# PES UNIVERSITY

**Department of Computer Science & Engineering**


## UE21CS342BA5 - Block Chain

## Assignment-2 Report

## "Designing a Blockchain-Based Marketplace for Unique Digital Assets"

**Submitted By:**

| | |
|---|---|
| PES2UG21CS287 | Md Sami |
| PES2UG21CS266 | Mahamad Sakeeb Gadyal |
| PES2UG22CS300 | Md Jabir |

**Submitted to:**

Dr. Geetha Dayalan

Associate Professor

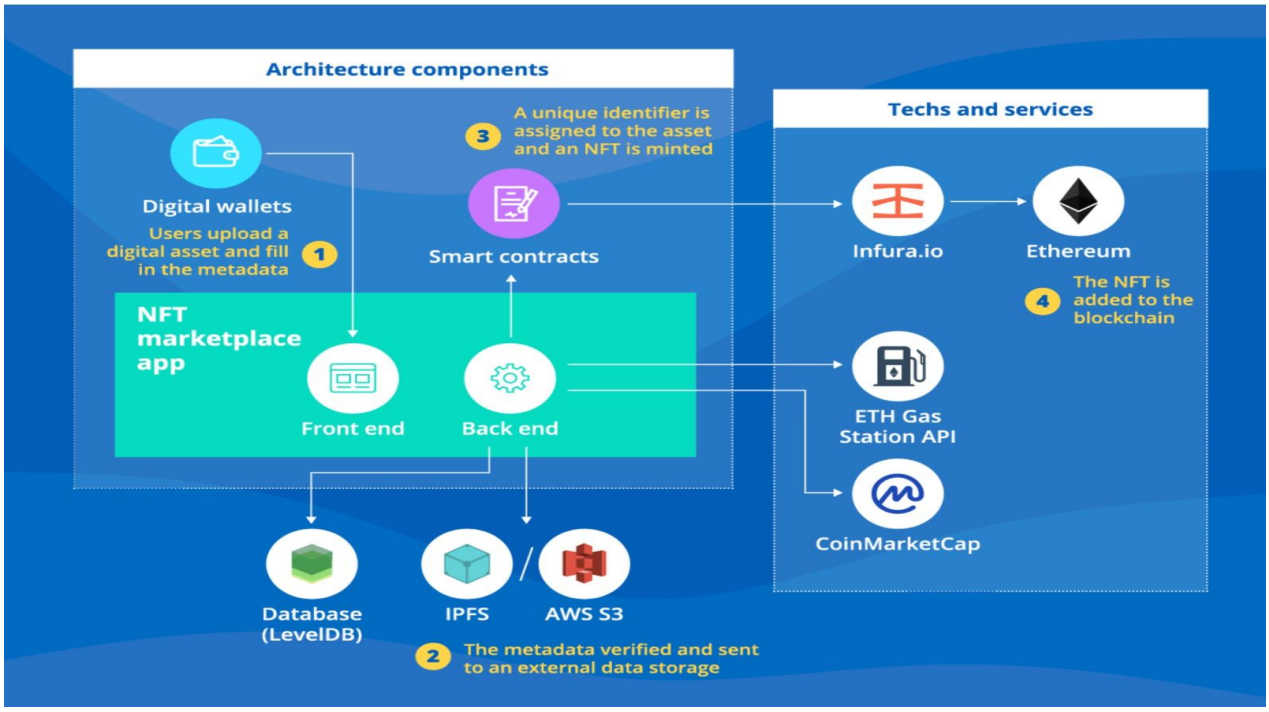
## Abstract:

The rise of blockchain technology has paved the way for innovative applications, including the creation of decentralized marketplaces for unique digital assets. In this paper, we propose the design of a blockchain-based marketplace tailored for trading unique digital assets, such as digital art, collectibles, and virtual real estate. The marketplace leverages the decentralized nature of blockchain to provide transparency, security, and immutability to transactions involving these assets.

The proposed marketplace architecture consists of several key components, including smart contracts, a decentralized ledger, a user interface, and identity management systems. Smart contracts are utilized to define the rules and conditions of asset ownership, transfer, and exchange, ensuring trustless execution of transactions. The decentralized ledger serves as a tamper-proof record of all asset transactions, providing transparency and auditability to participants.

To enhance user experience, the marketplace offers an intuitive user interface that allows users to browse, buy, sell, and transfer digital assets seamlessly. Additionally, identity management systems are implemented to verify the authenticity of asset creators and owners, mitigating the risk of fraud and counterfeit assets.

## Architecture of Problem Statement:




## Type of Blockchain to be Used:

**Public Blockchain:** Public blockchains, such as Ethereum, are decentralized networks where anyone can participate, transact, and interact with smart contracts.
**Permissioned Blockchain:** Permissioned blockchains restrict access to participants who have been granted permission by the network administrators.
**Sidechains:** Sidechains are independent blockchains that are interoperable with a main blockchain, allowing for scalability and specialization of functionalities. Sidechains can be used to offload non-critical transactions from the main blockchain, improving scalability and reducing congestion.
**Non-Fungible Token (NFT) Standards:** Non-fungible tokens represent unique digital assets that are indivisible and distinguishable from each other. Standards like ERC-721 and ERC-1155 on Ethereum provide guidelines for creating and managing NFTs, including digital art, collectibles, and virtual real estate.

