



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

#### ☐ Deploy Contracts

- ☐ Deploy an ERC721 NFT contract (for minting NFTs).
- ☐ Deploy a Marketplace contract that allows listing, buying, cancelling, and updating NFT prices.

#### ☐ Connect Wallet

- ☐ Connect MetaMask on Sepolia Testnet to the frontend DApp.

#### ☐ Approve Marketplace

- ☐ Call `setApprovalForAll(marketplaceAddress, true)` from NFT contract to allow the marketplace to manage NFTs.

#### ☐ List NFT

- ☐ Call `listItem(nftAddress, tokenId, price)` → NFT is listed in marketplace with a unique `listingId`.

#### ☐ Buy NFT

- ☐ Buyer calls `buy(listingId)` while sending ETH equal to the listing price.
- ☐ Ownership of NFT transfers to the buyer.

#### ☐ Cancel Listing

- ☐ Seller can call `cancel(listingId)` to remove their NFT from sale.

#### ☐ Update Price

- ☐ Seller can call `updatePrice(listingId, newPrice)` to change the price of a listed NFT.

#### ☐ Withdraw Proceeds

- ☐ Seller calls `withdrawProceeds()` to claim ETH earned from NFT sales.

### \* Software used

1. MetaMask Wallet
2. Remix IDE.
3. MS Word.
4. Brave for researching.

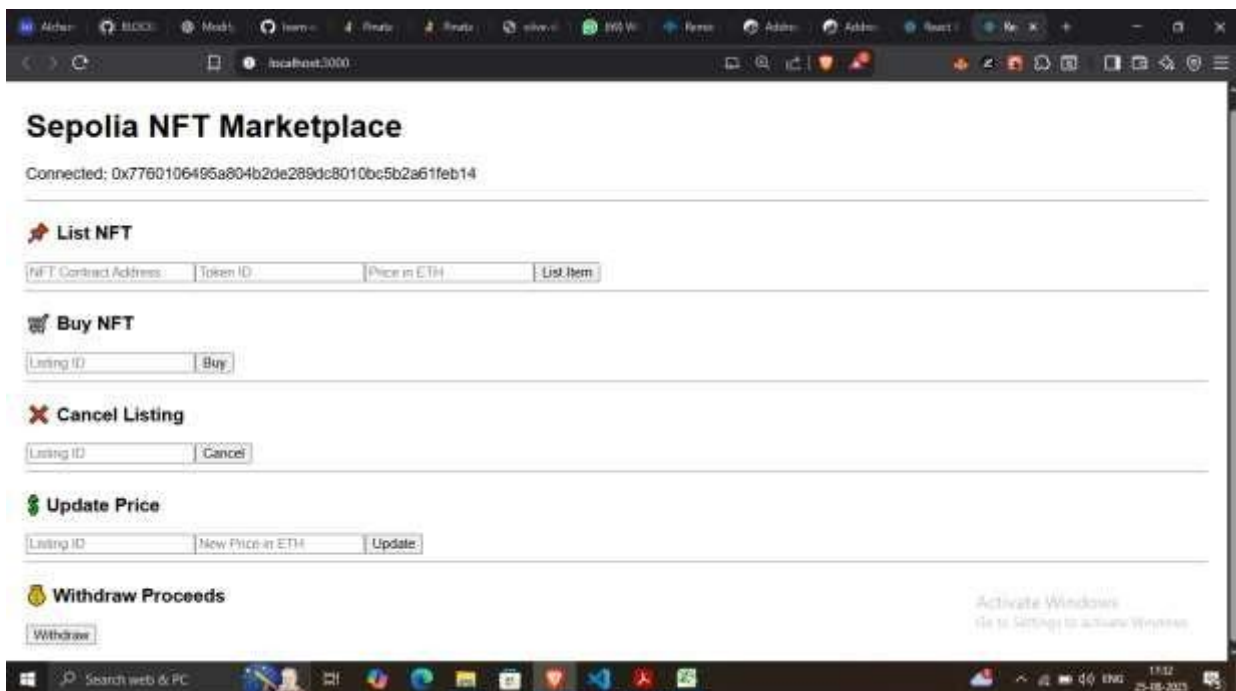
## \* Implementation Phase: Final Output (no error)

- ☐ NFTs can be listed, purchased, cancelled, or updated on Sepolia Testnet.
- ☐ Transactions trigger success messages in the DApp UI (NFT listed successfully, Price updated!, etc.).
- ☐ ETH proceeds are stored securely until withdrawn by sellers.
- ☐ Ownership of NFTs correctly transfers between accounts after purchase.

