# Laboratory Work: Smart Contract Development & Deployment

By Ogay David

**Topic:** ERC20 Tokenomics, Airdrop Logic, and NFT (ERC721) Integration with IPFS.

### 🎯 Objective

In this practical assignment, you will learn the full lifecycle of blockchain assets. You will create your own cryptocurrency (ERC20), build a distribution mechanism (Airdrop) to send it to multiple users, and finally, create and deploy a unique NFT collection (ERC721) with decentralized metadata storage using IPFS.

### 🛠 Prerequisites

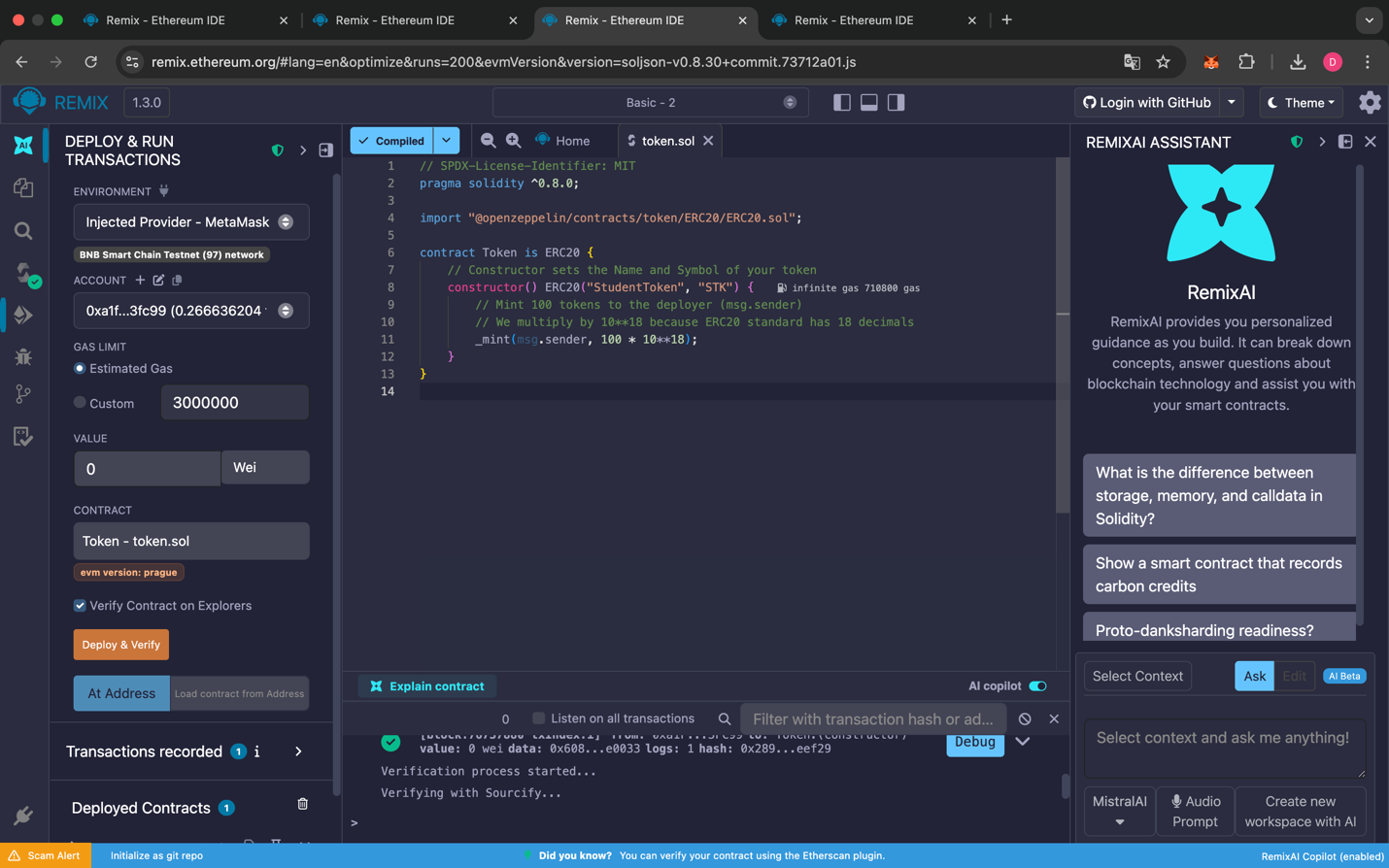
* **IDE:** [Remix Ethereum](https://remix.ethereum.org/)
* **Wallet:** MetaMask (Configured for Sepolia/Amoy Testnet or Remix VM)
* **Storage:** [Pinata](https://pinata.cloud/) (For IPFS management)

## Part 1: Create Your Own Cryptocurrency (ERC20)

**Task:** Develop and deploy a standard ERC20 token smart contract.

**Instructions:**

1. Open Remix IDE and create a file named Token.sol.
2. Copy and paste the code below.
3. Deploy the contract.
4. **Check:** verify that you have received 100 tokens in your wallet balance.

**Code (Token.sol):**

Solidity

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

import "@openzeppelin/contracts/token/ERC20/ERC20.sol";

contract Token is ERC20 {

// Constructor sets the Name and Symbol of your token

constructor() ERC20("StudentToken", "STK") {

// Mint 100 tokens to the deployer (msg.sender)

// We multiply by 10\*\*18 because ERC20 standard has 18 decimals

\_mint(msg.sender, 100 \* 10\*\*18);