## The Stakeholders Involved:

Designing a blockchain-based marketplace for unique digital assets involves various stakeholders, each playing a crucial role in the ecosystem. Here's a breakdown of key stakeholders:Developers

1. Users
2. Asset Creators
3. Investors
4. Validators/Miners
5. Marketplace Operators
6. Regulators
7. Third-Party Service Providers

## Smart Contract:

```
pragma solidity ^0.8.4;

import "@openzeppelin/contracts/utils/Counters.sol";
import "@openzeppelin/contracts/token/ERC721/extensions/ERC721URIStorage.sol";
import "@openzeppelin/contracts/token/ERC721/ERC721.sol";

import "hardhat/console.sol";

contract NFTMarketplace is ERC721URIStorage {
    using Counters for Counters.Counter;
    Counters.Counter private _tokenIds;
    Counters.Counter private _itemsSold;

    uint256 listingPrice = 0.000001 ether;
    address payable owner;

    mapping(uint256 => MarketItem) private idToMarketItem;

    struct MarketItem {
        uint256 tokenId;
        address payable seller;
        address payable owner;
        uint256 price;
        bool sold;    }
```

```
  event MarketItemCreated(
      uint256 indexed tokenId,
      address seller,
      address owner,
      uint256 price,
      bool sold
  );

  modifier onlyOwner() {
      require(
          msg.sender == owner,
          "only owner of the marketplace can change the listing price"
      );
      _;
  }

  constructor() ERC721("Metaverse Tokens", "METT") {
      owner = payable(msg.sender);
  }

  /* Updates the listing price of the contract */
  function updateListingPrice(uint256 _listingPrice)
      public
```

```
    payable
    onlyOwner
{
    require(
        owner == msg.sender,
        "Only marketplace owner can update listing price."
    );
    listingPrice = _listingPrice;
}

/* Returns the listing price of the contract */
function getListingPrice() public view returns (uint256) {
    return listingPrice;
}

/* Mints a token and lists it in the marketplace */
function createToken(string memory tokenURI, uint256 price)
    public
    payable
    returns (uint256)
{
    _tokenIds.increment();
    uint256 newTokenId = _tokenIds.current();
```

```
    _mint(msg.sender, newTokenId);
    _setTokenURI(newTokenId, tokenURI);
    createMarketItem(newTokenId, price);
    return newTokenId;
}

function createMarketItem(uint256 tokenId, uint256 price) private {
    require(price > 0, "Price must be at least 1 wei");
    require(
        msg.value == listingPrice,
        "Price must be equal to listing price"
    );

    idToMarketItem[tokenId] = MarketItem(
        tokenId,
        payable(msg.sender),
        payable(address(this)),
        price,
        false
    );

    _transfer(msg.sender, address(this), tokenId);
    emit MarketItemCreated(
```

```
            tokenId,
            msg.sender,
            address(this),
            price,
            false
        );
    }

    /* allows someone to resell a token they have purchased */
    function resellToken(uint256 tokenId, uint256 price) public payable {
        require(
            idToMarketItem[tokenId].owner == msg.sender,
            "Only item owner can perform this operation"
        );
        require(
            msg.value == listingPrice,
            "Price must be equal to listing price"
        );
        idToMarketItem[tokenId].sold = false;
        idToMarketItem[tokenId].price = price;
        idToMarketItem[tokenId].seller = payable(msg.sender);
        idToMarketItem[tokenId].owner = payable(address(this));
        _itemsSold.decrement();

        _transfer(msg.sender, address(this), tokenId);
    }

    /* Creates the sale of a marketplace item */
    /* Transfers ownership of the item, as well as funds between parties */
    function createMarketSale(uint256 tokenId) public payable {
        uint256 price = idToMarketItem[tokenId].price;
        require(
            msg.value == price,
            "Please submit the asking price in order to complete the purchase"
        );
        idToMarketItem[tokenId].owner = payable(msg.sender);
        idToMarketItem[tokenId].sold = true;
        _itemsSold.increment();
        _transfer(address(this), msg.sender, tokenId);
        payable(owner).transfer(listingPrice);
        payable(idToMarketItem[tokenId].seller).transfer(msg.value);
        idToMarketItem[tokenId].seller = payable(address(0));
    }

    /* Returns all unsold market items */
```

