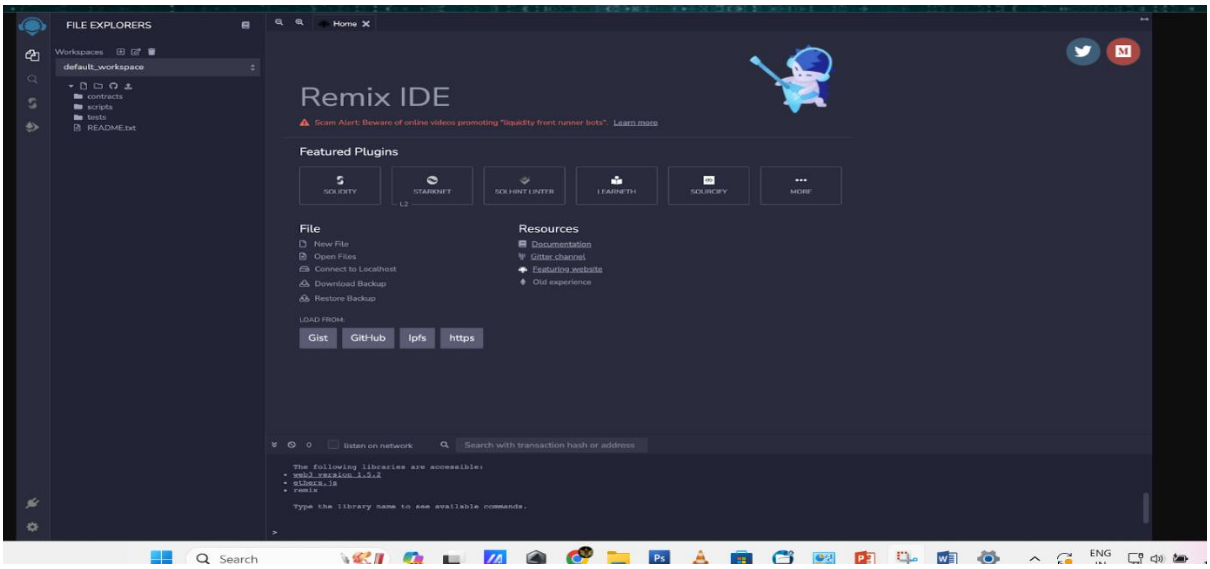
IMPLEMENTATION:

Remix IDE:

Remix IDE is a no-setup tool with a GUI for developing smart contracts. Used by experts and beginners alike, Remix will get you going in double time. Remix plays well with other tools, and allows for a simple deployment process to the chain of your choice. Remix is famous for its visual debugger.

STEP 1:

Open **Remix IDE**.

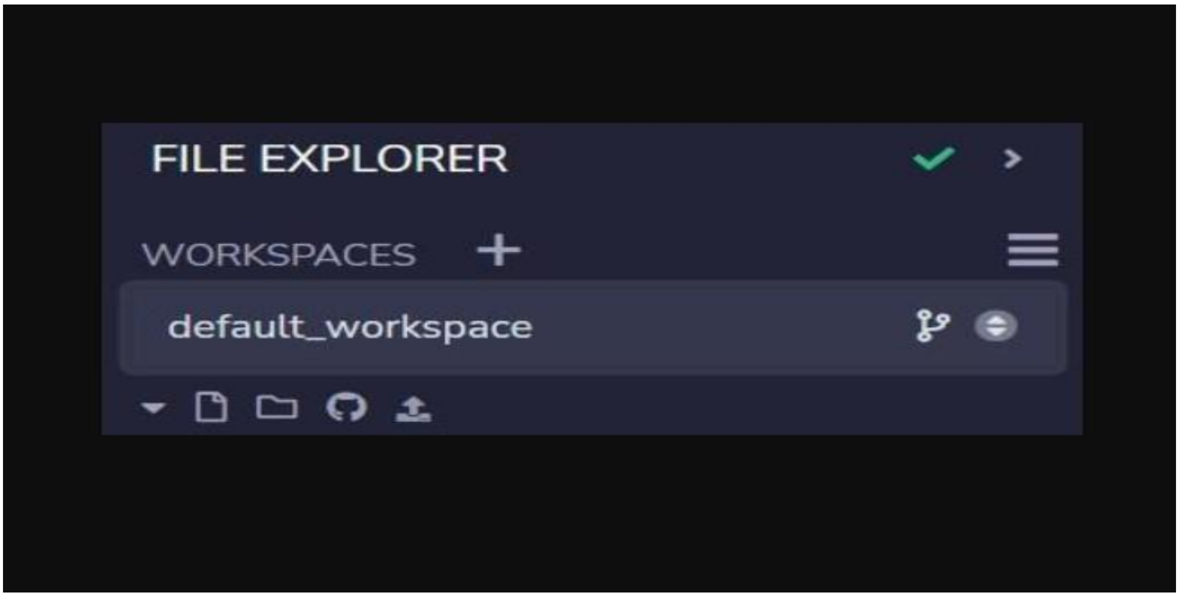


Remix IDE

Scam Alert: Beware of online videos promoting "liquidity front runner bots". Learn more