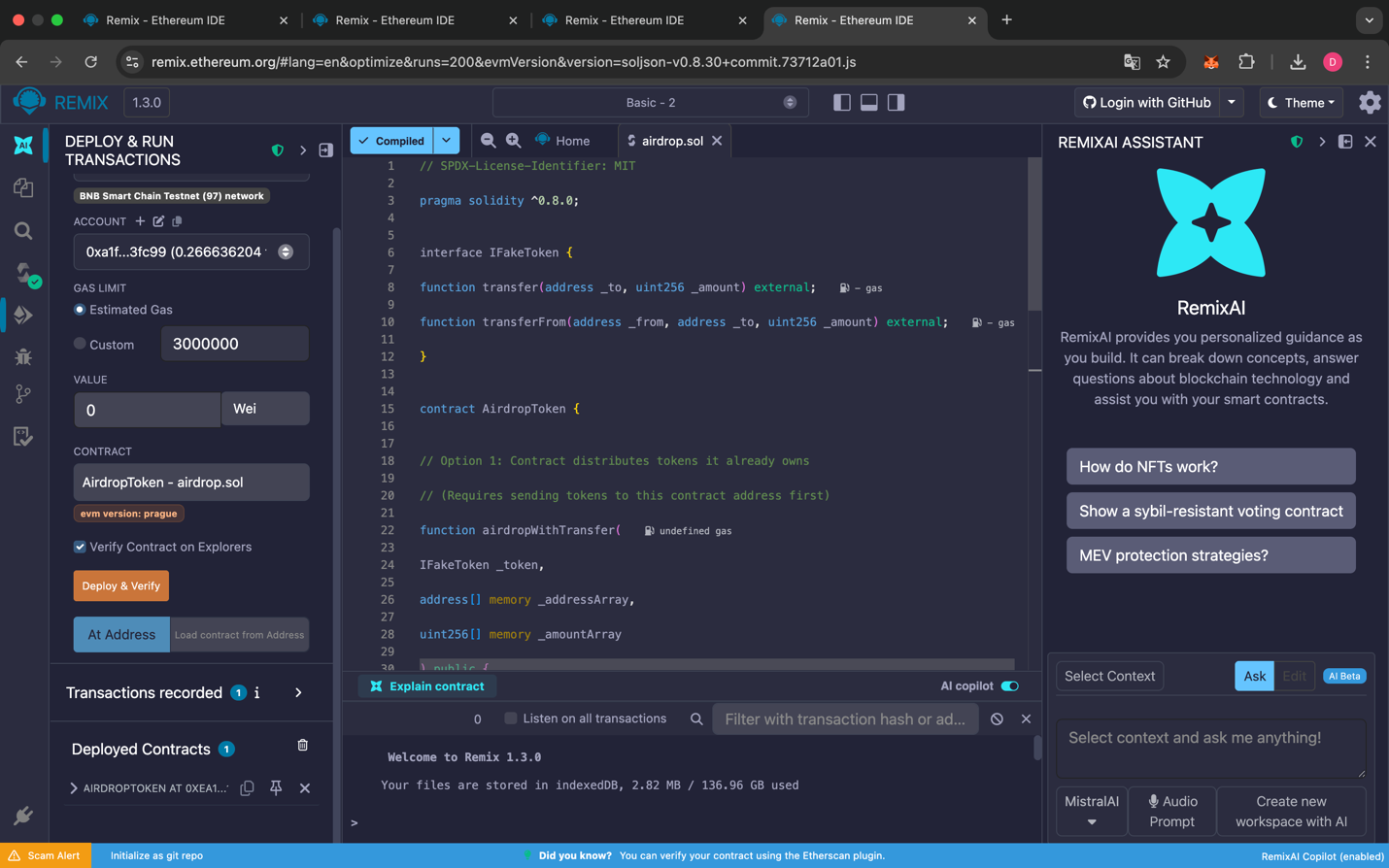
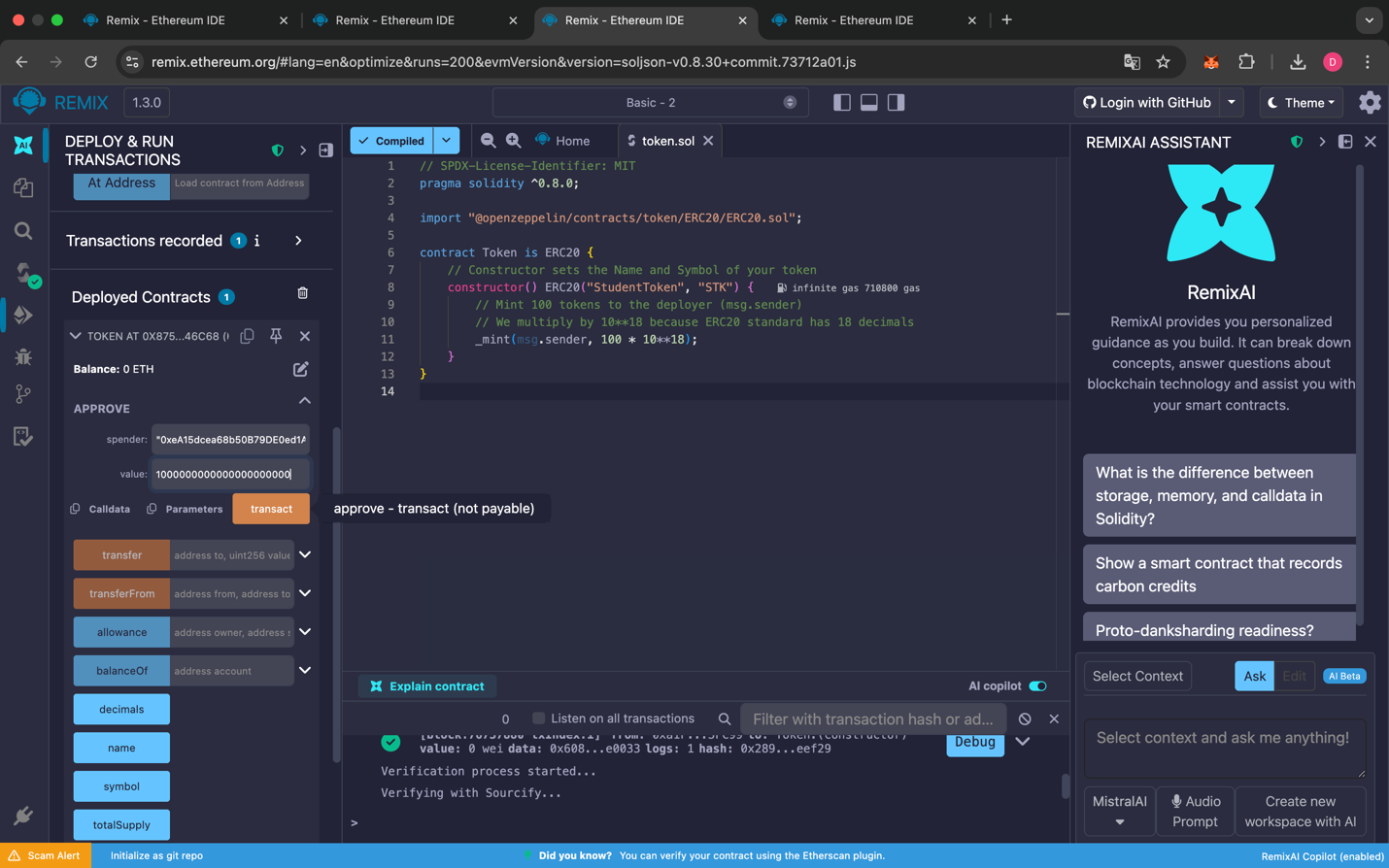
}

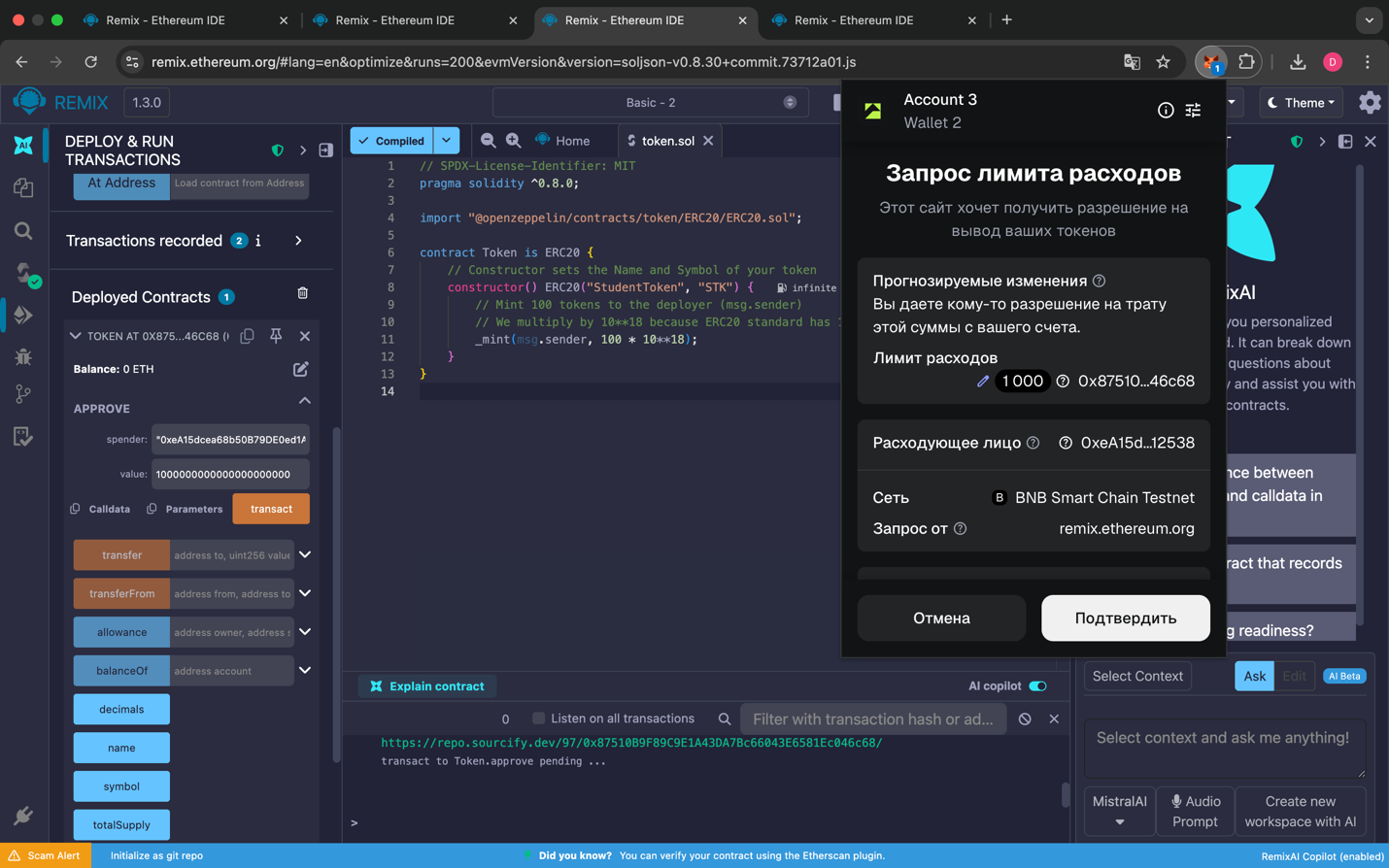
}

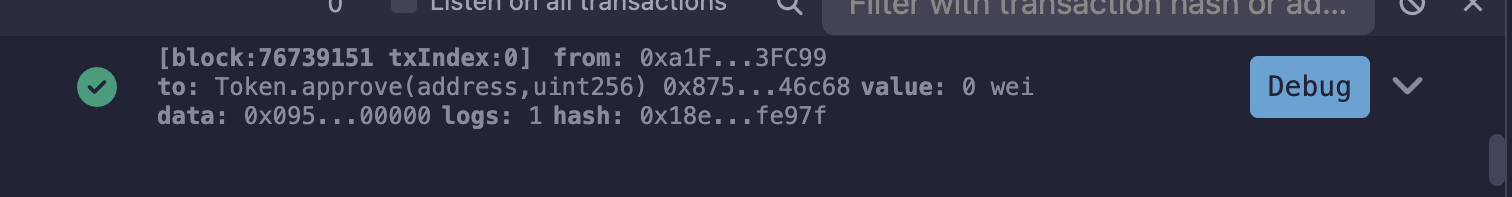
## Part 2: Token Distribution (Airdrop System)

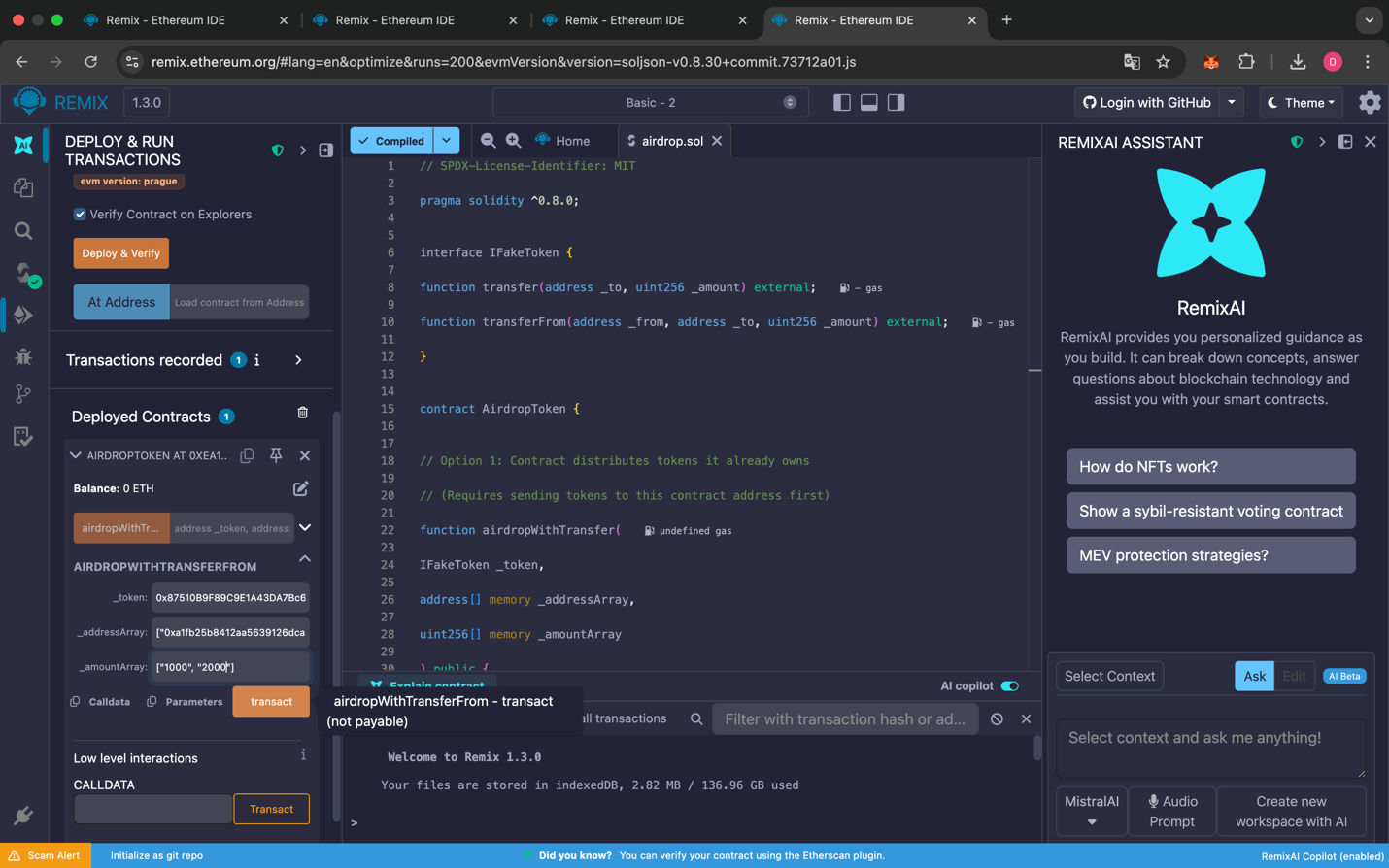
**Task:** Create a mechanism to distribute your tokens to multiple addresses in a single transaction to save gas.

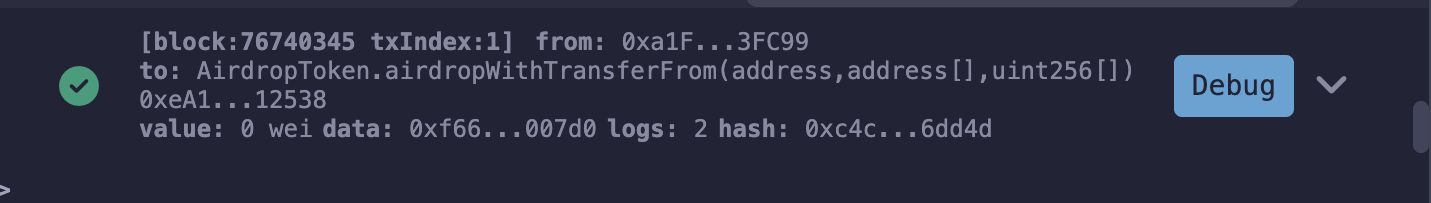
**Instructions:**

1. Create a file named Airdrop.sol.
2. Copy the code below and **Deploy** this contract.
3. **Approve the Contract:**
   * Go back to your deployed **Token Contract** (from Part 1).
   * Find the approve function.
   * In the spender field, paste the **Address of the Airdrop Contract**.
   * In the amount field, enter a large number (e.g., 1000000000000000000000).
   * Click **Transact**.





1. **Execute Airdrop:**
   * Go to the **Airdrop Contract**.
   * Use the airdropWithTransferFrom function.
   * Input the Token Address, an array of recipient addresses ["0x...", "0x..."], and an array of amounts.



**Code (Airdrop.sol):**

Solidity

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

interface IFakeToken {

function transfer(address \_to, uint256 \_amount) external;

function transferFrom(address \_from, address \_to, uint256 \_amount) external;

}

contract AirdropToken {

// Option 1: Contract distributes tokens it already owns

// (Requires sending tokens to this contract address first)

function airdropWithTransfer(

IFakeToken \_token,

address[] memory \_addressArray,

uint256[] memory \_amountArray

) public {

require(\_addressArray.length == \_amountArray.length, "Arrays must have same length");

for (uint256 i = 0; i < \_addressArray.length; i++) {

\_token.transfer(\_addressArray[i], \_amountArray[i]);

}

}

// Option 2: Contract distributes YOUR tokens

// (Requires you to 'Approve' this contract address first)

function airdropWithTransferFrom(

IFakeToken \_token,

address[] memory \_addressArray,

uint256[] memory \_amountArray

) public {

require(\_addressArray.length == \_amountArray.length, "Arrays must have same length");

for (uint256 i = 0; i < \_addressArray.length; i++) {

\_token.transferFrom(msg.sender, \_addressArray[i], \_amountArray[i]);

}

}

}

## Part 3: NFT Collection with Metadata (ERC721 + IPFS)

**Task:** Create a digital art collection where the image and data are stored on the decentralized web (IPFS).

### Step A: Preparing IPFS Metadata (Crucial Step)

1. **Register & Upload Image:**
   * Go to [Pinata](https://pinata.cloud/" \t "_blank) and create a free account.
   * Click **Add Files**  **File**.
   * Select your art image (PNG/JPG) from your computer and upload it.
2. **Get Image CID:**
   * Once uploaded, find your file in the list.
   * Look at the **CID column**. It is a long string of characters (e.g., QmXyZ123...).
   * **Copy this CID.**
3. **Create Metadata File:**
   * Create a new text file on your computer named 0.json.
   * Paste the code below into the file. **Replace** YOUR\_IMAGE\_CID with the code you just copied from Pinata.

JSON

{

"name": "Student NFT #0",

"description": "My first practical NFT work",

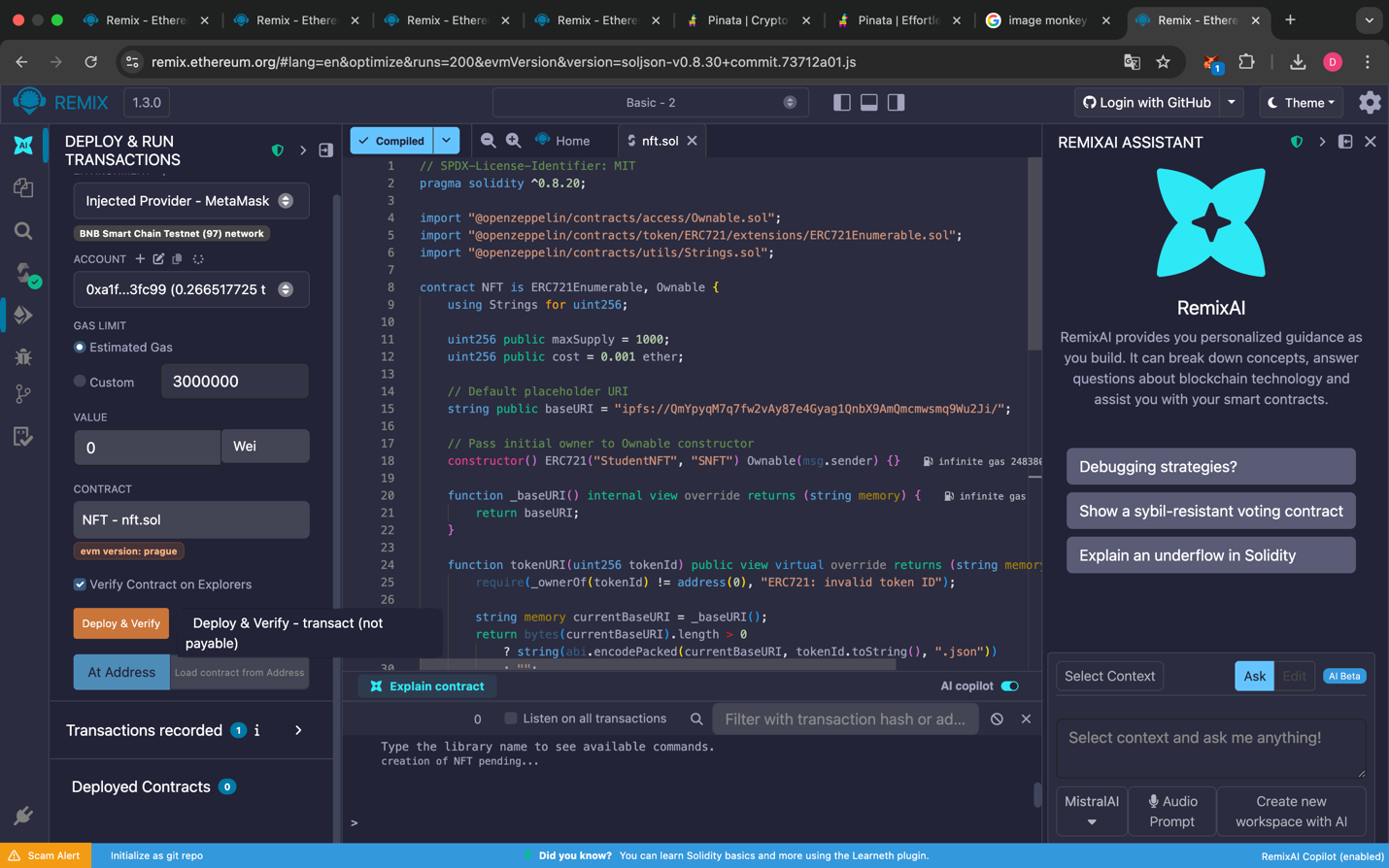
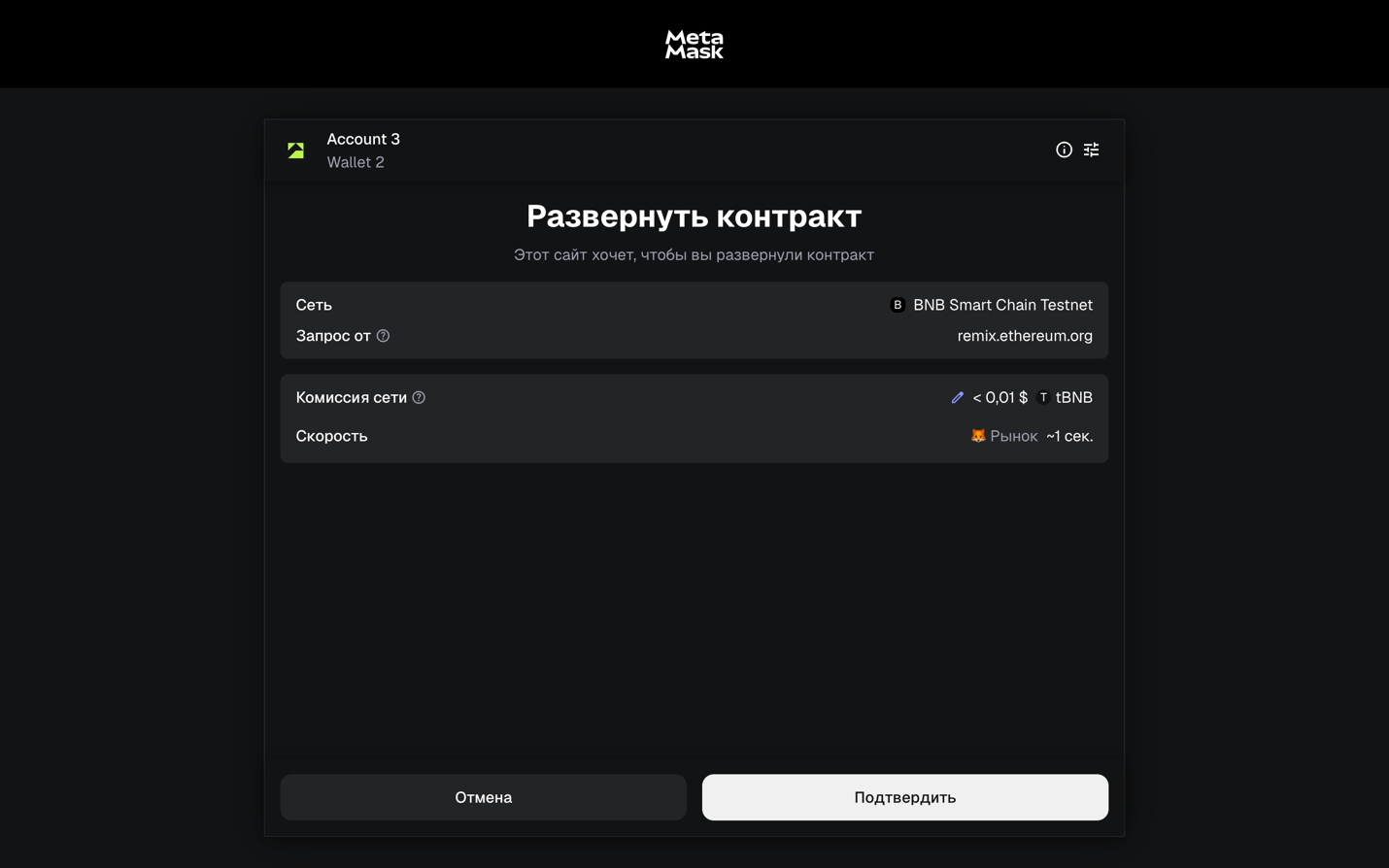
"image": "ipfs://QmXyZ123..."

}

*(Note: Ensure the format is ipfs:// followed immediately by your CID).*

1. **Upload Metadata Folder:**
   * Put the 0.json file into a folder named metadata.
   * Go back to Pinata. Click **Add Files**  **Folder**.
   * Upload the metadata folder.
   * **Copy the CID of this folder.** (This is your FOLDER\_CID).

### Step B: Smart Contract Deployment

1. Create NFT.sol in Remix and use the code below.
2. **Deploy** the contract.

**Code (NFT.sol):**

Solidity

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.20;

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

import "@openzeppelin/contracts/token/ERC721/extensions/ERC721Enumerable.sol";

import "@openzeppelin/contracts/utils/Strings.sol";

contract NFT is ERC721Enumerable, Ownable {

using Strings for uint256;

uint256 public maxSupply = 1000;

uint256 public cost = 0.001 ether;

// Default placeholder URI

string public baseURI = "ipfs://QmYpyqM7q7fw2vAy87e4Gyag1QnbX9AmQmcmwsmq9Wu2Ji/";

// Pass initial owner to Ownable constructor

constructor() ERC721("StudentNFT", "SNFT") Ownable(msg.sender) {}

function \_baseURI() internal view override returns (string memory) {

return baseURI;

}

function tokenURI(uint256 tokenId) public view virtual override returns (string memory) {

require(\_ownerOf(tokenId) != address(0), "ERC721: invalid token ID");

string memory currentBaseURI = \_baseURI();

return bytes(currentBaseURI).length > 0

? string(abi.encodePacked(currentBaseURI, tokenId.toString(), ".json"))

: "";

}

function changeBaseURI(string memory \_newBaseURI) public onlyOwner {

baseURI = \_newBaseURI;

}

function safeMint(address \_to) public payable {

uint256 \_currentSupply = totalSupply();

require(\_currentSupply < maxSupply, "Max supply reached");

require(msg.value == cost, "Please add valid amount of ETH (0.001)");

\_safeMint(\_to, \_currentSupply);

}

function withdraw() public onlyOwner {

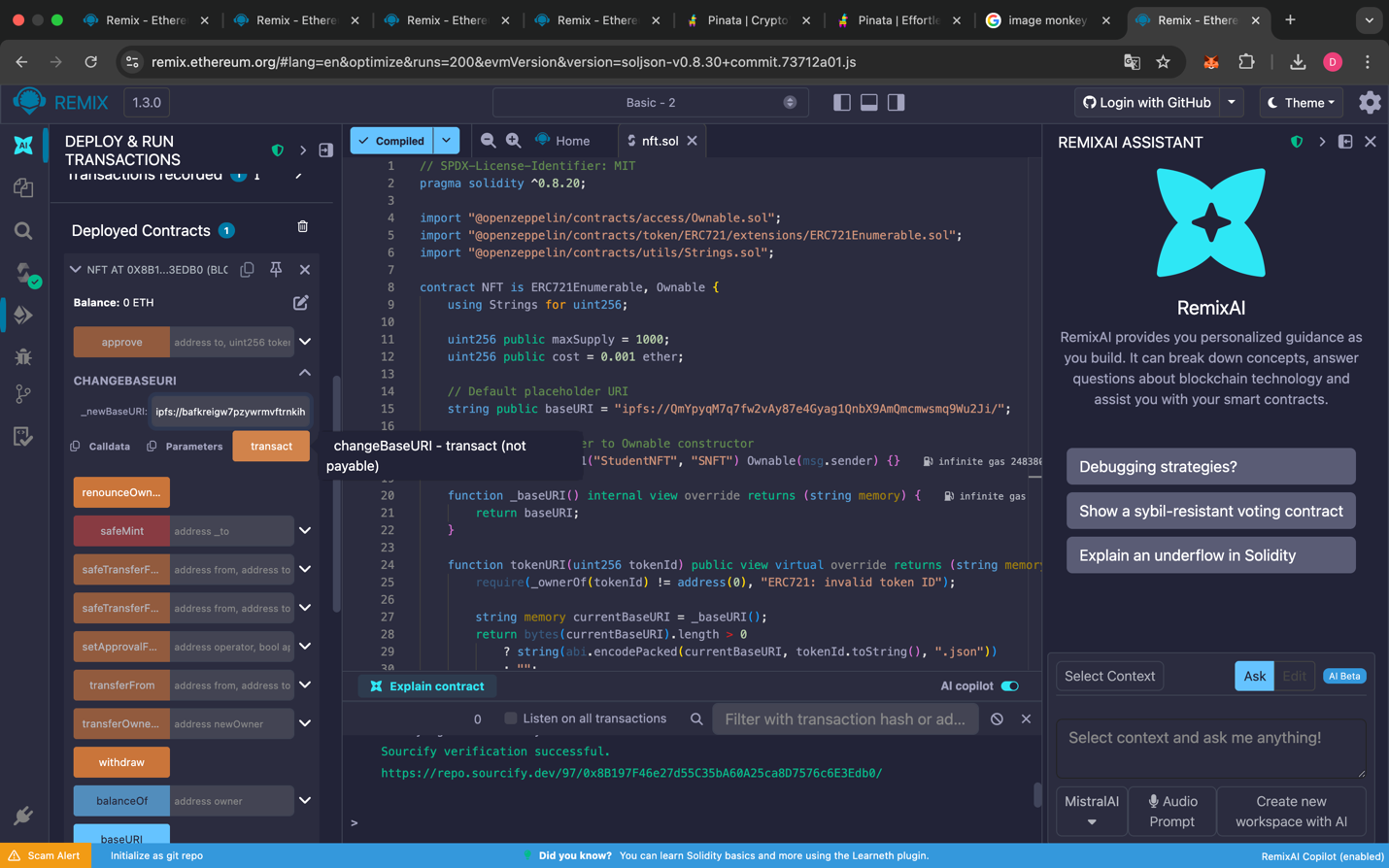
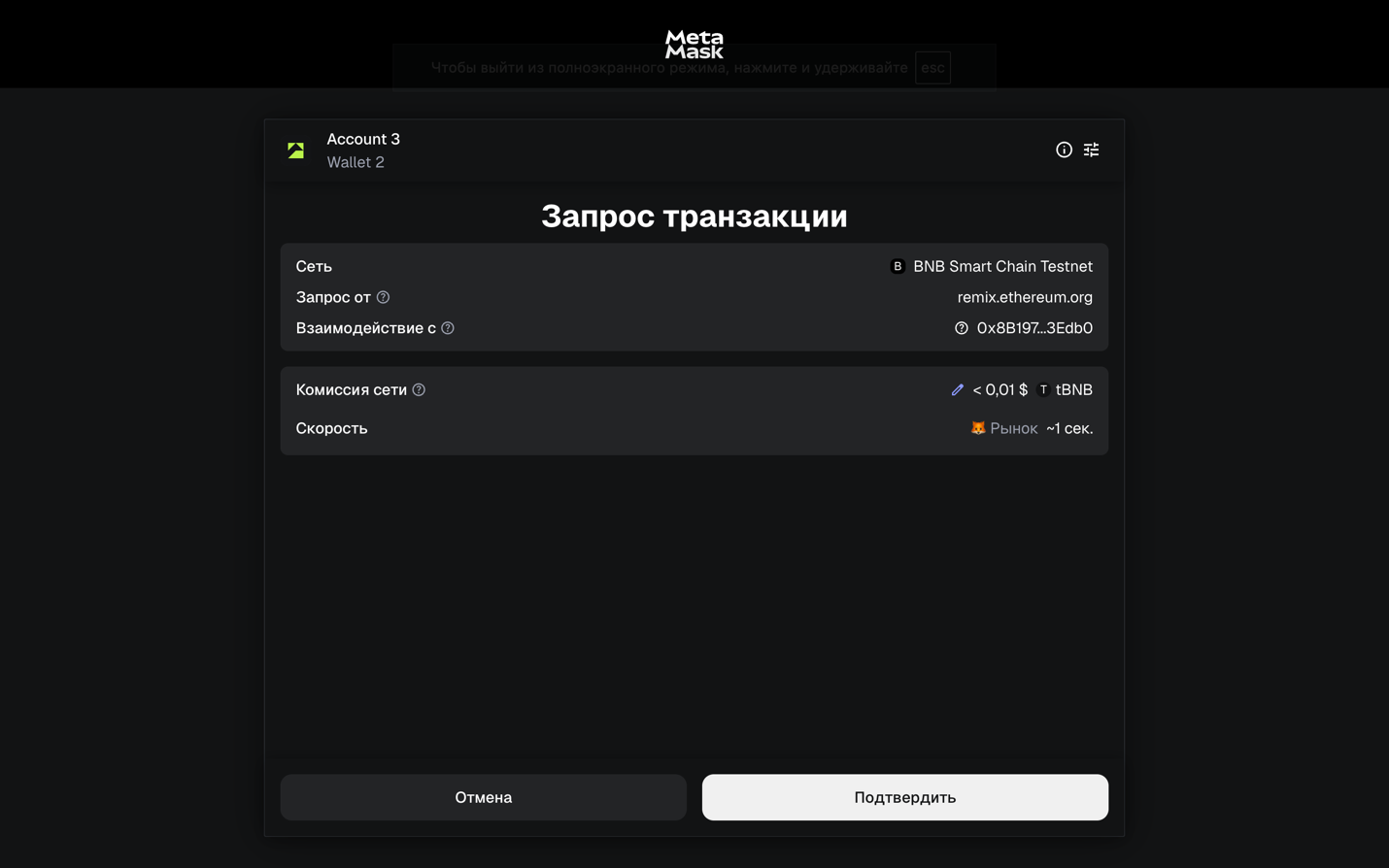
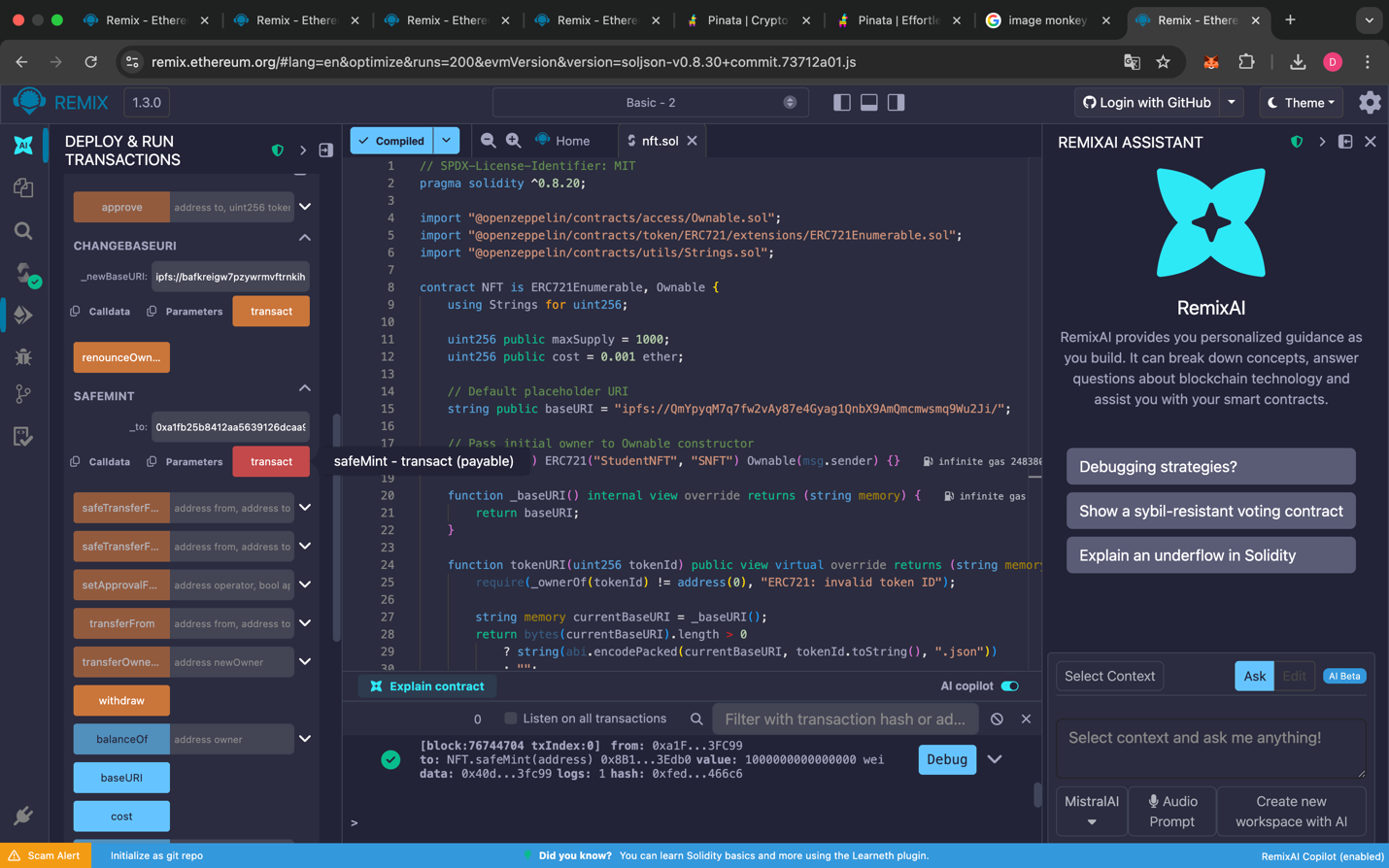
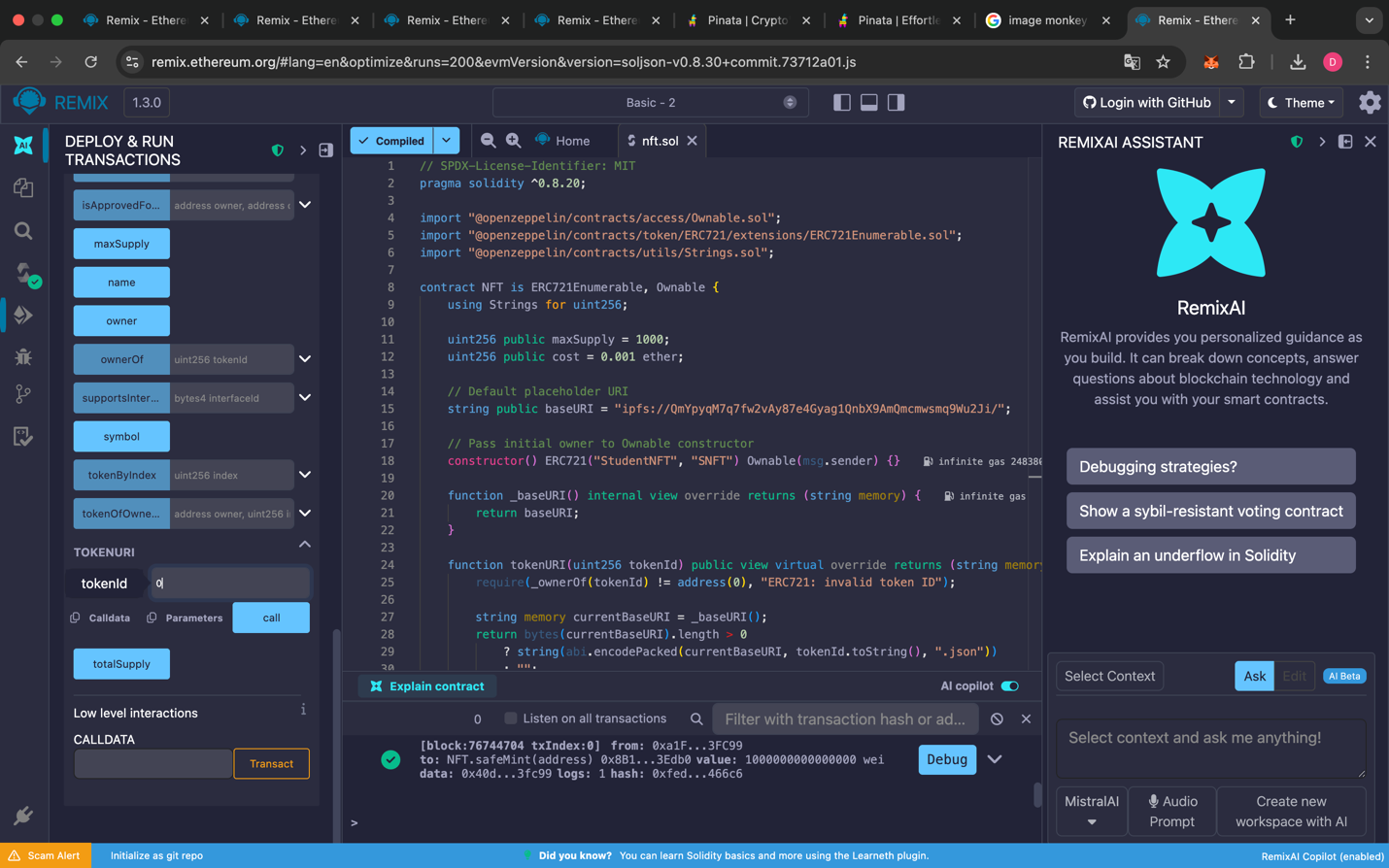
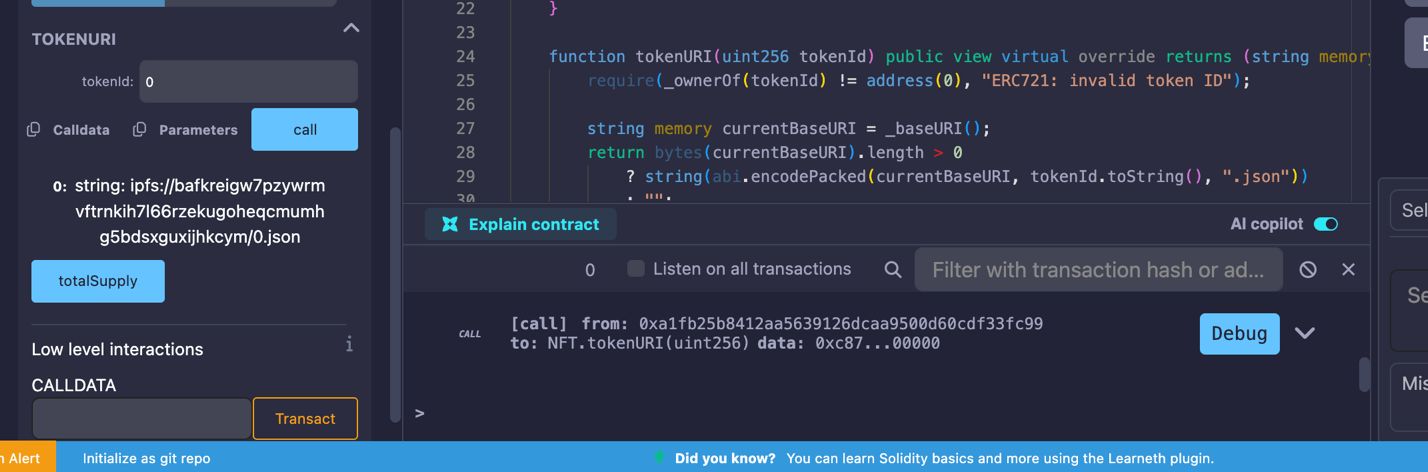
(bool success, ) = payable(msg.sender).call{value: address(this).balance}("");

require(success, "Withdraw failed");

}

}

### Step C: Linking & Minting

1. **Link IPFS:**
   * In the deployed NFT contract, find the changeBaseURI function.
   * Paste your Folder CID in this format: ipfs://YOUR\_FOLDER\_CID/
   * **Important:** You MUST include the slash / at the end.
   * Click **Transact**.
2. **Mint NFT:**
   * Find the safeMint function.
   * Enter your address in \_to.
   * Scroll up to the **Value** field (top left of Remix). Enter 1000000 and select **Gwei** (this equals 0.001 Ether).
   * Click **Transact**.
3. **Verify:**
   * Call tokenURI with id 0.
   * It should return: ipfs://YOUR\_FOLDER\_CID/0.json.

### 📝 Submission Requirements

1. **Source Code:** The solidity code for all three contracts.
2. **Screenshots:**
   * Your Token balance after deployment.
   * The successful Airdrop transaction logs.
   * The tokenURI result in Remix showing your unique IPFS link.
3. **Links:**
   * Link to your metadata file on the IPFS gateway (e.g., <https://gateway.pinata.cloud/ipfs/bafybeiab5gzrdc3adlzdaeq3o6h6tgaftsun42tvxvx6vjvsgl6zfp25q4/0.json>).