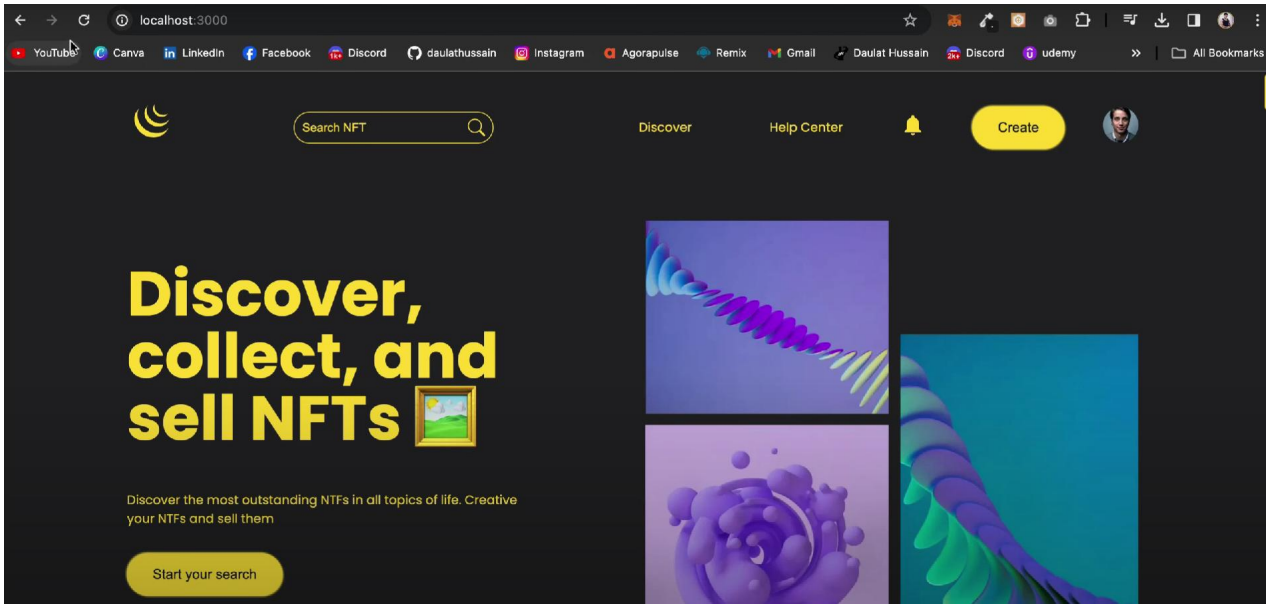
```
    function fetchMarketItems() public view returns (MarketItem[] memory) {
        uint256 itemCount = _tokenIds.current();
        uint256 unsoldItemCount = _tokenIds.current() - _itemsSold.current();
        uint256 currentIndex = 0;

        MarketItem[] memory items = new MarketItem[](unsoldItemCount);
        for (uint256 i = 0; i < itemCount; i++) {
            if (idToMarketItem[i + 1].owner == address(this)) {
                uint256 currentId = i + 1;
                MarketItem storage currentItem = idToMarketItem[currentId];
                items[currentIndex] = currentItem;
                currentIndex += 1;
            }
        }
        return items;
    }

    /* Returns only items that a user has purchased */
    function fetchMyNFTs() public view returns (MarketItem[] memory) {
        uint256 totalItemCount = _tokenIds.current();
        uint256 itemCount = 0;
        uint256 currentIndex = 0;
```

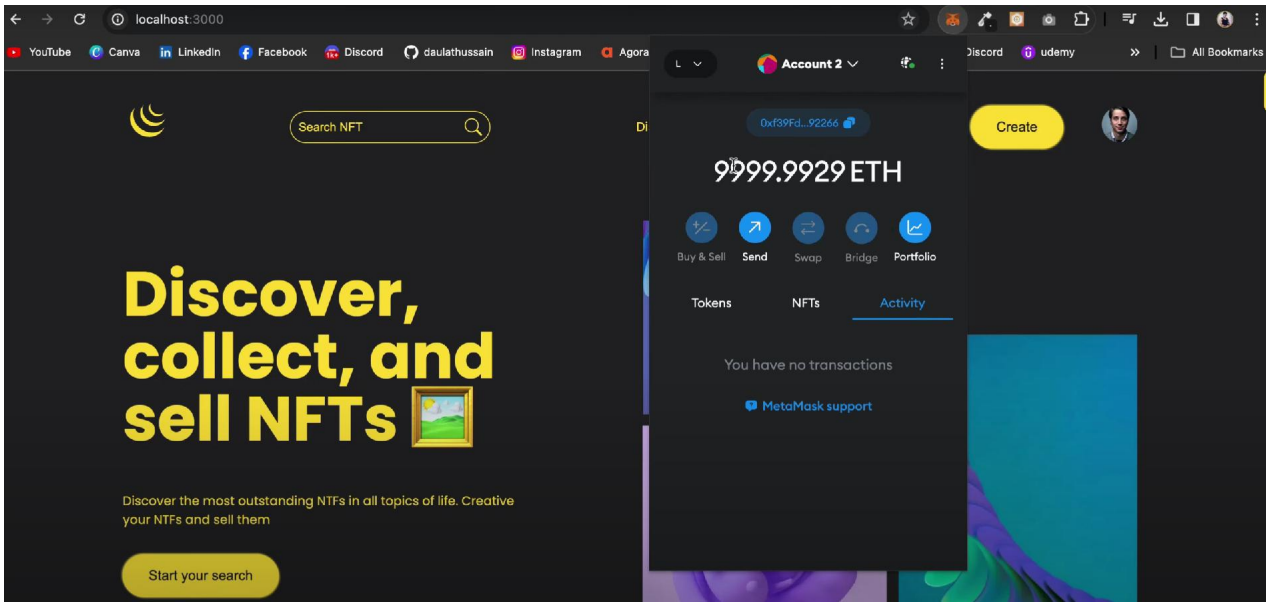
```
        for (uint256 i = 0; i < totalItemCount; i++) {
            if (idToMarketItem[i + 1].owner == msg.sender) {
                itemCount += 1;
            }
        }

        MarketItem[] memory items = new MarketItem[](itemCount);
        for (uint256 i = 0; i < totalItemCount; i++) {
            if (idToMarketItem[i + 1].owner == msg.sender) {
                uint256 currentId = i + 1;
                MarketItem storage currentItem = idToMarketItem[currentId];
                items[currentIndex] = currentItem;
                currentIndex += 1;
            }
        }
        return items;
    }

    /* Returns only items a user has listed */
    function fetchItemsListed() public view returns (MarketItem[] memory) {
        uint256 totalItemCount = _tokenIds.current();
        uint256 itemCount = 0;
        uint256 currentIndex = 0;
```

```
        for (uint256 i = 0; i < totalItemCount; i++) {
            if (idToMarketItem[i + 1].seller == msg.sender) {
                itemCount += 1;
            }
        }

        MarketItem[] memory items = new MarketItem[](itemCount);
        for (uint256 i = 0; i < totalItemCount; i++) {
            if (idToMarketItem[i + 1].seller == msg.sender) {
                uint256 currentId = i + 1;
                MarketItem storage currentItem = idToMarketItem[currentId];
                items[currentIndex] = currentItem;
                currentIndex += 1;
            }
        }
        return items;
    }
}
```

