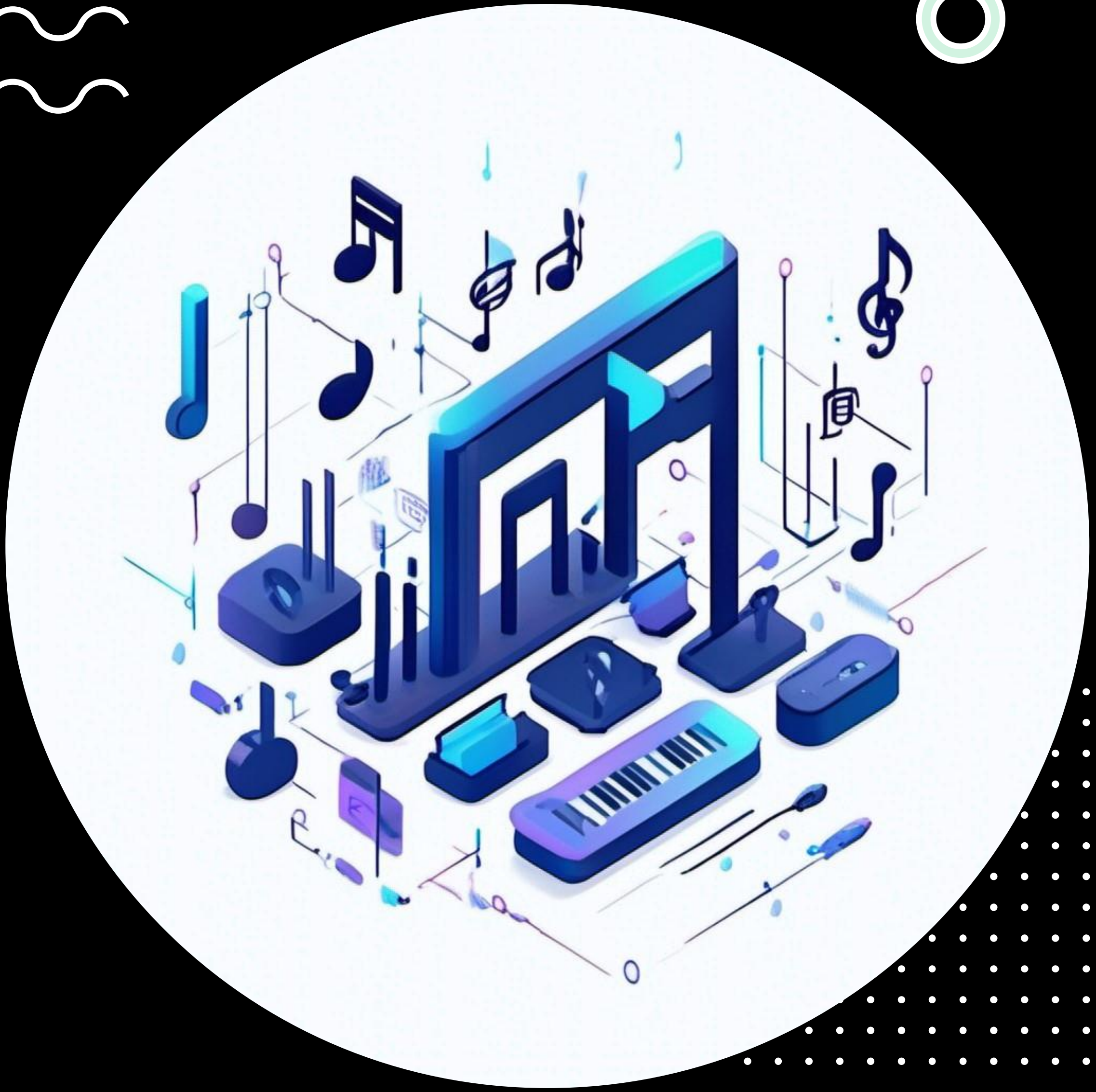
OPENCHAIN

A Solidity smart contract for creating and selling digital content with buyer tracking and availability management