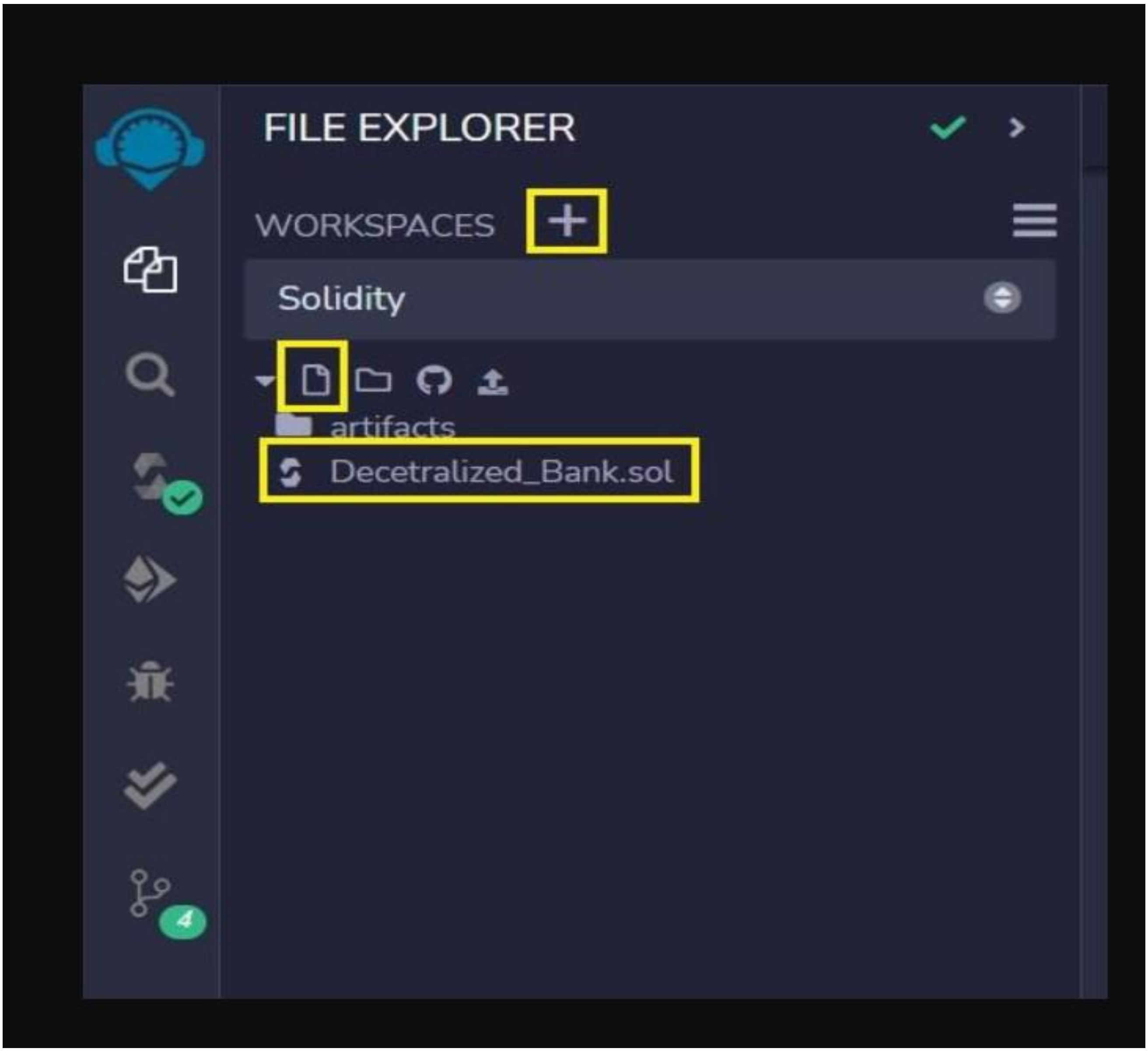
```
The following libraries are accessible:
• web3 version 1.5.2
• ethers.js
• remix

Type the library name to see available commands.
```

- Click on **File Explorers** and Create a new **WORKSPACE (by Clicking on the + icon).**



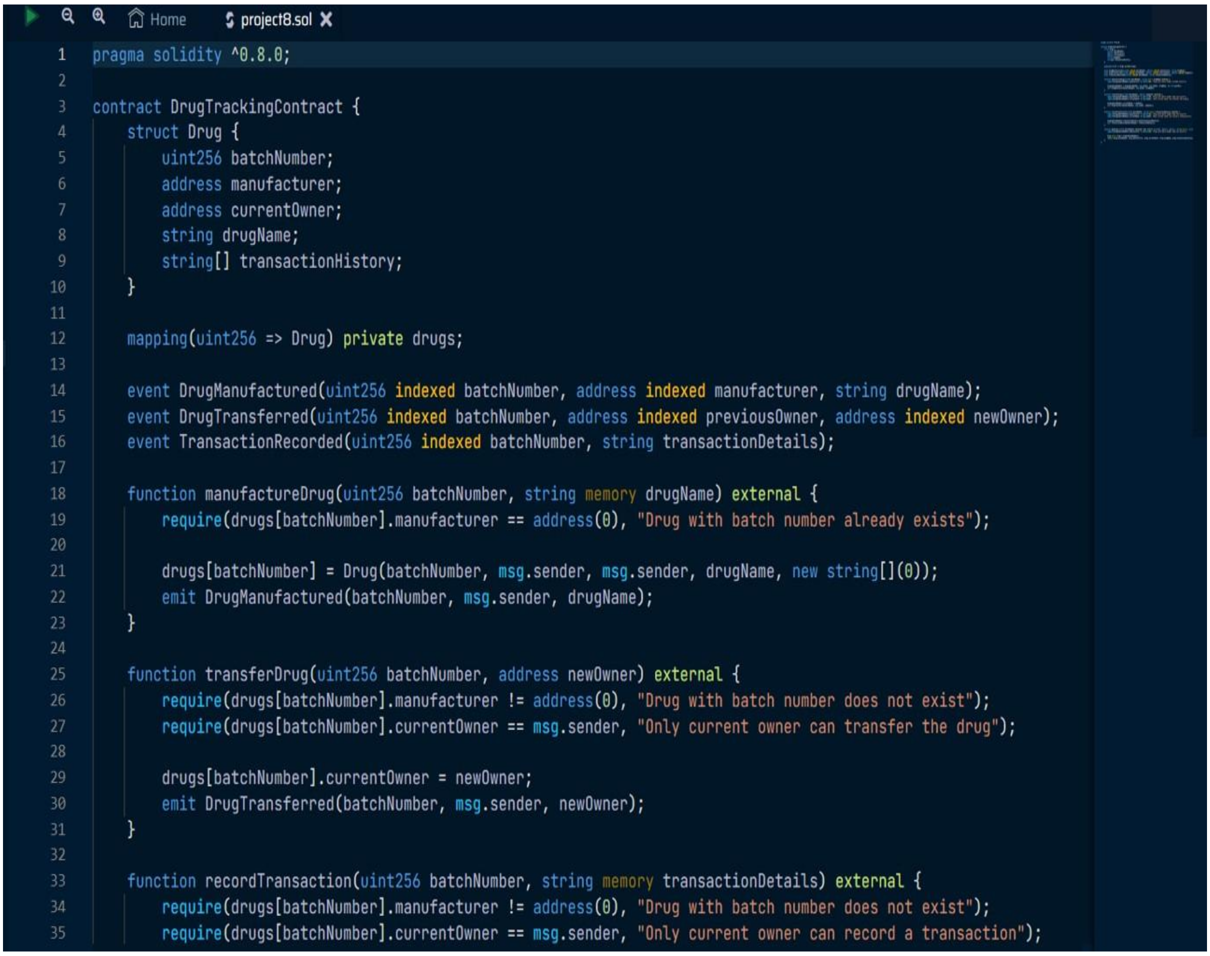
STEP 2:

Click on File Explorers and select Solidity in the environment and create a new file filename. sol by clicking on New File section.