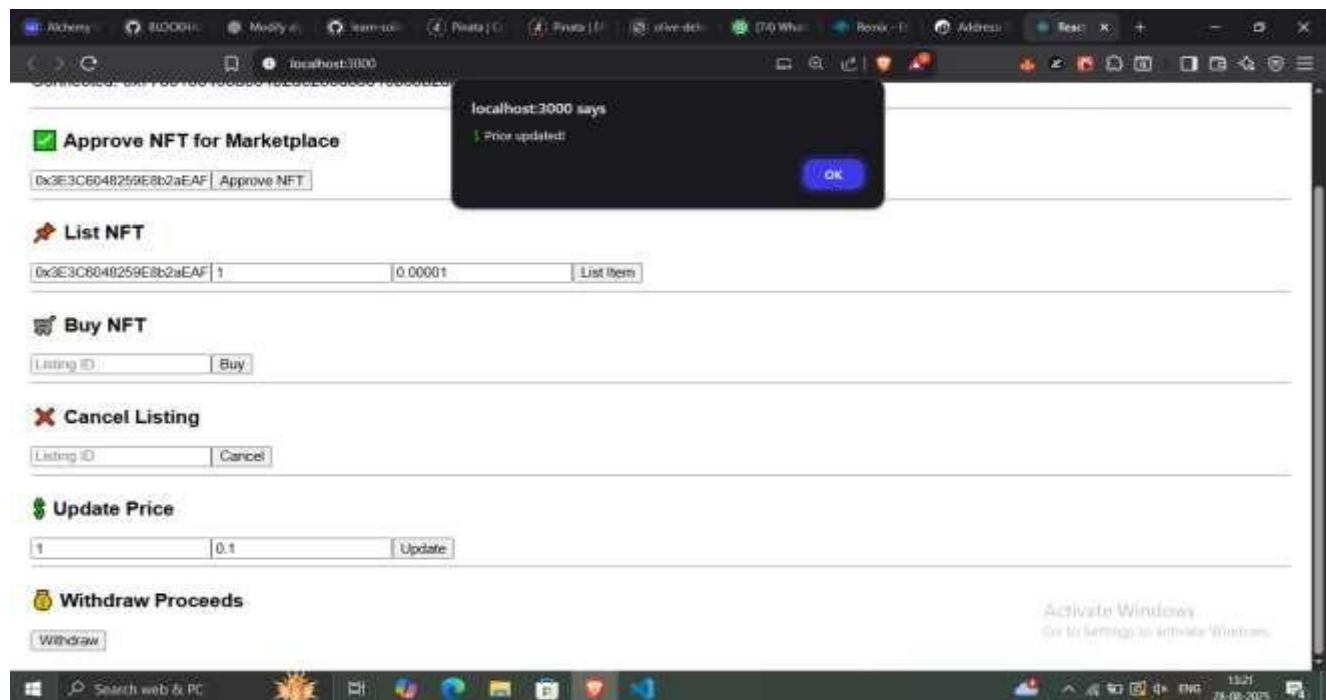
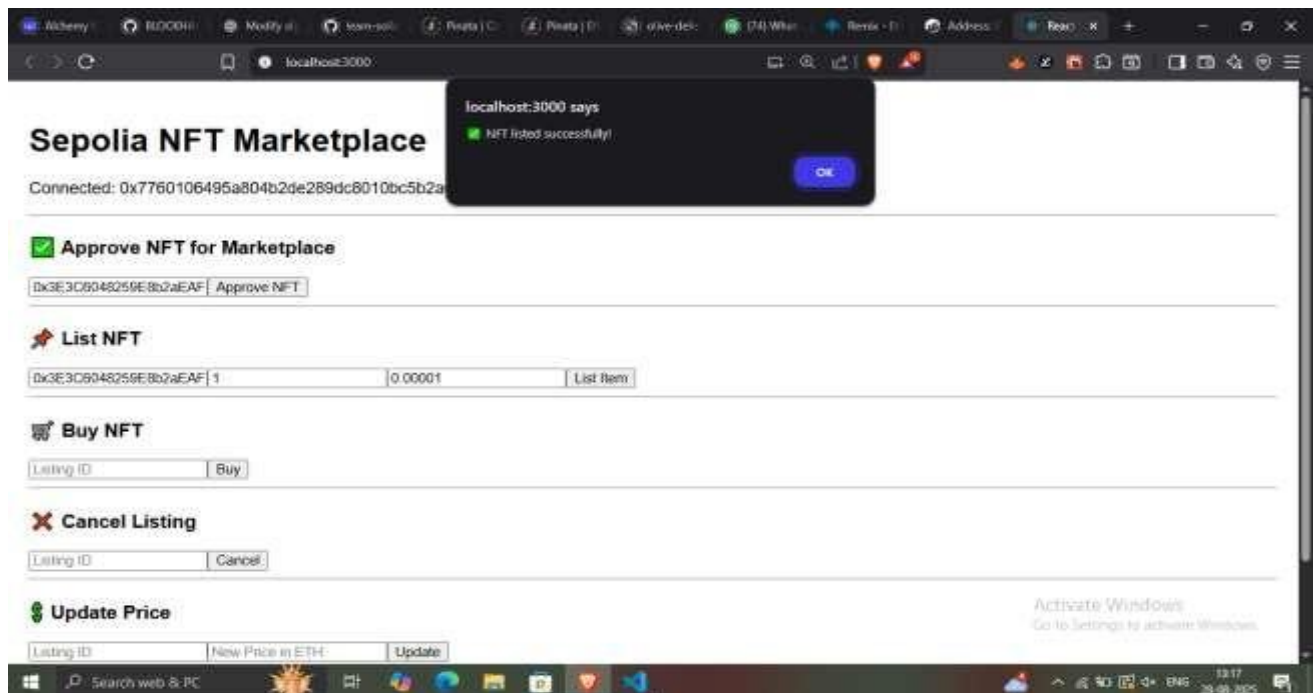
```

1  import { useState } from "react";
2  import { ethers } from "ethers";
3  import { CONTRACT_ABI, CONTRACT_ADDRESS } from "../Marketplace";
4
5  // Minimal ERC721 ABI (only approval-related functions)
6  const ERC721_ABI = [
7    "function setApprovalForAll(address operator, bool approved) external",
8    "function isApprovedForAll(address owner, address operator) view returns (bool)"
9  ];
10
11 function App() {
12   const [account, setAccount] = useState(null);
13   const [listingId, setListingId] = useState("");
14   const [nftAddress, setNftAddress] = useState("");
15   const [tokenId, setTokenId] = useState("");
16   const [price, setPrice] = useState("");
17
18   let provider;
19   if (window.ethereum) {
20     provider = new ethers.providers.Web3Provider(window.ethereum);
21     provider.on("accountsChanged", () => {
22       setAccount(provider.getSigner().getAddress());
23     });
24     provider.on("chainChanged", () => {
25       window.location.reload();
26     });
27     provider.getSigner().connect(provider).getAddress().then((address) => {
28       setAccount(address);
29     });
30   }
31 }

```



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



**\* Observations:**

- ☐ Marketplace requires **approval** before listing NFTs.
- ☐ Each NFT listing is tracked by a unique listingId.
- ☐ Buyers must pay **exact ETH** price, otherwise transaction fails.
- ☐ Smart contract prevents self-purchase by the seller.
- ☐ Sellers can cancel or update their NFT listings anytime.
- ☐ Proceeds are not sent instantly \_\_ instead they are **withdrawn securely**.
- ☐ Marketplace ensures **trustless peer-to-peer NFT trading** using blockchain logic.

**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:**