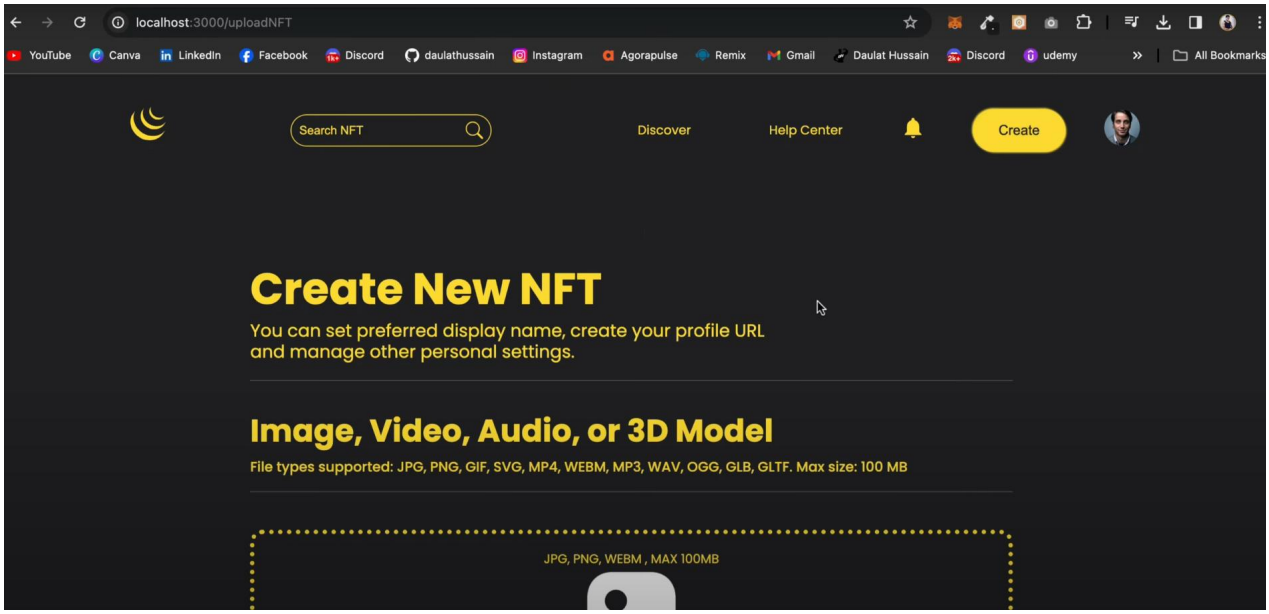
# Output screenshots:

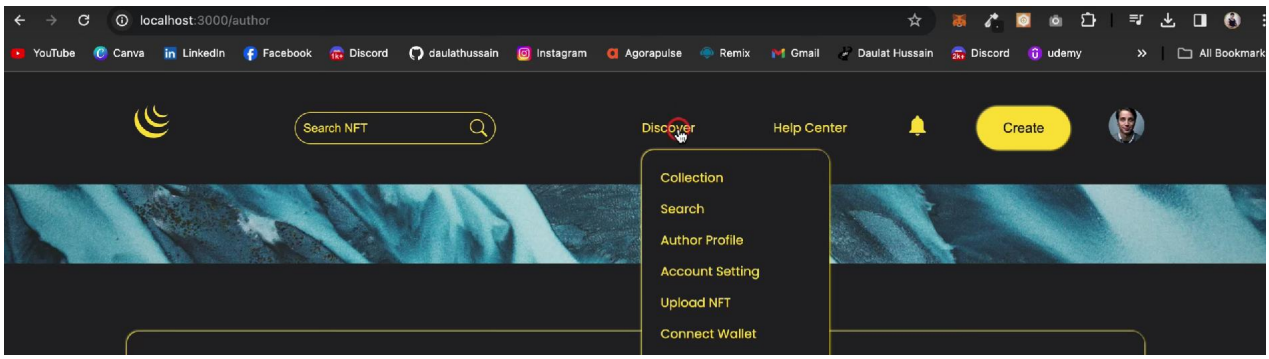
## Front page :

