

Blockchain for Artist Royalties

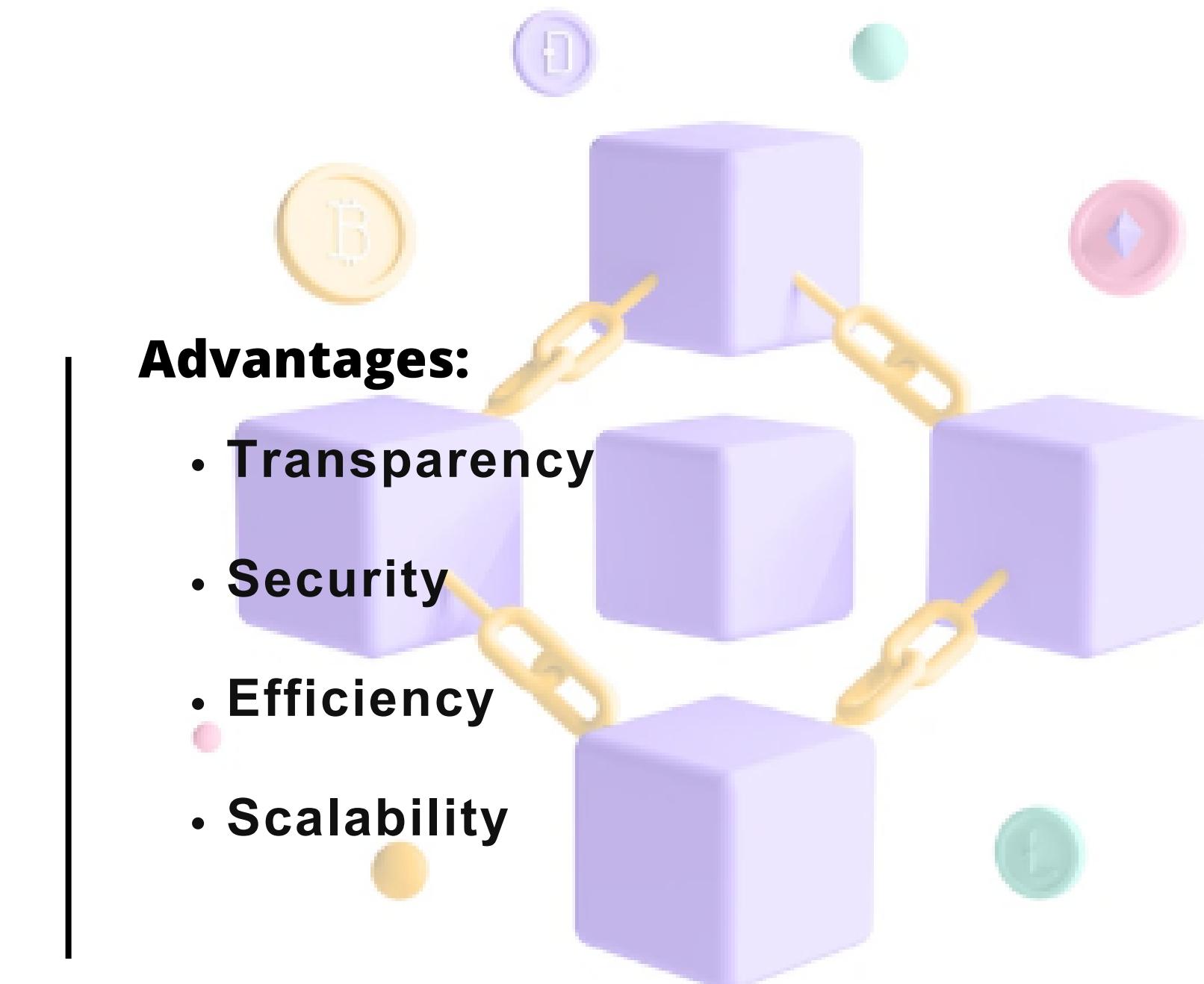
Presented by Aditi Salvi D17B / 62

What is Blockchain?

