LAMPIRAN

Lampiran 1. Jadwal Penelitian

No.	Kegiatan	Bulan											
		Feb' 24	Mar '24	Apr '24	Mei '24	Jun '24	Jul' 24	Agu '24	Sep '24	Okt '24	Nov '24	Des '24	Jan '25
1.	Studi Literarur												
2.	Perancangan Sistem dan Aplikasi												
3.	Pengembangan Sistem dan Aplikasi												
4.	Pengujian Sistem dan Aplikasi												
5.	Penulisan Skripsi												

Lampiran 2. Kontrak MintArtwork.sol

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.4;

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

contract MintArtwork is ERC721, Ownable {
    uint public artworkCount = 0;

    struct Artwork {
        uint id;
        address signer; // artist
        string title;
        string description;
        string category;
        string file;
        string thumbnail;
        string sample; // Sample URL field
        uint price; // Price field
    }

    Artwork[] private artworks; // Array to store artworks

    event CreatedId(uint id);

    constructor() ERC721("MintedArtwork", "ART") {} // Initialize ERC721 token
with a name and symbol

    function create(
        string memory _title,
```

```
        string memory _description,
        string memory _category,
        string memory _fileUrl,
        string memory _thumbnailUrl,
        string memory _sampleUrl, // Sample URL parameter
        uint _price // Price parameter
    ) public {
        artworkCount++;

        // Store the artwork details
        artworks.push(
            Artwork(
                artworkCount,
                msg.sender,
                _title,
                _description,
                _category,
                _fileUrl,
                _thumbnailUrl,
                _sampleUrl, // Store the sample URL
                _price // Store the price
            )
        );

        // Mint an NFT with the artwork ID as the token ID
        _safeMint(msg.sender, artworkCount);

        emit CreatedId(artworkCount);
    }

    function getArtwork(uint _id) public view returns (Artwork memory) {
        return artworks[_id - 1];
    }

    function getAll() public view returns (Artwork[] memory) {
```

```
        return artworks;
    }

    function getArtworksBySigner(address _signer) public view returns
(Artwork[] memory) {
        uint count = 0;

        for (uint i = 0; i < artworkCount; i++) {
            if (artworks[i].signer == _signer) {
                count++;
            }
        }

        Artwork[] memory result = new Artwork[](count);
        uint counter = 0;

        for (uint i = 0; i < artworkCount; i++) {
            if (artworks[i].signer == _signer) {
                result[counter] = artworks[i];
                counter++;
            }
        }

        return result;
    }

    function getArtworksByCategory(string memory _category) public view
returns (Artwork[] memory) {
        uint count = 0;

        // Count the number of artworks in the specified category
        for (uint i = 0; i < artworkCount; i++) {
            if     (keccak256(abi.encodePacked(artworks[i].category))     ==
keccak256(abi.encodePacked(_category))) {
```

```
                count++;
            }
        }

        Artwork[] memory result = new Artwork[](count);
        uint counter = 0;

        // Populate the result array with artworks in the specified category
        for (uint i = 0; i < artworkCount; i++) {
            if     (keccak256(abi.encodePacked(artworks[i].category))     ==
keccak256(abi.encodePacked(_category))) {
                result[counter] = artworks[i];
                counter++;
            }
        }