JAYASHRE K

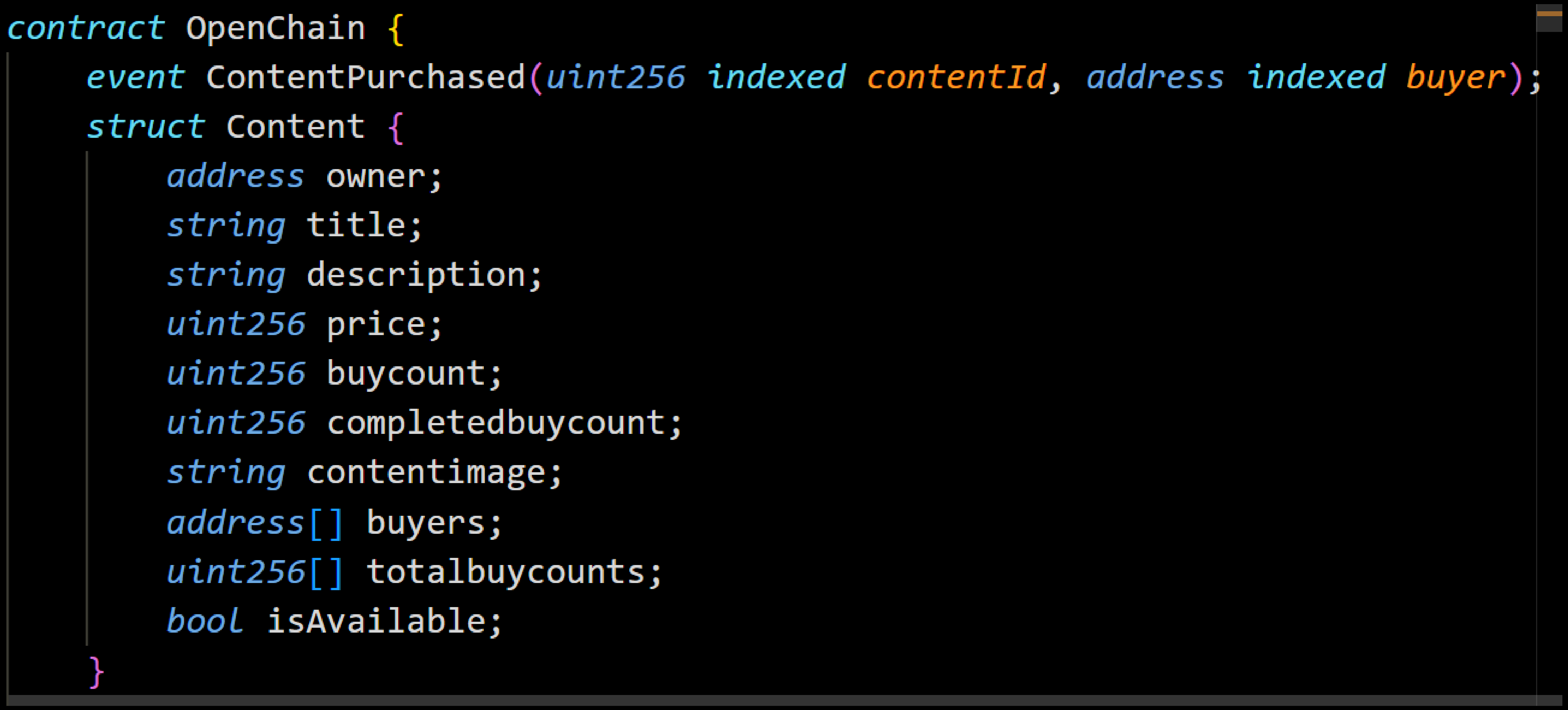
22011103020



FUNCTIONS OF OPENCHAIN

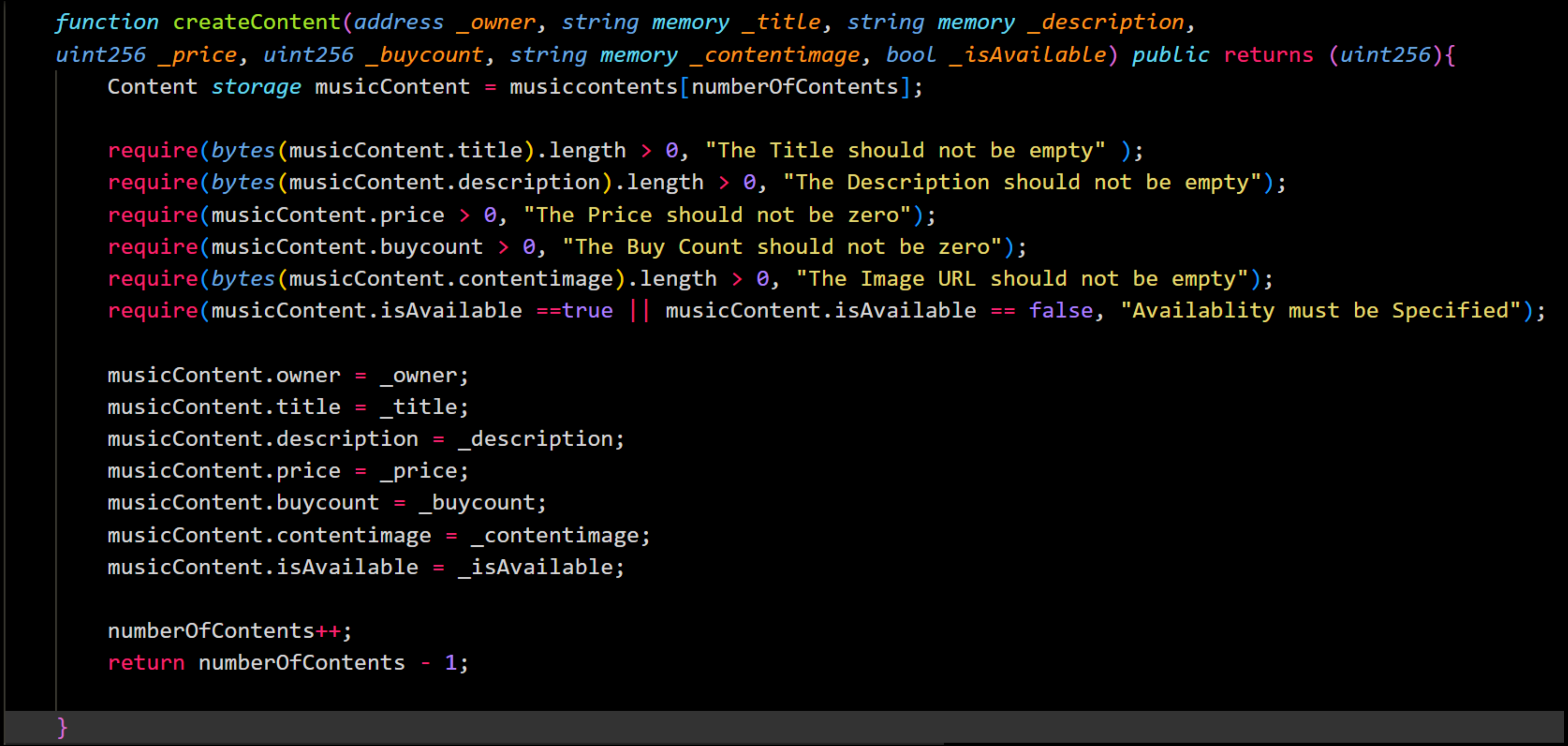
- Create and publish digital content.
- Set pricing and availability for content.
- Facilitate secure content purchases with Ether.
- Track and manage buyers.
- Maintain content availability.
- Customize content details (title, description, image).
- Establish immutable proof of ownership and transaction history.

SMART CONTRACT



```
contract OpenChain {
    event ContentPurchased(uint256 indexed contentId, address indexed buyer);
    struct Content {
        address owner;
        string title;
        string description;
        uint256 price;
        uint256 buycount;
        uint256 completedbuycount;
        string contentimage;
        address[] buyers;
        uint256[] totalbuycounts;
        bool isAvailable;
    }
```

CREATE AND PUBLISH CONTENT



```
function createContent(address _owner, string memory _title, string memory _description,
uint256 _price, uint256 _buycount, string memory _contentimage, bool _isAvailable) public returns (uint256){
    Content storage musicContent = musiccontents[numberOfContents];

    require(bytes(musicContent.title).length > 0, "The Title should not be empty" );
    require(bytes(musicContent.description).length > 0, "The Description should not be empty");
    require(musicContent.price > 0, "The Price should not be zero");
    require(musicContent.buycount > 0, "The Buy Count should not be zero");
    require(bytes(musicContent.contentimage).length > 0, "The Image URL should not be empty");
