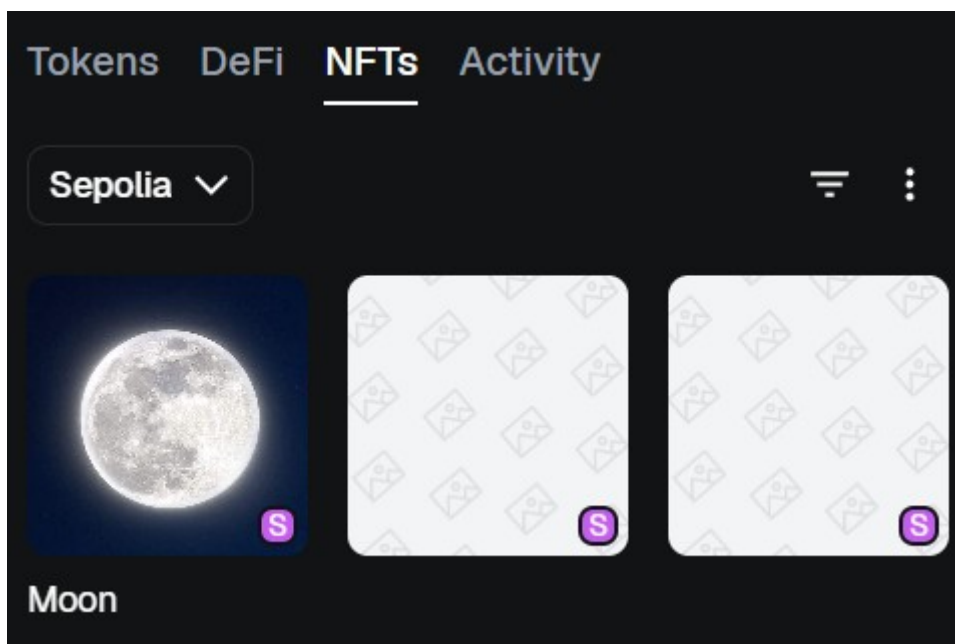
This confirms successful minting and ownership verification.



## \* Implementation Phase: Final Output (no error)

Applied and Action Learning

- NFT was successfully minted and deployed on Sepolia Testnet.
- Metadata (name, image, and description) fetched correctly from IPFS.
- Transaction verified through Etherscan, confirming blockchain interaction.
- Ownership recorded immutably on Ethereum blockchain.



## \* Observations

- 1.NFT smart contract deployed successfully on Sepolia Testnet.
- 2.mintNFT() function executed and minted a new NFT token.
- 3.IPFS metadata correctly linked and displayed (image, name, and description).
- 4.Ownership confirmed through blockchain record and visible in MetaMask.
- 5.The experiment demonstrated decentralized asset creation using ERC-721 standard.

## ASSESSMENT

Rubrics	Full Mark	Marks Obtained	Remarks
Concept	10		
Planning and Execution/ Practical Simulation/ Programming	10		
Result and Interpretation	10		
Record of Applied and Action Learning	10		
Viva	10		
<b>Total</b>	<b>50</b>		

**Signature of the Student:**

Name :

Regn. No. :

**Signature of the Faculty:**

Page No.....

*\* As applicable according to the experiment.  
Two sheets per experiment (10-20) to be used.*