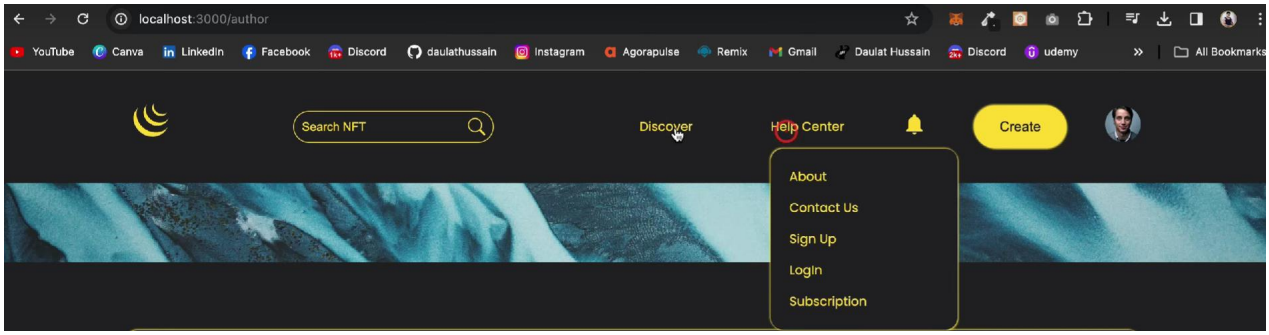



localhost:3000
Search NFT
Account 2
0xf39Fd...92266
9999.9929 ETH
Buy & Sell
Send
Swap
Bridge
Portfolio
Tokens
NFTs
Activity
You have no transactions
MetaMask support
Create
Discover, collect, and sell NFTs
Discover the most outstanding NFTs in all topics of life. Creative your NFTs and sell them
Start your search

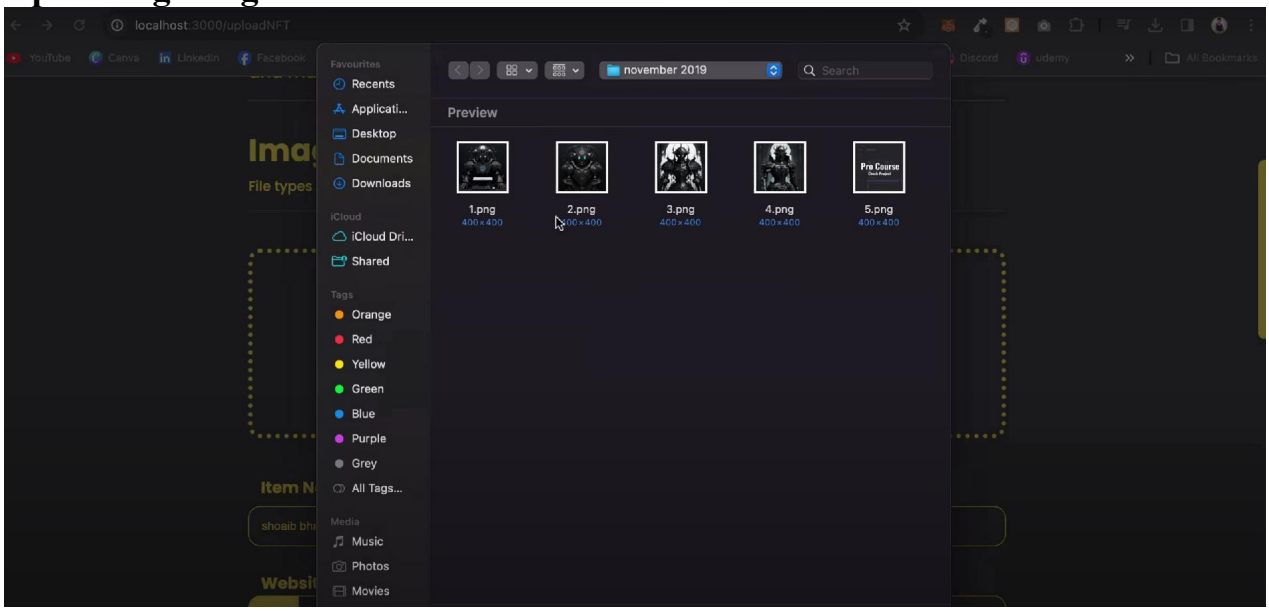
## Create:

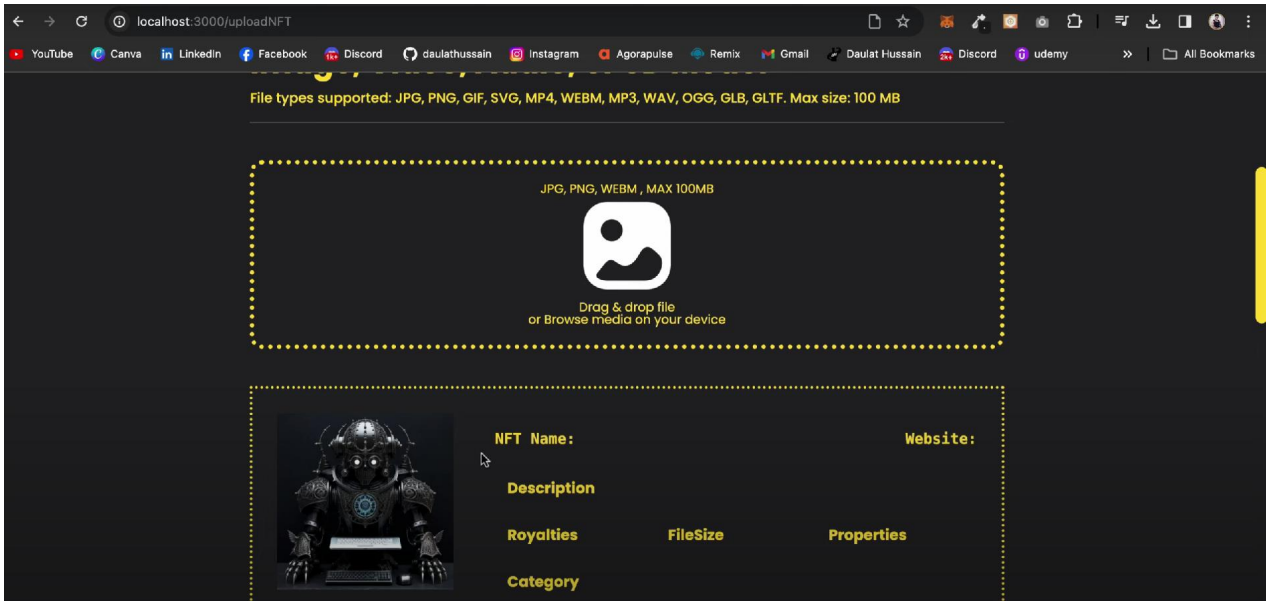

localhost:3000/uploadNFT
Search NFT
Discover
Help Center
Create
Create New NFT
You can set preferred display name, create your profile URL and manage other personal settings.
Image, Video, Audio, or 3D Model
File types supported: JPG, PNG, GIF, SVG, MP4, WEBM, MP3, WAV, OGG, GLB, GLTF. Max size: 100 MB
JPG, PNG, WEBM , MAX 100MB

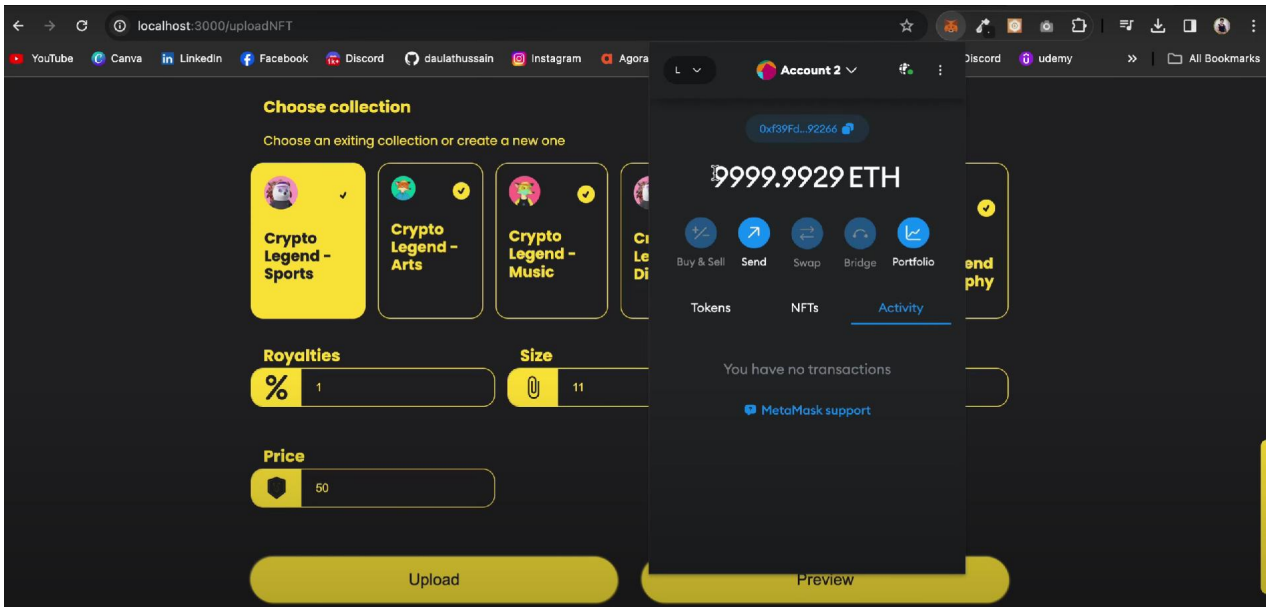

localhost:3000/author
Search NFT
Discover
Help Center
Create
Collection
Search
Author Profile
Account Setting
Upload NFT
Connect Wallet