A blockchain is a distributed database that maintains a continuously growing list of ordered records called blocks. Each block contains a timestamp and a link to a previous block. This allows the blockchain to be used to record transactions securely and transparently without the need for a central authority.

Advantages:

- Transparency
- Security
- Efficiency
- Scalability



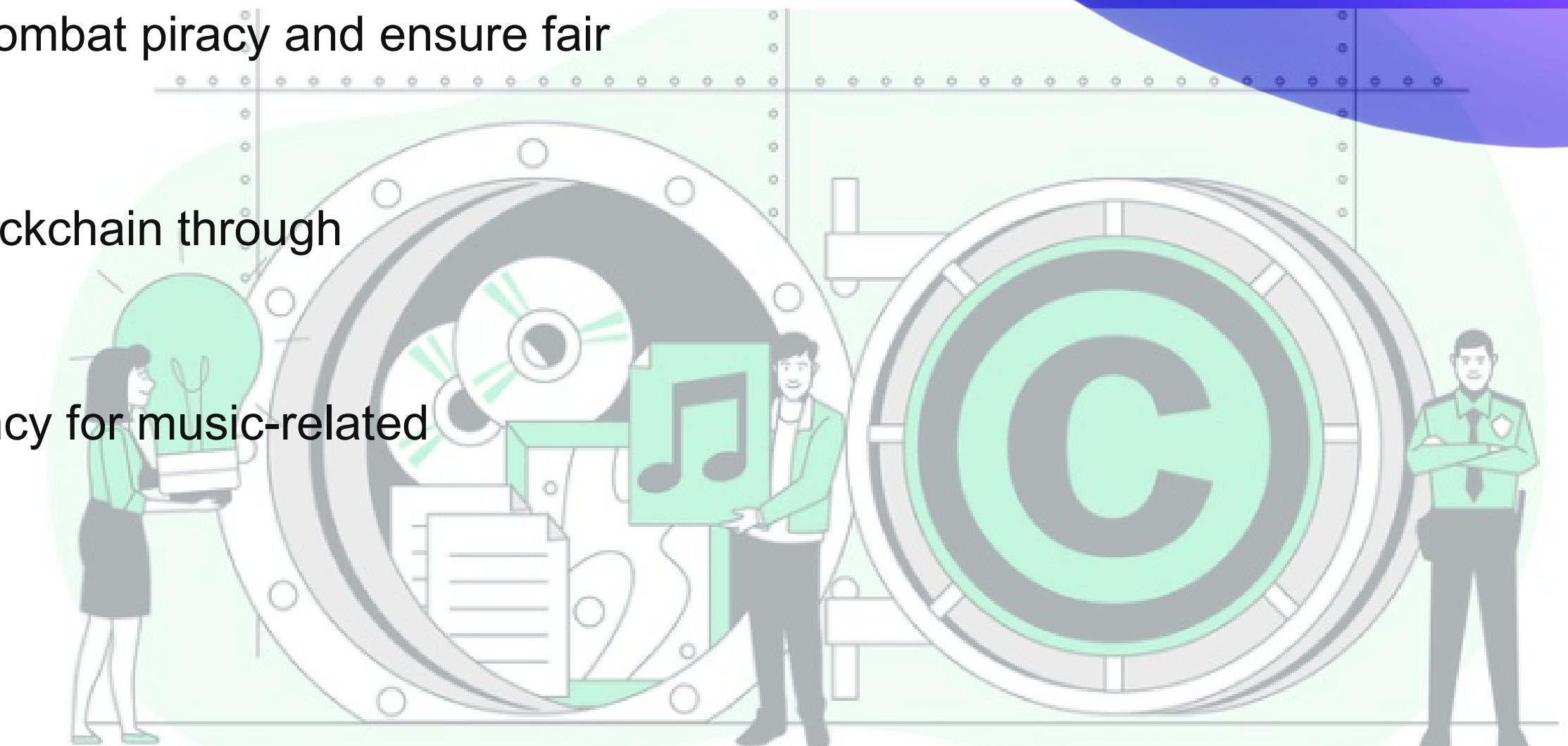
Artist Royalties:

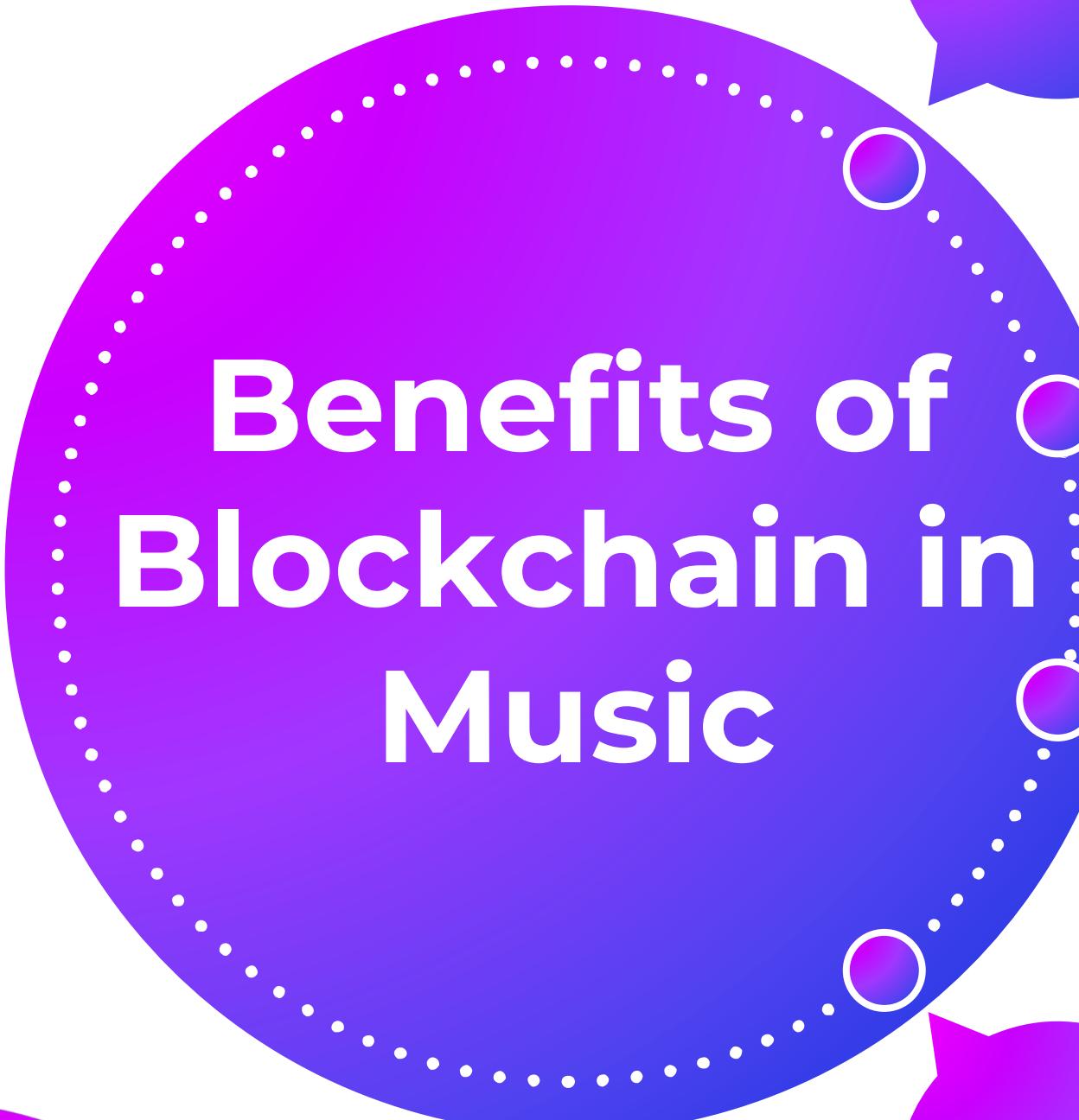


- Artist royalties are like payments that musicians and artists get when their music is used or sold.
- It's a way for artists to earn money from their hard work and creativity.
- **Key issues in the existing system include:**
 - Lack of Authentic Copyright Database
 - Delayed and Unfair Payments
 - Lack of Data Transparency

Blockchain in Music

- Technology impact on music industry: Need for modernization in royalty management and rights tracking.
- Challenges in the music industry include music piracy and complex ownership laws.
- Blockchain's distributed ledger can combat piracy and ensure fair royalty payments.
- Emerging artists can benefit from blockchain through crowdfunding and prompt payments.
- Blockchain can support global currency for music-related transactions and offers high security.