        return result;
    }
```

Lampiran 3. Kontrak ArtworkTransaction.sol

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.4;

import "./MintArtwork.sol";

contract ArtworkTransaction {
    struct Transaction {
        uint id;
        address signer; // The buyer's address
        uint artworkId; // The ID of the artwork being purchased
    }
```

```
    Transaction[] private transactions;
    uint transactionCounter = 0;

    MintArtwork mintArtwork_Contract;
    address payable adminAddress =
payable(0x1383D86Bf720BE4Be028588f36D40892c3bfE279);

    // event ArtworkPurchased(uint id);
    event ArtworkPurchased(
    uint id,
    address buyer,
    uint artworkId,
    uint artworkPrice,
    uint adminFee
    );

    constructor(address _mintArtwork_Contract) {
        mintArtwork_Contract = MintArtwork(_mintArtwork_Contract);
    }

    function buyArtwork(uint _artworkId) public payable {
    require(_artworkId > 0, "Invalid artwork ID");

    // Retrieve the artwork to check its price
    MintArtwork.Artwork memory artwork =
mintArtwork_Contract.getArtwork(_artworkId);

    // Calculate admin fee
    uint256 adminFee = (artwork.price * 5) / 100; // 5% admin fee
    uint256 totalPrice = artwork.price + adminFee;

    // Check Buyer's balance
    require(msg.value >= totalPrice, "Insufficient payment");
```

```
    // Record the transaction
    transactionCounter++;
    transactions.push(Transaction(transactionCounter,            msg.sender,
_artworkId));

    // Transfer the payment to the artist
    payable(artwork.signer).transfer(artwork.price);

    // Transfer admin fee to admin
    (bool adminSuccess, ) = adminAddress.call{value: adminFee}("");
    require(adminSuccess, "Admin fee transfer failed");

    // Refund the buyer if they sent too much
    if (msg.value > totalPrice) {
        (bool   refundSuccess,   )   =   msg.sender.call{value:   msg.value   -
totalPrice}("");
        require(refundSuccess, "Refund failed");
    }

    // Emit an event for the purchase
    emit   ArtworkPurchased(transactionCounter,   msg.sender,   _artworkId,
artwork.price, adminFee);
    }

    function  getTransaction(uint256  _id)  public  view  returns  (Transaction
memory) {
        return transactions[_id - 1];
    }

    function getAllTransaction() public view returns (Transaction[] memory) {
        return transactions;
    }
```

```
    function getTransactionPerItem(uint256 _artworkId) public view returns
(Transaction[] memory) {
        uint count = 0;

        // Count the number of transactions for the given artwork ID
        for (uint i = 0; i < transactions.length; i++) {
            if (transactions[i].artworkId == _artworkId) {
                count++;
            }
        }

        // Create an array to hold the transactions for the given artwork ID
        Transaction[] memory result = new Transaction[](count);
        uint index = 0;

        for (uint i = 0; i < transactions.length; i++) {
            if (transactions[i].artworkId == _artworkId) {
                result[index] = transactions[i];
                index++;
            }
        }

        return result;
    }

    function getTransactionPerWallet(address _wallet) public view returns
(Transaction[] memory) {
    uint count = 0;

    for (uint i = 0; i < transactions.length; i++) {
        if (transactions[i].signer == _wallet) {
            count++;
        }
    }
```

```
    Transaction[] memory result = new Transaction[](count);
    uint index = 0;

    for (uint i = 0; i < transactions.length; i++) {
        if (transactions[i].signer == _wallet) {
            result[index] = transactions[i];
            index++;
        }
    }
    return result;
    }
}
```

Lampiran 4. Kode Fungsi *Shuffle*

```
function shuffleBase64(base64Data) {
    const halfLength = base64Data.length / 2;
    const modResult = Math.floor(halfLength) % 3;
    let shuffled = base64Data.split('');

    if (modResult === 0) {
        // Shuffle pattern 2
        for (let i = 0; i < shuffled.length; i += 2) {
            if (i + 1 < shuffled.length) {
                let temp = shuffled[i];
                shuffled[i] = shuffled[i + 1];
                shuffled[i + 1] = temp;
            }
        }
    } else if (modResult === 1) {
        // Shuffle pattern 1 and 0
        for (let i = 0; i < shuffled.length; i += 3) {
            if (i + 2 < shuffled.length) {
                let temp = shuffled[i];
                shuffled[i] = shuffled[i + 1];
```

```
                shuffled[i + 1] = shuffled[i + 2];
                shuffled[i + 2] = temp;
            }
        }
    } else {
        // Shuffle pattern 1
        for (let i = 0; i < shuffled.length; i += 2) {
            if (i + 1 < shuffled.length) {
                let temp = shuffled[i];
                shuffled[i] = shuffled[i + 1];
                shuffled[i + 1] = temp;
            }
        }
    }
    return shuffled.join('');
}
```