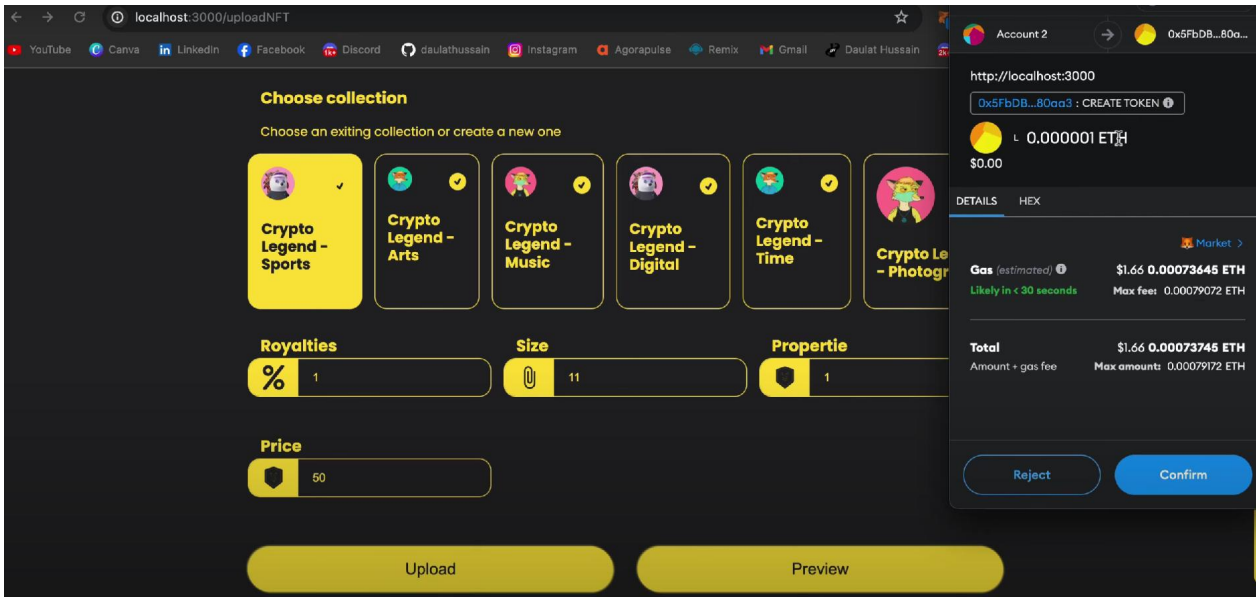

## Uploading image:



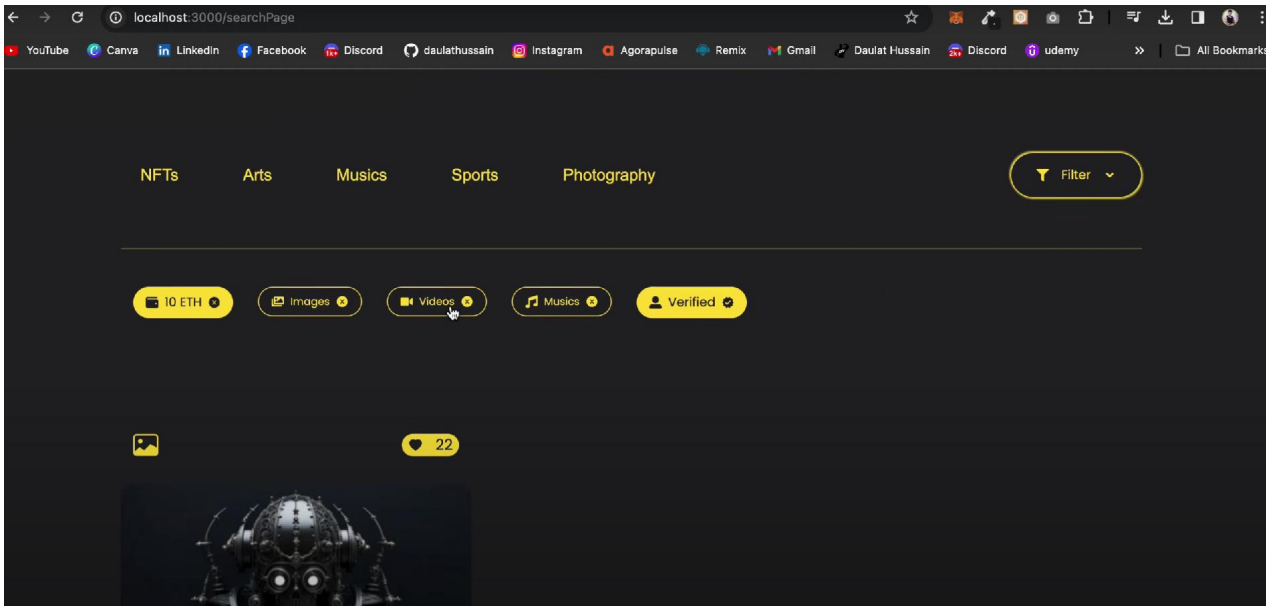





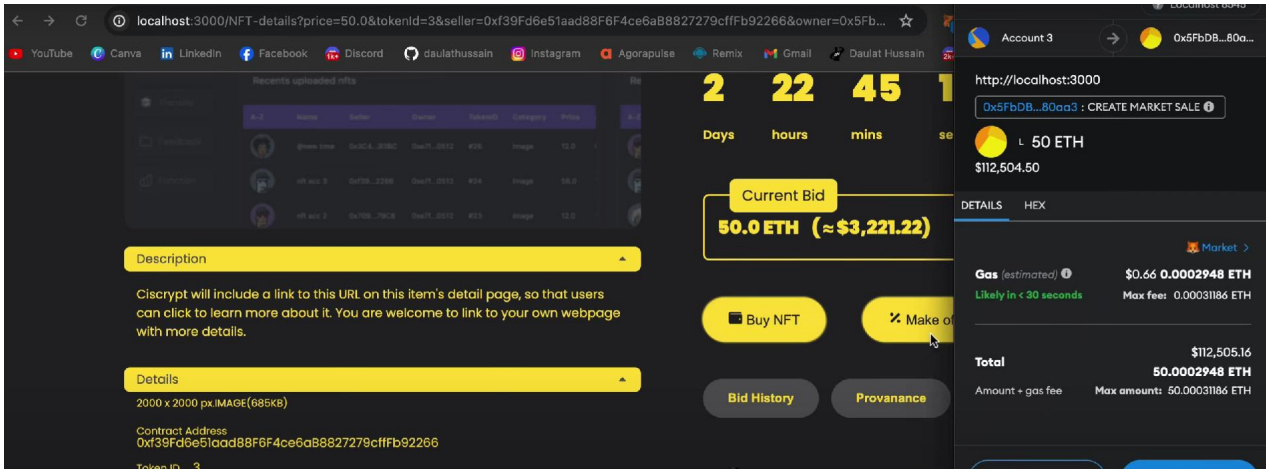

## Transaction happens here(metamask):



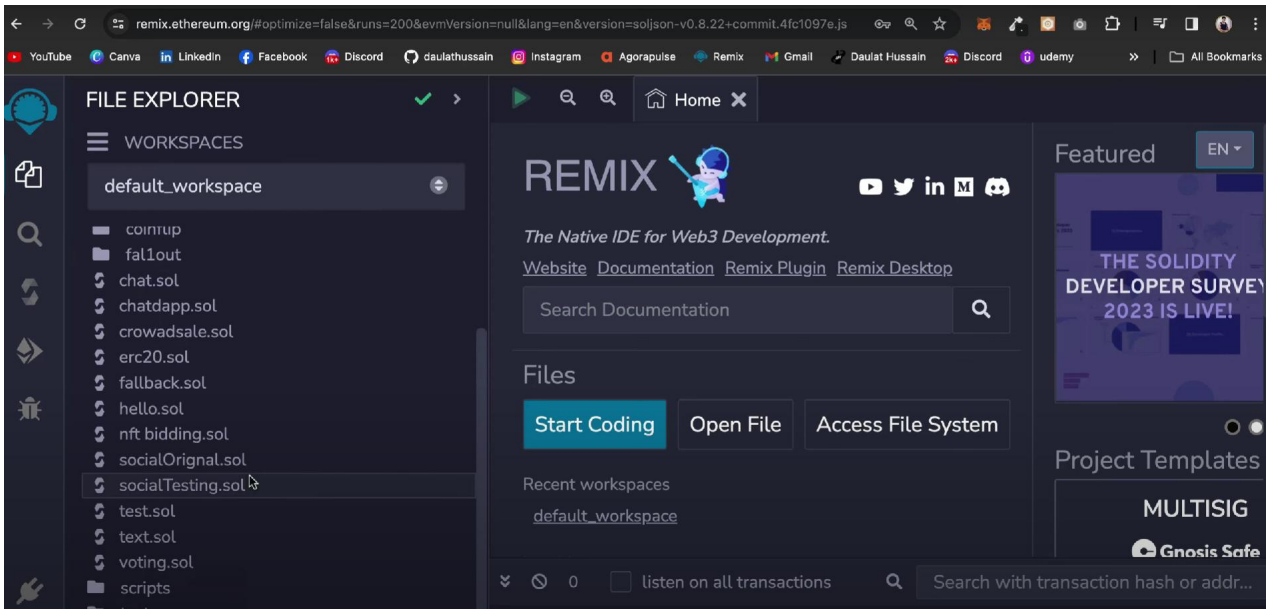