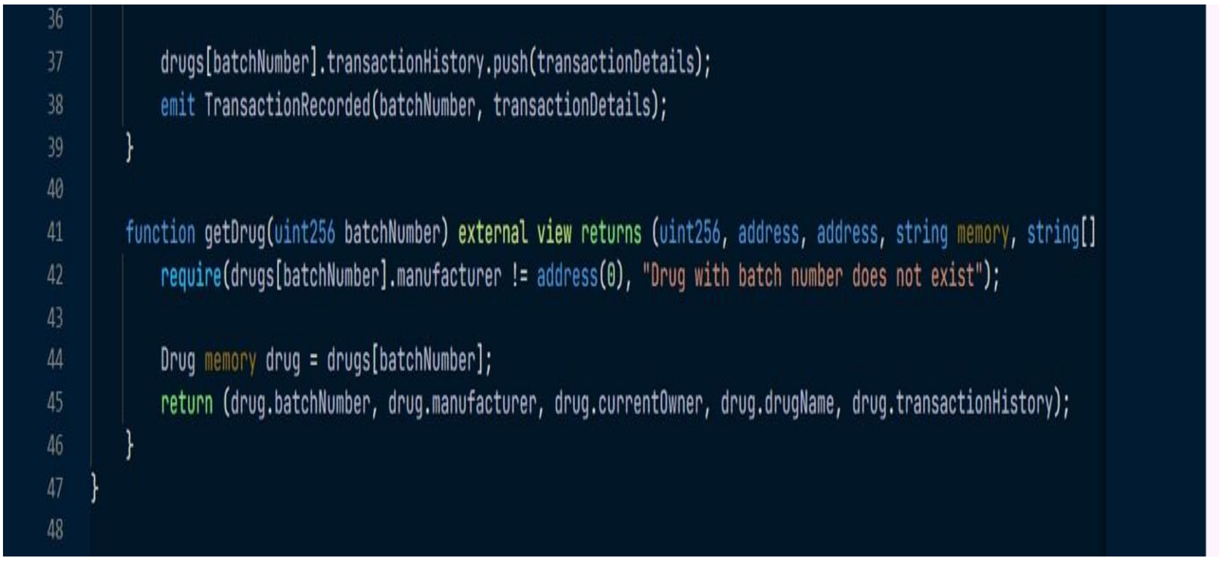


STEP 3:

- Add the solidity code in the Filename. sol file.

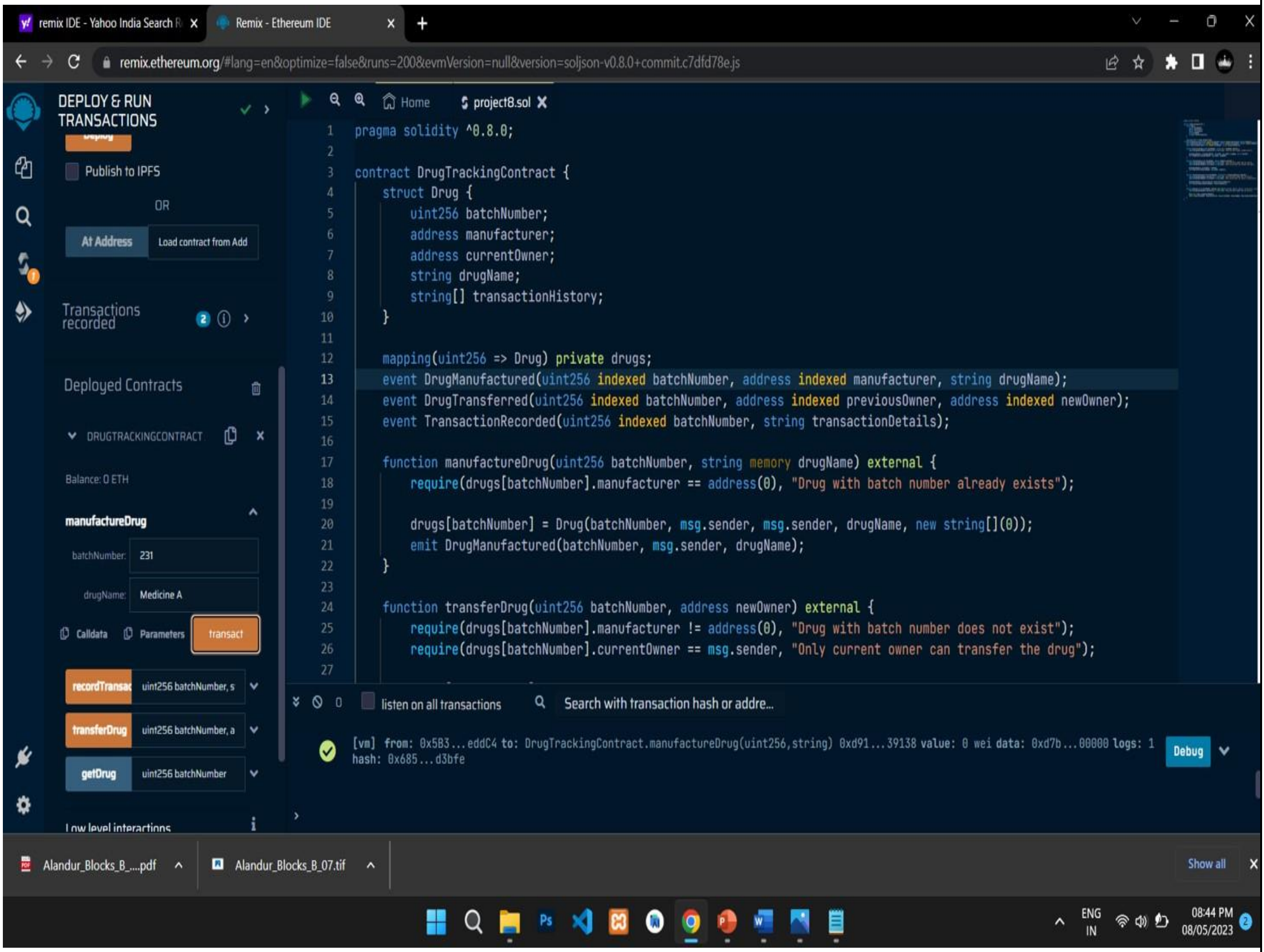


```
pragma solidity ^0.8.0;

contract DrugTrackingContract {
    struct Drug {
        uint256 batchNumber;
        address manufacturer;
        address currentOwner;
        string drugName;
        string[] transactionHistory;
    }

    mapping(uint256 => Drug) private drugs;

    event DrugManufactured(uint256 indexed batchNumber, address indexed manufacturer, string drugName);
    event DrugTransferred(uint256 indexed batchNumber, address indexed previousOwner, address indexed newOwner);
    event TransactionRecorded(uint256 indexed batchNumber, string transactionDetails);

    function manufactureDrug(uint256 batchNumber, string memory drugName) external {
        require(drugs[batchNumber].manufacturer == address(0), "Drug with batch number already exists");

        drugs[batchNumber] = Drug(batchNumber, msg.sender, msg.sender, drugName, new string[](0));
        emit DrugManufactured(batchNumber, msg.sender, drugName);
    }

    function transferDrug(uint256 batchNumber, address newOwner) external {
        require(drugs[batchNumber].manufacturer != address(0), "Drug with batch number does not exist");
        require(drugs[batchNumber].currentOwner == msg.sender, "Only current owner can transfer the drug");

        drugs[batchNumber].currentOwner = newOwner;
        emit DrugTransferred(batchNumber, msg.sender, newOwner);
    }

    function recordTransaction(uint256 batchNumber, string memory transactionDetails) external {
        require(drugs[batchNumber].manufacturer != address(0), "Drug with batch number does not exist");
        require(drugs[batchNumber].currentOwner == msg.sender, "Only current owner can record a transaction");
```



```

        drugs[batchNumber].transactionHistory.push(transactionDetails);
        emit TransactionRecorded(batchNumber, transactionDetails);
    }

    function getDrug(uint256 batchNumber) external view returns (uint256, address, address, string memory, string[]
        require(drugs[batchNumber].manufacturer != address(0), "Drug with batch number does not exist");

        Drug memory drug = drugs[batchNumber];
        return (drug.batchNumber, drug.manufacturer, drug.currentOwner, drug.drugName, drug.transactionHistory);
    }
}

```

Step 4: Create a file in contract folder and paste the smart contract code in the file



Balance: 0 ETH

manufactureDrug

batchNumber: 231

drugName: Medicine A

recordTransac: uint256 batchNumber, s

transferDrug: uint256 batchNumber, a

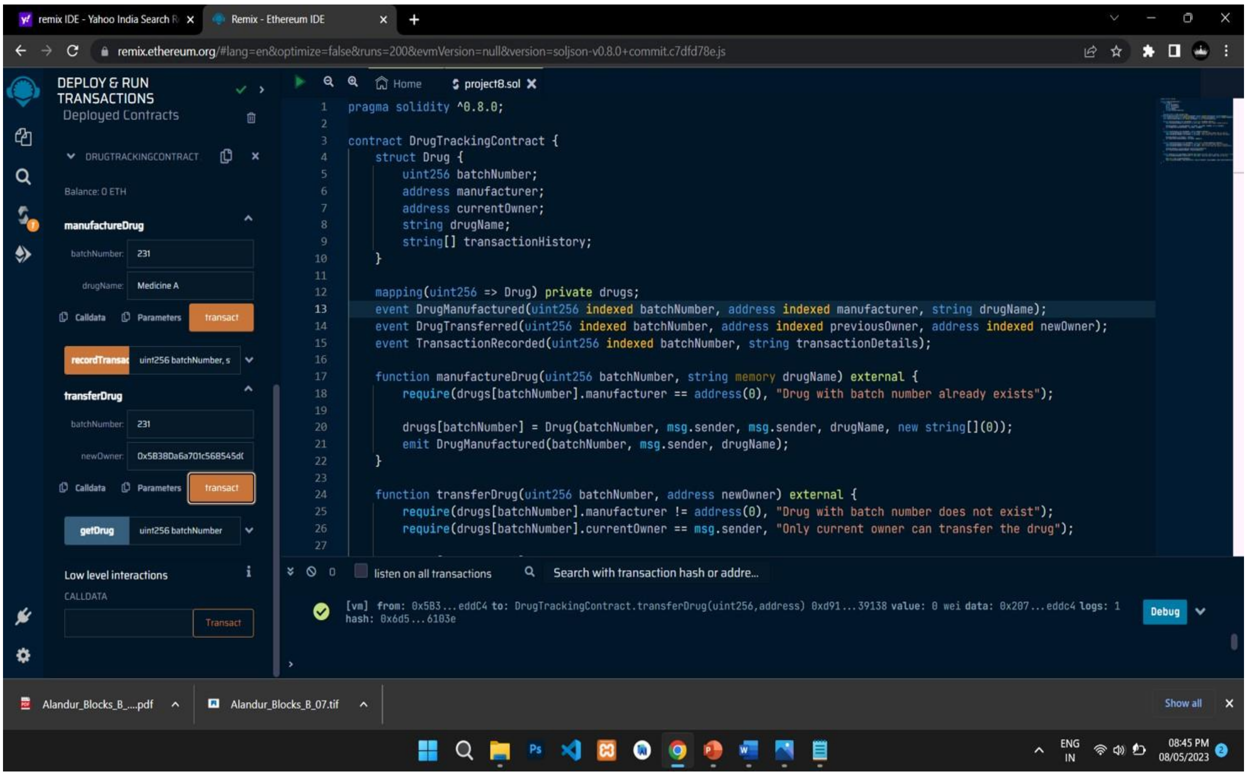
getDrug: uint256 batchNumber

```solidity
pragma solidity ^0.8.0;

contract DrugTrackingContract {
    struct Drug {
        uint256 batchNumber;
        address manufacturer;
        address currentOwner;
        string drugName;
        string[] transactionHistory;
    }

    mapping(uint256 => Drug) private drugs;
    event DrugManufactured(uint256 indexed batchNumber, address indexed manufacturer, string drugName);
    event DrugTransferred(uint256 indexed batchNumber, address indexed previousOwner, address indexed newOwner);
    event TransactionRecorded(uint256 indexed batchNumber, string transactionDetails);

    function manufactureDrug(uint256 batchNumber, string memory drugName) external {
        require(drugs[batchNumber].manufacturer == address(0), "Drug with batch number already exists");

        drugs[batchNumber] = Drug(batchNumber, msg.sender, msg.sender, drugName, new string[](0));
        emit DrugManufactured(batchNumber, msg.sender, drugName);
    }

    function transferDrug(uint256 batchNumber, address newOwner) external {
        require(drugs[batchNumber].manufacturer != address(0), "Drug with batch number does not exist");
        require(drugs[batchNumber].currentOwner == msg.sender, "Only current owner can transfer the drug");
```

listen on all transactions

Search with transaction hash or addre...

```
[vm] from: 0x5B3...eddC4 to: DrugTrackingContract.manufactureDrug(uint256,string) 0xd91...39138 value: 0 wei data: 0xd7b...00000 logs: 1
hash: 0x685...d3bfe
```



Balance: 0 ETH

manufactureDrug

batchNumber: 231

drugName: Medicine A

recordTransac: uint256 batchNumber, s

transferDrug

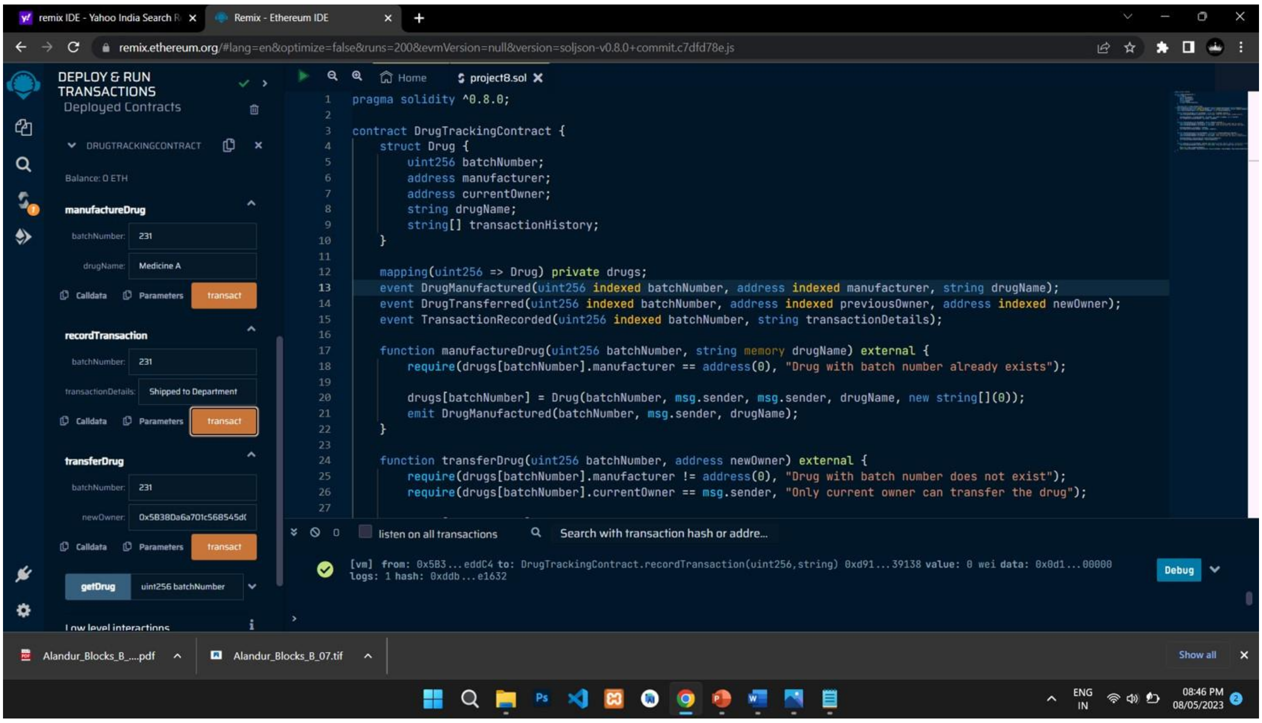
batchNumber: 231

newOwner: 0x5B38Da6a701c568545d(

getDrug: uint256 batchNumber

```
pragma solidity ^0.8.0;

contract DrugTrackingContract {
    struct Drug {
        uint256 batchNumber;
        address manufacturer;
        address currentOwner;
        string drugName;
        string[] transactionHistory;
    }

    mapping(uint256 => Drug) private drugs;
    event DrugManufactured(uint256 indexed batchNumber, address indexed manufacturer, string drugName);
    event DrugTransferred(uint256 indexed batchNumber, address indexed previousOwner, address indexed newOwner);
    event TransactionRecorded(uint256 indexed batchNumber, string transactionDetails);

    function manufactureDrug(uint256 batchNumber, string memory drugName) external {
        require(drugs[batchNumber].manufacturer == address(0), "Drug with batch number already exists");

        drugs[batchNumber] = Drug(batchNumber, msg.sender, msg.sender, drugName, new string[](0));
        emit DrugManufactured(batchNumber, msg.sender, drugName);
    }

    function transferDrug(uint256 batchNumber, address newOwner) external {
        require(drugs[batchNumber].manufacturer != address(0), "Drug with batch number does not exist");
        require(drugs[batchNumber].currentOwner == msg.sender, "Only current owner can transfer the drug");
```

```
[vm] from: 0x5B3...eddC4 to: DrugTrackingContract.transferDrug(uint256,address) 0xd91...39138 value: 0 wei data: 0x207...eddc4 logs: 1
hash: 0x6d5...6103e
```


remix IDE - Yahoo India Search R
Remix - Ethereum IDE
remix.ethereum.org/#lang=en&optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.0+commit.c7dfd78e.js
DEPLOY & RUN TRANSACTIONS
Deployed Contracts
DRUGTRACKINGCONTRACT
Home
project8.sol
Calldata
Parameters
transact
getDrug
Low level interactions
listen on all transactions
Search with transaction hash or addre...
Debug
Alandur_Blocks_B_....pdf
Alandur_Blocks_B_07.tif
Show all
ENG
IN
08:46 PM
08/05/2023