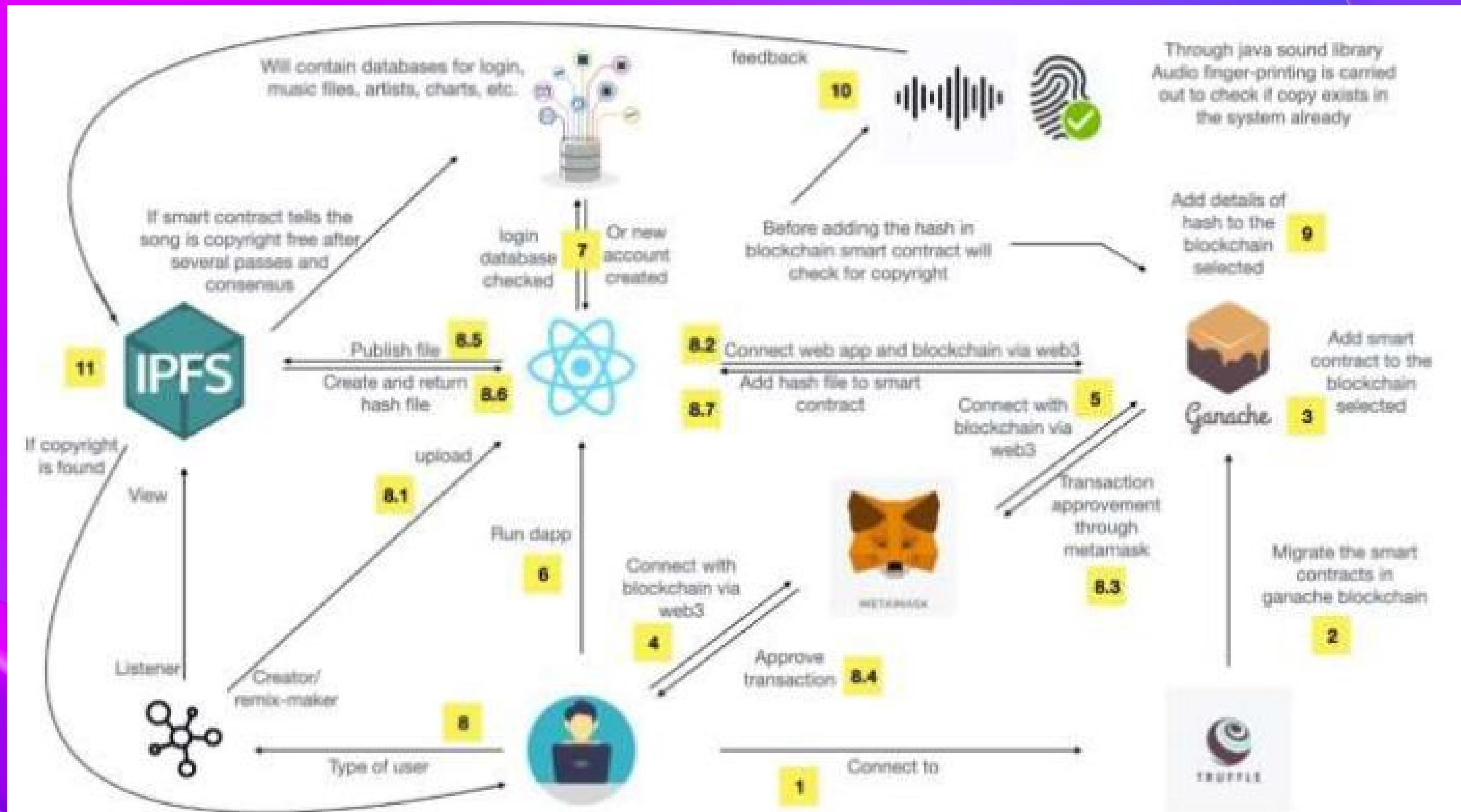




Benefits of Blockchain in Music

- 01 Transparency
- 02 Reduced
Intermediaries
- 03 Smart Contracts
- 04 Fair Compensation for Remixes and
Collaborations