## Image Uploaded:

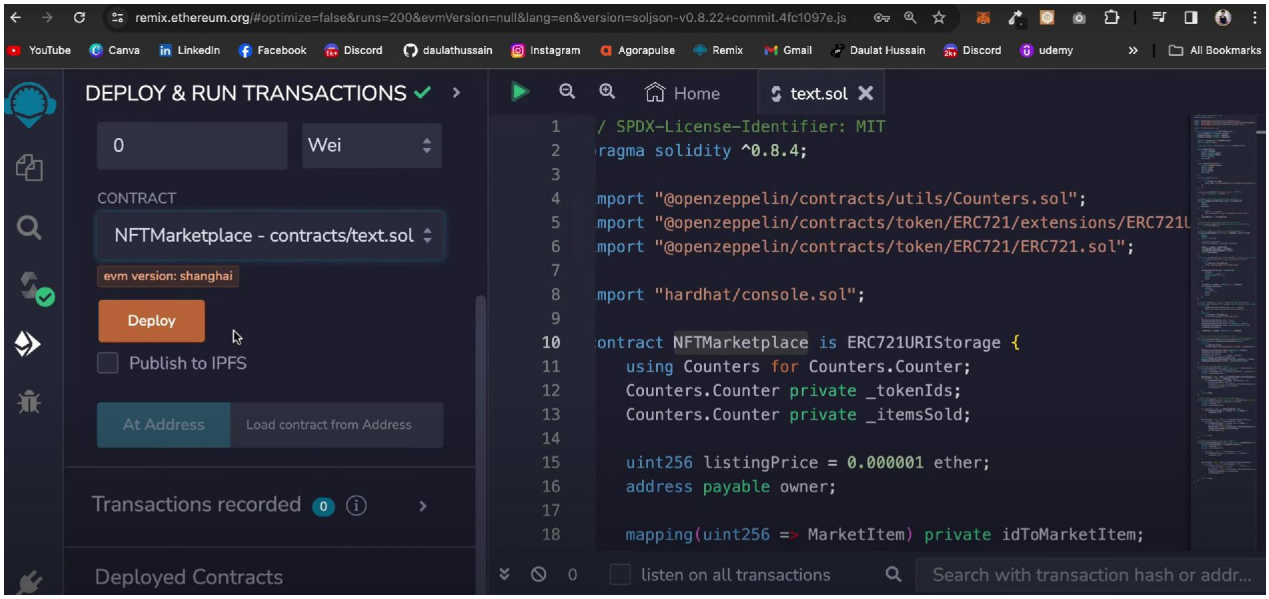



## BUY NFT:



## Remix:




**Github Repo:https://github.com/PES2UG21CS287/Block-Chain-Assignment-2-Report.git**