    require(musicContent.isAvailable ==true || musicContent.isAvailable == false, "Availablity must be Specified");

    musicContent.owner = _owner;
    musicContent.title = _title;
    musicContent.description = _description;
    musicContent.price = _price;
    musicContent.buycount = _buycount;
    musicContent.contentimage = _contentimage;
    musicContent.isAvailable = _isAvailable;

    numberOfContents++;
    return numberOfContents - 1;

}
```

```solidity
function buyContent(uint256 _id) public payable {
    Content storage musicContent = musiccontents[_id];
    require(musicContent.isAvailable, "Content is not available for purchase");
    require(msg.value >= musicContent.price, "Insufficient Funds to purchase content");

    address payable owner = payable(musicContent.owner);

    owner.transfer(msg.value);

    musicContent.buyers.push(msg.sender);
    musicContent.totalbuycounts.push(musicContent.buycount);
    musicContent.completedbuycount++;

    if (musicContent.completedbuycount >= musicContent.buycount) {
        musicContent.isAvailable = false;
    }

    emit ContentPurchased(_id, msg.sender);
}
```

BUY CONTENT

GET CONTENT LIST AND BUYERS LIST

```
function getBuyers (uint256 _id) view public returns (address[] memory, uint256[] memory) {
    return (musiccontents[_id].buyers, musiccontents[_id].totalbuycounts);
}

function getContents() public view returns (Content[] memory) {
    Content[] memory allContents = new Content[](numberOfContents);
    for (uint i =0; i< numberOfContents; i++) {
        Content storage item = musiccontents[i];

        allContents[i] = item;
    }

    return allContents;
}
```