Lampiran 5. Kode Fungsi *Deshuffle*

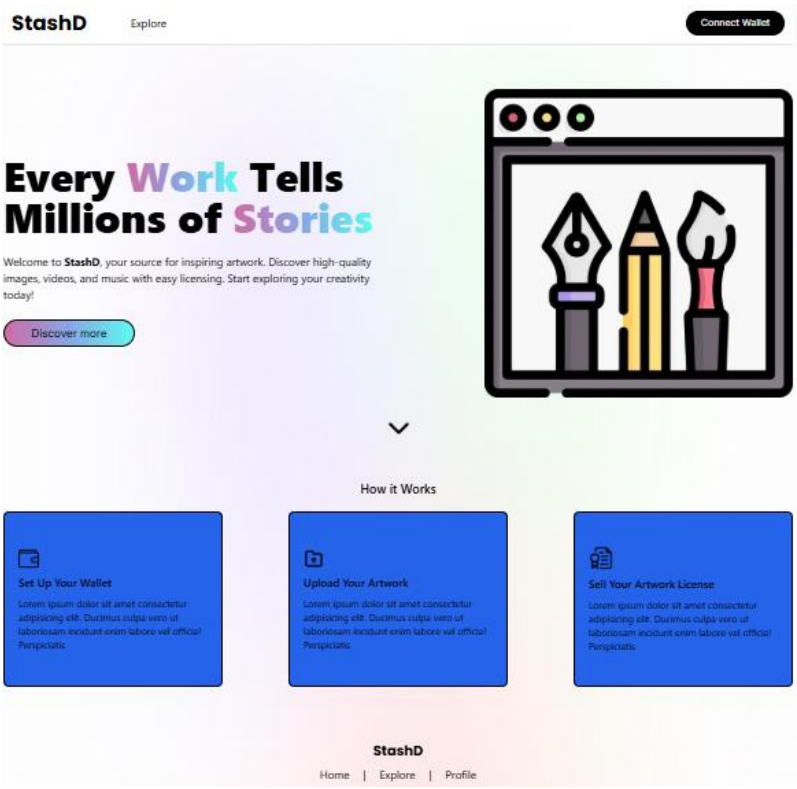
```
function deshuffleBase64(shuffledData) {
        const halfLength = shuffledData.length / 2;
        const modResult = Math.floor(halfLength) % 3;
        let deshuffled = shuffledData.split('');

        if (modResult === 0) {
            // Reverse pattern 2
            for (let i = 0; i < deshuffled.length; i += 2) {
                if (i + 1 < deshuffled.length) {
                    let temp = deshuffled[i];
                    deshuffled[i] = deshuffled[i + 1];
                    deshuffled[i + 1] = temp;
                }
            }
        } else if (modResult === 1) {
```

```
            // Reverse pattern 1 and 0
            for (let i = deshuffled.length - 3; i >= 0; i -= 3) {
                if (i + 2 < deshuffled.length) {
                    let temp = deshuffled[i + 2];
                    deshuffled[i + 2] = deshuffled[i + 1];
                    deshuffled[i + 1] = deshuffled[i];
                    deshuffled[i] = temp;
                }
            }
        } else {
            // Reverse pattern 1
            for (let i = 0; i < deshuffled.length; i += 2) {
                if (i + 1 < deshuffled.length) {
                    let temp = deshuffled[i];
                    deshuffled[i] = deshuffled[i + 1];
                    deshuffled[i + 1] = temp;
                }
            }
        }
        return deshuffled.join('');
    }
```