Overview of the proposed System



Phase 1: Hash Generation

- Generated hash for copyright protection
- Performed various other tasks

Phase 2: Audio Fingerprinting

- focuses on audio fingerprinting for songs
- fingerprints, along with song metadata, are stored in a database to detect and prevent unauthorized uploads

Phase 3: Wallet Transaction

- It covers transactions like royalty transfers for remix makers and stream rate-dependent transfers.

Conclusion

- In conclusion, blockchain technology has the potential to revolutionize the way artist royalties are managed and distributed in the music industry, offering a range of benefits
- Overall, blockchain enhances transparency, fairness, security, and efficiency in artist royalty management, ultimately benefiting artists and stakeholders in the music industry.

Reference

Blockchain Based Model for Royalty Payments of Artists and Remix-Makers by Ms. Shreya Bilonikara, Ms. Carol Mendoncab, Ms. Divita Phadakalec, Prof. Monali Shettyd

https://drive.google.com/file/d/1VKGNckG6k8dLkDaOeYImqv_tn_iGpPRV/view?usp=sharing

EXPERIMENT 6: Smart Contract 1:

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract ArtistRegistrationContract {
    struct Artist {
        address artistAddress;
        string artistName;
        uint256 royaltyPercentage;
    }
    mapping(address => Artist) public artists;
    event ArtistRegistered(address indexed artistAddress, string artistName, uint256 royaltyPercentage);
    function registerArtist(string memory _artistName, uint256 _royaltyPercentage) public {
        require(_royaltyPercentage >= 0 && _royaltyPercentage <= 100, "Invalid royalty percentage");
        require(artists[msg.sender].artistAddress == address(0), "Artist is already registered");
        artists[msg.sender] = Artist({
            artistAddress: msg.sender,
            artistName: _artistName,
            royaltyPercentage: _royaltyPercentage
        });
        emit ArtistRegistered(msg.sender, _artistName, _royaltyPercentage);
    }
}
```

EXPERIMENT 6:

Smart Contract 1:

The screenshot shows a development interface for a Ethereum-based smart contract. The interface includes a log of transactions, a sidebar for deployed contracts, and a form for interacting with the contract.

Log:

- [vm] from: 0x5B3...eddC4 to: ArtistRegistrationContract.(constructor) value: 0 wei data: 0x608...20033 logs: 0 hash: 0x181...34f33 creation of ArtistRegistrationContract pending...
- [vm] from: 0x5B3...eddC4 to: ArtistRegistrationContract.(constructor) value: 0 wei data: 0x608...20033 logs: 0 hash: 0x208...bd838 transact to ArtistRegistrationContract.registerArtist pending ...
- [vm] from: 0x5B3...eddC4 to: ArtistRegistrationContract.registerArtist(string,uint256) 0x93f...C96CC value: 0 wei data: 0x68e...08080 logs: 1 hash: 0xF9F...6182c call to ArtistRegistrationContract.artists
- [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: ArtistRegistrationContract.artists(address) data: 0x969...eedc4

Deployed Contracts:

- ARTISTREGISTRATIONCONTRACT AT (0x5B3...eddC4) X

Balance: 0 ETH

registerArtist:

artistArtistName: Artist62

_royaltyPercentage: 50

Calldata Parameters transact

artists 0x5B38Da6a701c568545dCfc

0: address: artistAddress 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
1: string: artistName Artist62
2: uint256: royaltyPercentage 50

EXPERIMENT 6: Smart Contract 2

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract ArtistRoyaltyContract {
    address public artist;
    uint256 public royaltyPercentage;
    event ArtworkSold(address indexed buyer, uint256 price, uint256 royalties);
    constructor(uint256 _royaltyPercentage) {
        artist = msg.sender;
        royaltyPercentage = _royaltyPercentage;
    }
    function setRoyaltyPercentage(uint256 _newRoyaltyPercentage) public {
        require(msg.sender == artist, "Only the artist can set the royalty percentage");
        require(_newRoyaltyPercentage >= 0 && _newRoyaltyPercentage <= 100, "Invalid royalty percentage");
        royaltyPercentage = _newRoyaltyPercentage;
    }
    function sellArtwork() public payable {
        uint256 artistRoyalty = (msg.value * royaltyPercentage) / 100;
        (bool success, ) = artist.call{value: artistRoyalty}("");
        require(success, "Failed to send royalties to the artist");
        emit ArtworkSold(msg.sender, msg.value, artistRoyalty);
    }
}
```