Balance: 0 ETH

manufactureDrug

batchNumber: 231

drugName: Medicine A

recordTransaction

batchNumber: 231

transactionDetails: Shipped to Department

transferDrug

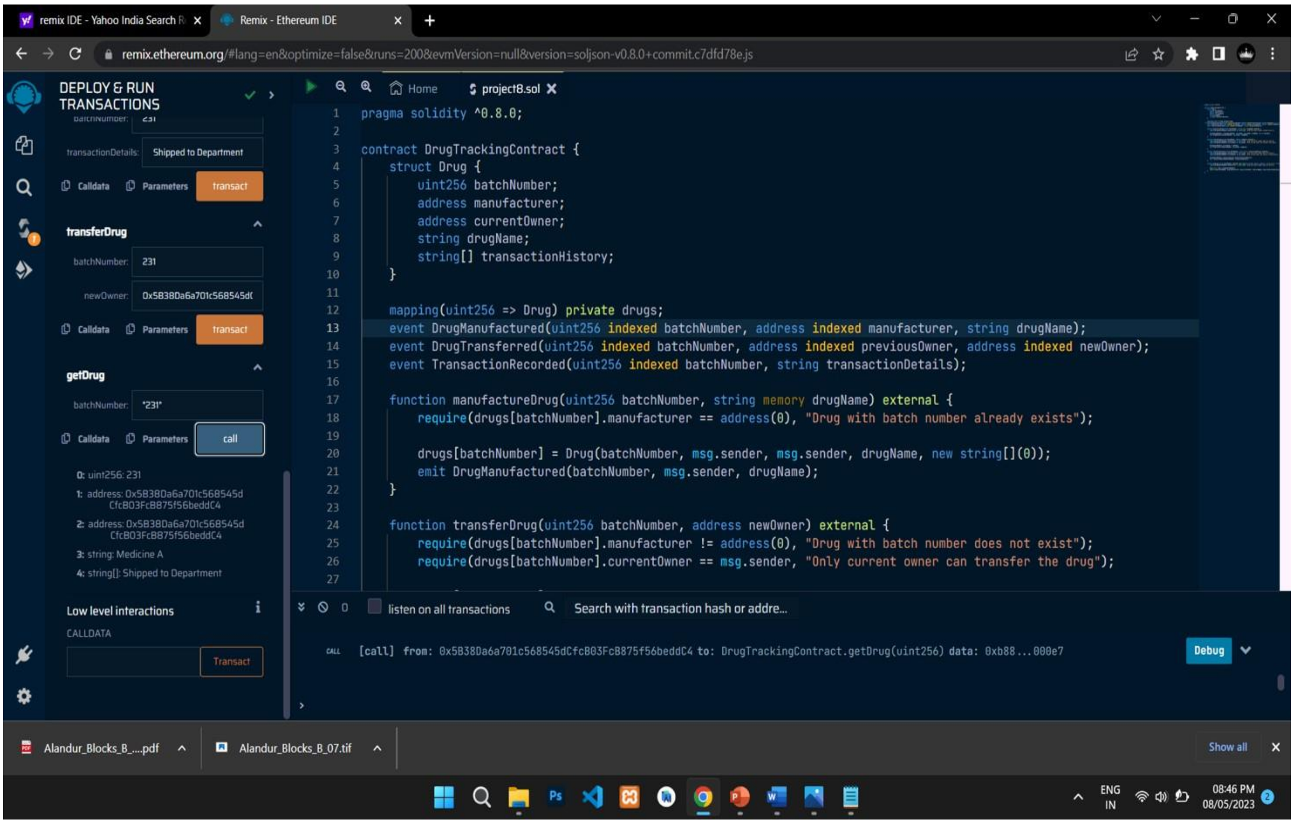
batchNumber: 231

newOwner: 0x5B38Da6a701c568545d(

getDrug: uint256 batchNumber

```
pragma solidity ^0.8.0;

contract DrugTrackingContract {
    struct Drug {
        uint256 batchNumber;
        address manufacturer;
        address currentOwner;
        string drugName;
        string[] transactionHistory;
    }

    mapping(uint256 => Drug) private drugs;
    event DrugManufactured(uint256 indexed batchNumber, address indexed manufacturer, string drugName);
    event DrugTransferred(uint256 indexed batchNumber, address indexed previousOwner, address indexed newOwner);
    event TransactionRecorded(uint256 indexed batchNumber, string transactionDetails);

    function manufactureDrug(uint256 batchNumber, string memory drugName) external {
        require(drugs[batchNumber].manufacturer == address(0), "Drug with batch number already exists");

        drugs[batchNumber] = Drug(batchNumber, msg.sender, msg.sender, drugName, new string[](0));
        emit DrugManufactured(batchNumber, msg.sender, drugName);
    }

    function transferDrug(uint256 batchNumber, address newOwner) external {
        require(drugs[batchNumber].manufacturer != address(0), "Drug with batch number does not exist");
        require(drugs[batchNumber].currentOwner == msg.sender, "Only current owner can transfer the drug");
```

```
[vm] from: 0x5B3...eddC4 to: DrugTrackingContract.recordTransaction(uint256,string) 0xd91...39138 value: 0 wei data: 0x0d1...00000
logs: 1 hash: 0xddb...e1632
```



remix IDE - Yahoo India Search R
Remix - Ethereum IDE
remix.ethereum.org/#lang=en&optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.0+commit.c7dfd78e.js
DEPLOY & RUN TRANSACTIONS
Home
project8.sol
Calldata
Parameters
transact
call
Low level interactions
CALLDATA
Transact
listen on all transactions
Search with transaction hash or addre...
Debug
Alandur_Blocks_B_....pdf
Alandur_Blocks_B_07.tif
Show all
ENG
IN
08:46 PM
08/05/2023

batchNumber: 231

transactionDetails: Shipped to Department

transferDrug

batchNumber: 231

newOwner: 0x5B38Da6a701c568545d(

getDrug

batchNumber: "231"

```
0: uint256: 231
1: address: 0x5B38Da6a701c568545d
            CfcB03FcB875f56beddC4
2: address: 0x5B38Da6a701c568545d
            CfcB03FcB875f56beddC4
3: string: Medicine A
4: string[]: Shipped to Department
```

```solidity
pragma solidity ^0.8.0;

contract DrugTrackingContract {
    struct Drug {
        uint256 batchNumber;
        address manufacturer;
        address currentOwner;
        string drugName;
        string[] transactionHistory;
    }

    mapping(uint256 => Drug) private drugs;
    event DrugManufactured(uint256 indexed batchNumber, address indexed manufacturer, string drugName);
    event DrugTransferred(uint256 indexed batchNumber, address indexed previousOwner, address indexed newOwner);
    event TransactionRecorded(uint256 indexed batchNumber, string transactionDetails);

    function manufactureDrug(uint256 batchNumber, string memory drugName) external {
        require(drugs[batchNumber].manufacturer == address(0), "Drug with batch number already exists");

        drugs[batchNumber] = Drug(batchNumber, msg.sender, msg.sender, drugName, new string[](0));
        emit DrugManufactured(batchNumber, msg.sender, drugName);
    }

    function transferDrug(uint256 batchNumber, address newOwner) external {
        require(drugs[batchNumber].manufacturer != address(0), "Drug with batch number does not exist");
        require(drugs[batchNumber].currentOwner == msg.sender, "Only current owner can transfer the drug");
```

```
CALL [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: DrugTrackingContract.getDrug(uint256) data: 0xb88...000e7
```