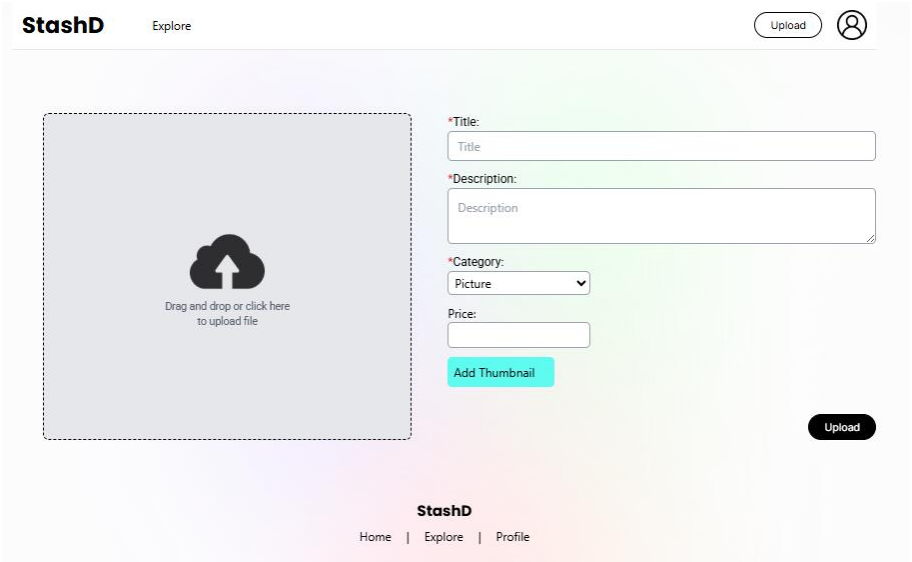
Lampiran 6. *Source Code* Aplikasi

https://github.com/MNRCHY/StashD

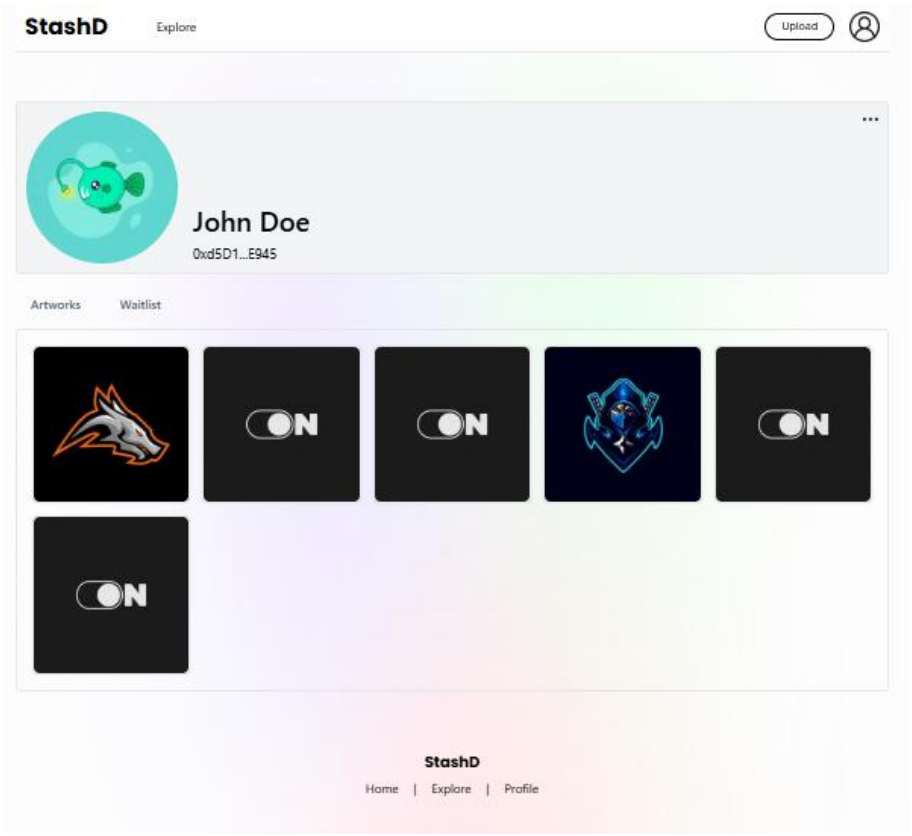
Lampiran 7. Tampilan Halaman Utama Aplikasi



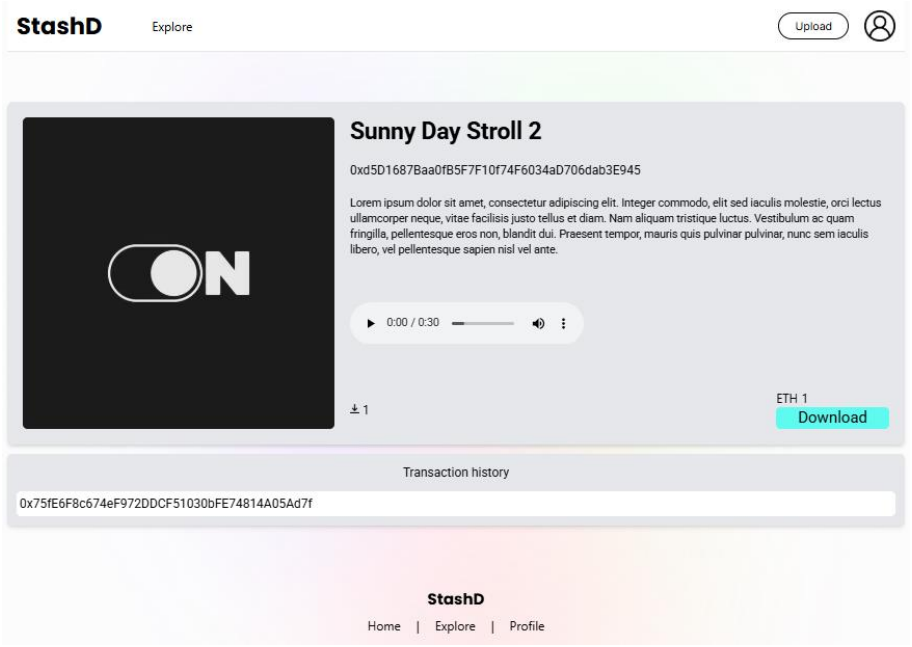
Lampiran 8. Tampilan Halaman *Upload*



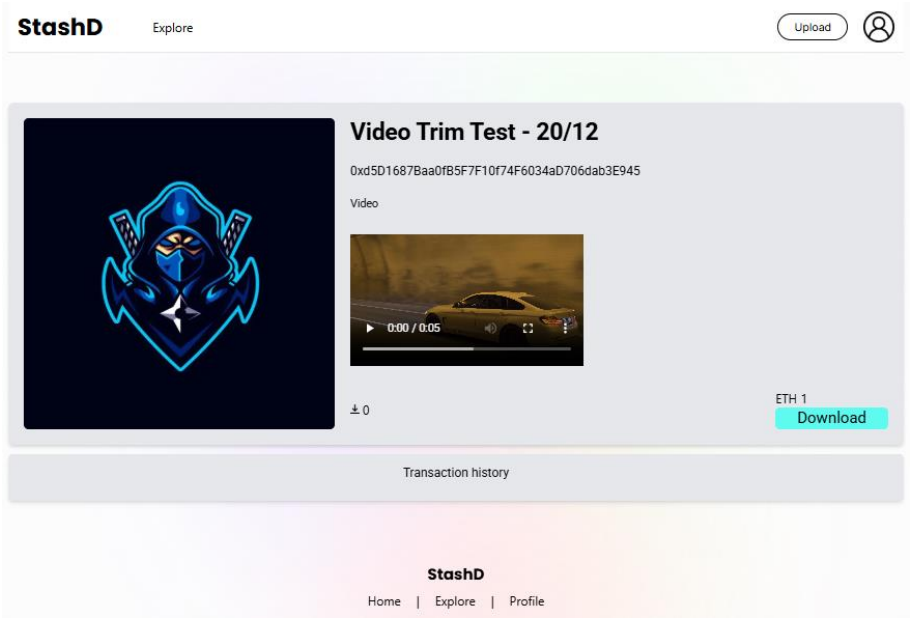
Lampiran 9. Tampilan Halaman *Profile*



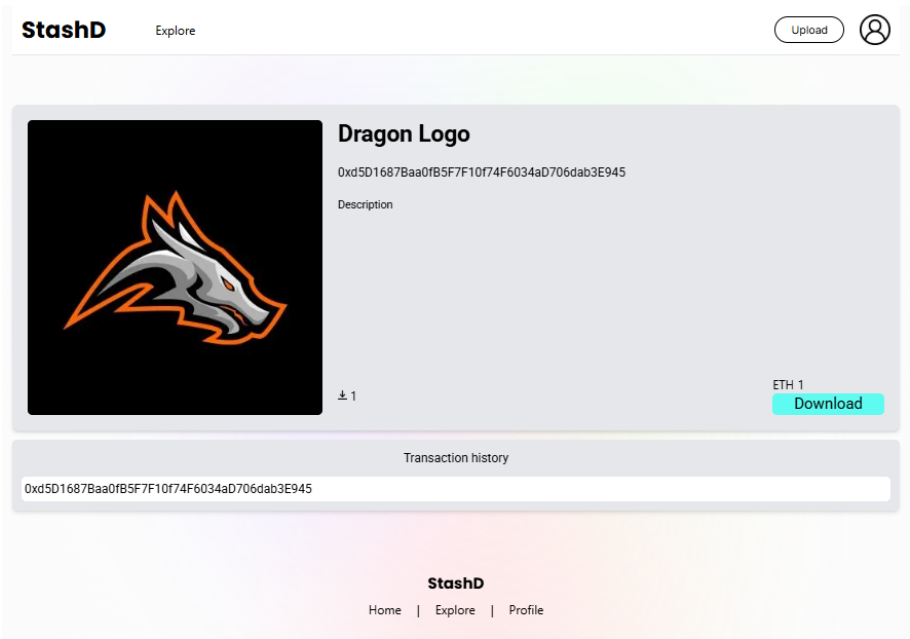
Lampiran 10. Tampilan Halaman Detail Karya Audio



Lampiran 11. Tampilan Halaman Detail Karya Video



Lampiran 12. Tampilan Halaman Detail Karya Gambar



Lampiran 13. Tampilan Halaman Dashboard Admin