EXPERIMENT 6:

Smart Contract 2:

```
[vm] from: 0x583...eddC4 to: ArtistRoyaltyContract.(constructor) value: 0 wei data: 0x608...80032 logs: 0
hash: 0x847...c4994
call to ArtistRoyaltyContract.artist

[call] from: 0x5838Da6a701c568545dCfcB803FcB8875f56beddC4 to: ArtistRoyaltyContract.artist() data: 0x43b...c1612
call to ArtistRoyaltyContract.royaltyPercentage

[call] from: 0x5838Da6a701c568545dCfcB803FcB8875f56beddC4 to: ArtistRoyaltyContract.royaltyPercentage()
data: 0x8a7...1bb2d
transact to ArtistRoyaltyContract.sellArtwork pending ...

[vm] from: 0x583...eddC4 to: ArtistRoyaltyContract.sellArtwork() 0xEf9...10e8f value: 0 wei data: 0x863...45af1
logs: 1 hash: 0x7aa...6b91b
transact to ArtistRoyaltyContract.setRoyaltyPercentage pending ...

[vm] from: 0x583...eddC4 to: ArtistRoyaltyContract.setRoyaltyPercentage(uint256) 0xEf9...10e8f value: 0 wei
data: 0x61b...8003c logs: 0 hash: 0xc84...4e722
call to ArtistRoyaltyContract.royaltyPercentage

[call] from: 0x5838Da6a701c568545dCfcB803FcB8875f56beddC4 to: ArtistRoyaltyContract.royaltyPercentage()
data: 0x8a7...1bb2d
```

Deployed Contracts

- ARTISTROYALTYCONTRACT AT 0xEf9.
Balance: 0 ETH

sellArtwork

setRoyaltyPerc... 60

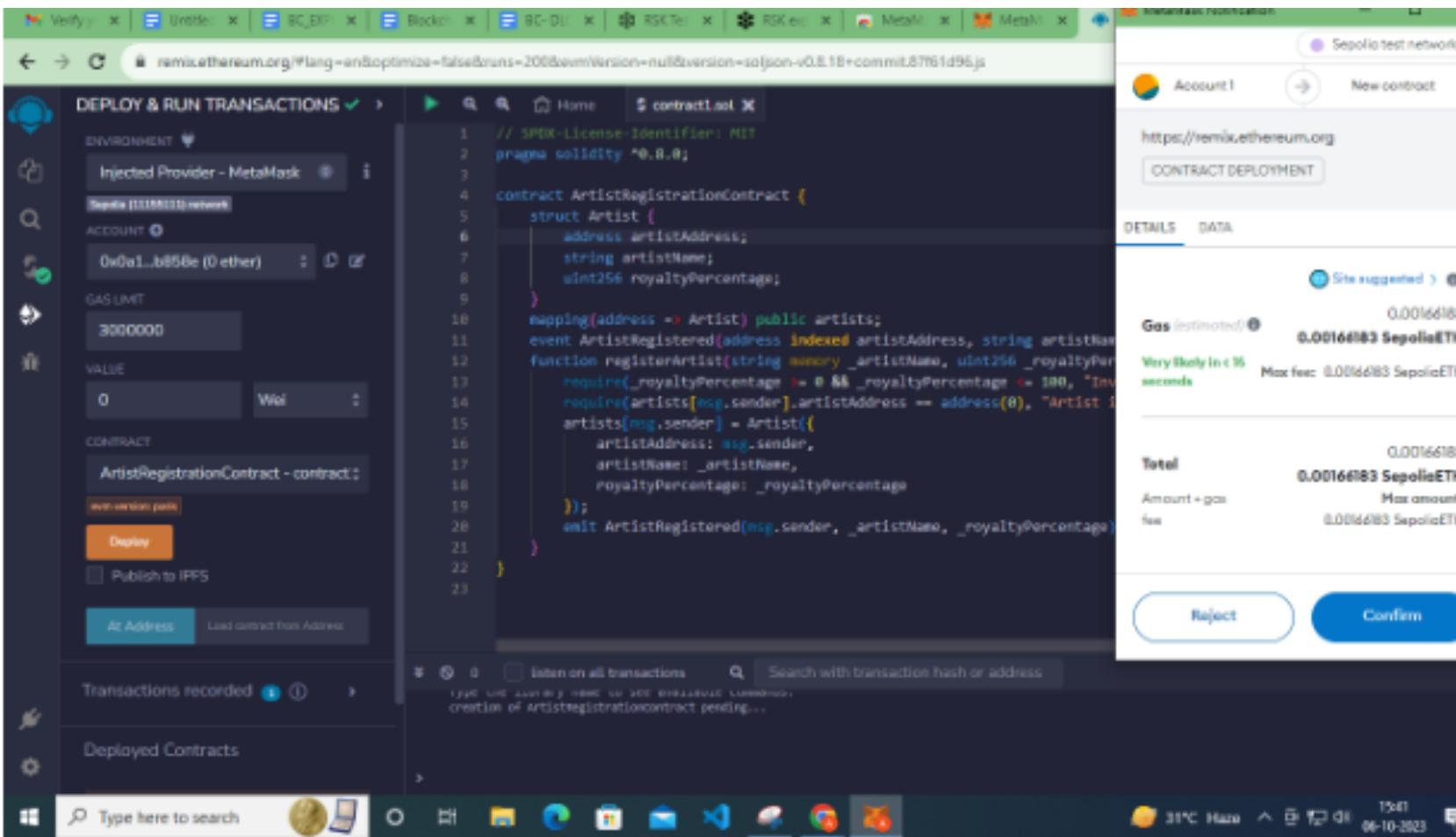
artist

0: address: 0x5838Da6a701c568545dCfcB03FcB8875f56beddC4

royaltyPercent...

0: uint256: 60

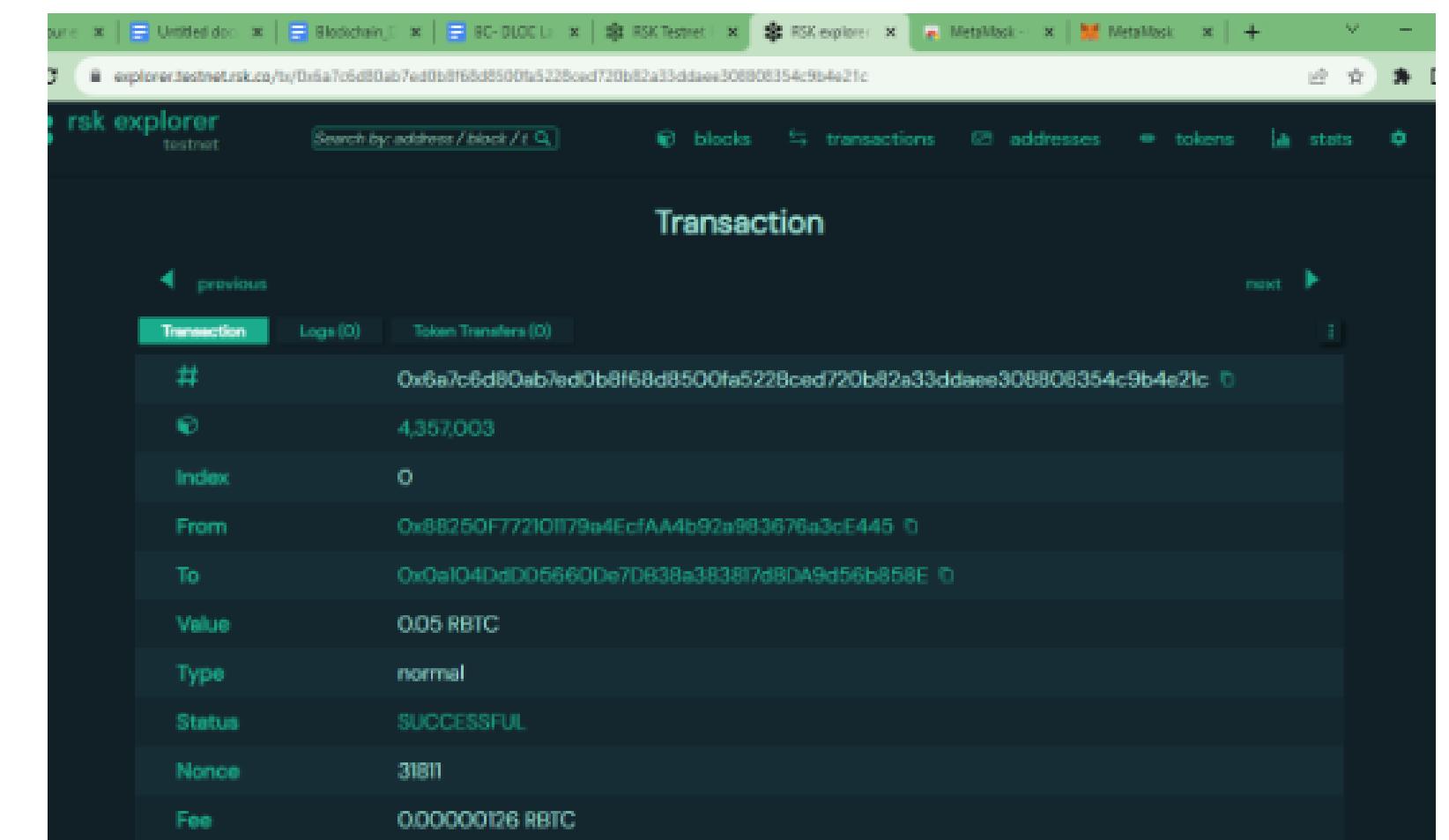
EXPERIMENT 7:



```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract ArtistRegistrationContract {
    struct Artist {
        address artistAddress;
        string artistName;
        uint256 royaltyPercentage;
    }
    mapping(address => Artist) public artists;
    event ArtistRegistered(address indexed artistAddress, string artistName, uint256 royaltyPercentage);

    function registerArtist(string memory _artistName, uint256 _royaltyPercentage) external {
        require(_royaltyPercentage >= 0 && _royaltyPercentage <= 100, "Invalid royalty percentage");
        require(artists[msg.sender].artistAddress == address(0), "Artist already registered");
        artists[msg.sender] = Artist({
            artistAddress: msg.sender,
            artistName: _artistName,
            royaltyPercentage: _royaltyPercentage
        });
        emit ArtistRegistered(msg.sender, _artistName, _royaltyPercentage);
    }
}
```



Transaction	Logs (0)	Token Transfers (0)
# 0x6a7c6d80ab7ed0b8f68d8500fa5228ced720b82a33ddae308808354c9b4e21c	4,357,003	
Index 0	From 0x88250f772101179a4Ec1AA4b92a983676a3cE445	To 0x0a104DdD05660De70838a383817d80A9d56b858E
Value 0.05 RBTC	Type normal	Status SUCCESSFUL
Nonce 31811	Fee 0.000000126 RBTC	

EXPERIMENT 8:

