**Name: Gondaliya Krish D(19BCE072)**

**Hadiya Pratik D (19BCE072)**

**Date: 10/11/2022**

**Course: Blockchain Technology (2CSDE93)**

**Practical 5**

**Aim:** To perform thorough study and installation of Remix IDE and Truffle IDE for deploying Smart Contracts and Decentralized Applications (dapps) and create and deploy a Smart Contract for any application such as finance, healthcare etc.

This is a group activity. The groups are formed by students and will be evaluated accordingly.

The group will create smart contact for the application of their interest as specified for the term paper evaluation.

**Code:**

pragma solidity >=0.5.3;

contract Open\_Source

{

    struct Tools

    {

        address id;

        string author\_name;

        string tool\_name;

        string contribution\_type;

        string Github\_link;

    }

    struct user

    {

        address id;

        string user;

        string User\_type;

        Tools tool;

    }

    struct issue

    {

        address id;

        Tools tool;

    }

    issue[] private Reports;

    user[] private Users;

    user[] private userdata;

    Tools[] private authordata;

    Tools[] private all\_tools;

    Tools[] private GUI\_tools;

    Tools[] private Website\_tools;

    Tools[] private Database\_tools;

    Tools[] private AI\_tools;

    Tools[] private Blockchain\_tools;

    function GUI\_contribution(string memory author\_name,string memory \_tool,string memory \_link) public

    {

        GUI\_tools.push(Tools(msg.sender,author\_name,\_tool,"GUI\_tool",\_link));

        all\_tools.push(Tools(msg.sender,author\_name,\_tool,"GUI\_tool",\_link));

    }

    function show\_GUI\_tools() public view returns(Tools[] memory)

    {

        return GUI\_tools;

    }

    function Website\_contribution(string memory author\_name,string memory \_tool,string memory \_link) public

    {

        Website\_tools.push(Tools(msg.sender,author\_name,\_tool,"Website\_tool",\_link));

        all\_tools.push(Tools(msg.sender,author\_name,\_tool,"Website\_tool",\_link));

    }

    function show\_Website\_tools() public view returns(Tools[] memory)

    {

        return Website\_tools;

    }

    function Database\_contribution(string memory author\_name,string memory \_tool,string memory \_link) public

    {

        Database\_tools.push(Tools(msg.sender,author\_name,\_tool,"Database\_tool",\_link));

        all\_tools.push(Tools(msg.sender,author\_name,\_tool,"Database\_tool",\_link));

    }

    function show\_Database\_tools() public view returns(Tools[] memory)

    {

        return Database\_tools;

    }

    function AI\_contribution(string memory author\_name,string memory \_tool,string memory \_link) public

    {

        AI\_tools.push(Tools(msg.sender,author\_name,\_tool,"AI\_tool",\_link));

        all\_tools.push(Tools(msg.sender,author\_name,\_tool,"AI\_tool",\_link));

    }

    function show\_AI\_tools() public view returns(Tools[] memory)

    {

        return AI\_tools;

    }

    function Blockchain\_contribution(string memory author\_name,string memory \_tool,string memory \_link) public

    {

        Blockchain\_tools.push(Tools(msg.sender,author\_name,\_tool,"Blockchain\_tool",\_link));

        all\_tools.push(Tools(msg.sender,author\_name,\_tool,"Blockchain\_tool",\_link));

    }

    function show\_Blockchain\_tools() public view returns(Tools[] memory)

    {

        return Blockchain\_tools;

    }

    function get\_authordata() public

    {

        for(uint i=0;i<all\_tools.length;i++)

        {

            if(all\_tools[i].id==msg.sender)

                authordata.push(all\_tools[i]);

        }

    }

    function show\_author\_data() public view returns (Tools[] memory)

    {

        return authordata;

    }

    function use(string memory user\_name,string memory user\_type,string memory tool\_type,uint tool\_index) public

    {

        if(keccak256(abi.encodePacked(tool\_type)) == keccak256(abi.encodePacked("GUI")))

        {

            Users.push(user(msg.sender,user\_name,user\_type,GUI\_tools[tool\_index]));

        }

        else if(keccak256(abi.encodePacked(tool\_type)) == keccak256(abi.encodePacked("Website")))

        {

            Users.push(user(msg.sender,user\_name,user\_type,Website\_tools[tool\_index]));

        }

        else if(keccak256(abi.encodePacked(tool\_type)) == keccak256(abi.encodePacked("Database")))

        {

            Users.push(user(msg.sender,user\_name,user\_type,Database\_tools[tool\_index]));

        }

        else if(keccak256(abi.encodePacked(tool\_type)) == keccak256(abi.encodePacked("AI")))

        {

            Users.push(user(msg.sender,user\_name,user\_type,AI\_tools[tool\_index]));

        }

        else if(keccak256(abi.encodePacked(tool\_type)) == keccak256(abi.encodePacked("Blockchain")))

        {

            Users.push(user(msg.sender,user\_name,user\_type,Blockchain\_tools[tool\_index]));

        }

    }

    function get\_userdata() public

    {

        for(uint i=0;i<Users.length;i++)

        {

            if(Users[i].id==msg.sender)

                userdata.push(Users[i]);

        }

    }

    function show\_user\_data() public view returns (user[] memory)

    {

        return userdata;

    }

    function Report\_issue(string memory tool\_type,uint tool\_index) public

    {

        if(keccak256(abi.encodePacked(tool\_type)) == keccak256(abi.encodePacked("GUI")))

        {

            Reports.push(issue(msg.sender,GUI\_tools[tool\_index]));

        }

        else if(keccak256(abi.encodePacked(tool\_type)) == keccak256(abi.encodePacked("Website")))

        {

            Reports.push(issue(msg.sender,Website\_tools[tool\_index]));

        }

        else if(keccak256(abi.encodePacked(tool\_type)) == keccak256(abi.encodePacked("Database")))

        {

            Reports.push(issue(msg.sender,Database\_tools[tool\_index]));

        }

        else if(keccak256(abi.encodePacked(tool\_type)) == keccak256(abi.encodePacked("AI")))

        {

            Reports.push(issue(msg.sender,AI\_tools[tool\_index]));

        }

        else if(keccak256(abi.encodePacked(tool\_type)) == keccak256(abi.encodePacked("Blockchain")))

        {

            Reports.push(issue(msg.sender,Blockchain\_tools[tool\_index]));

        }

    }

    function show\_reports() public view returns(issue[] memory)

    {

        return Reports;

    }

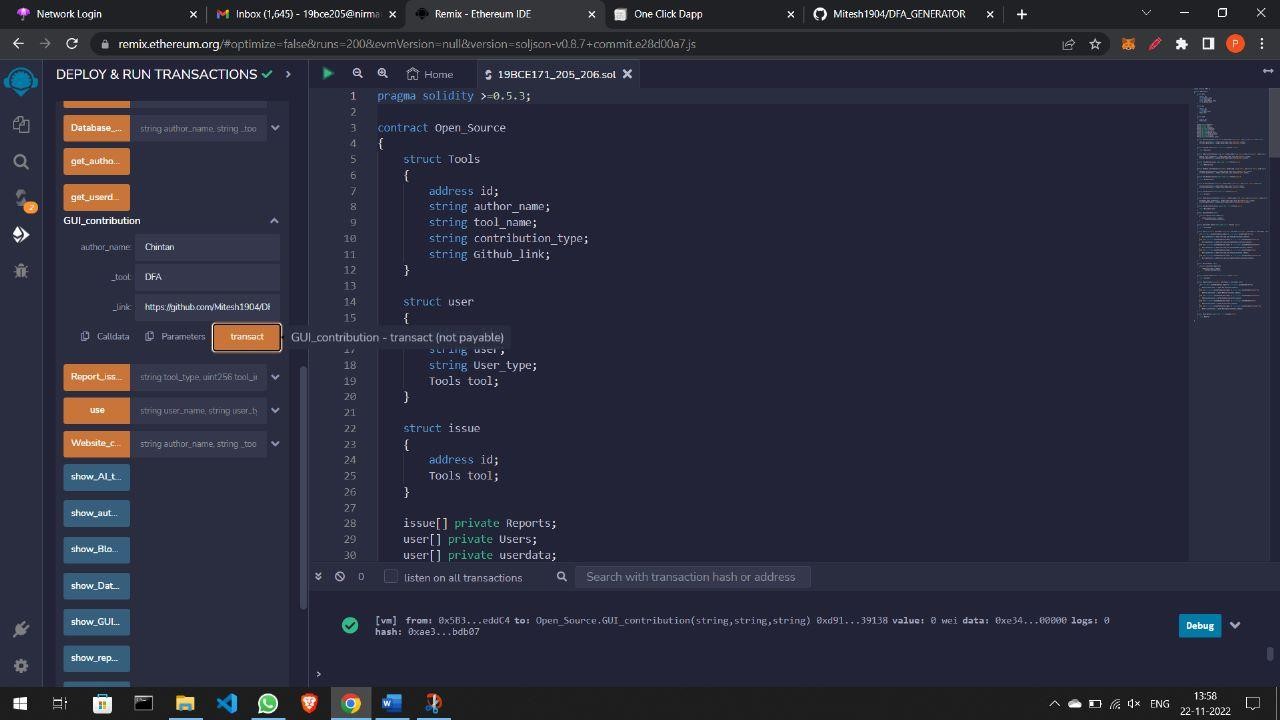
}

Output:

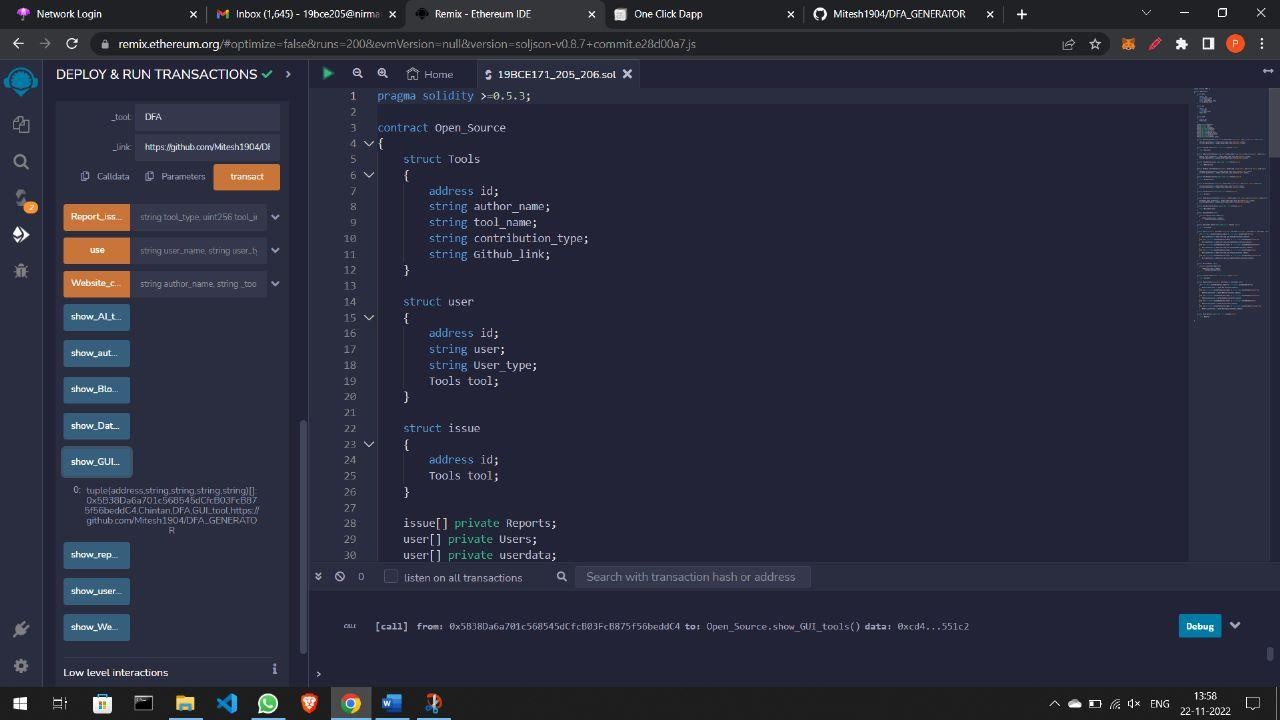
**Output:**

Deployed Smart Contract for Blockchain Based Open-Source Tools.

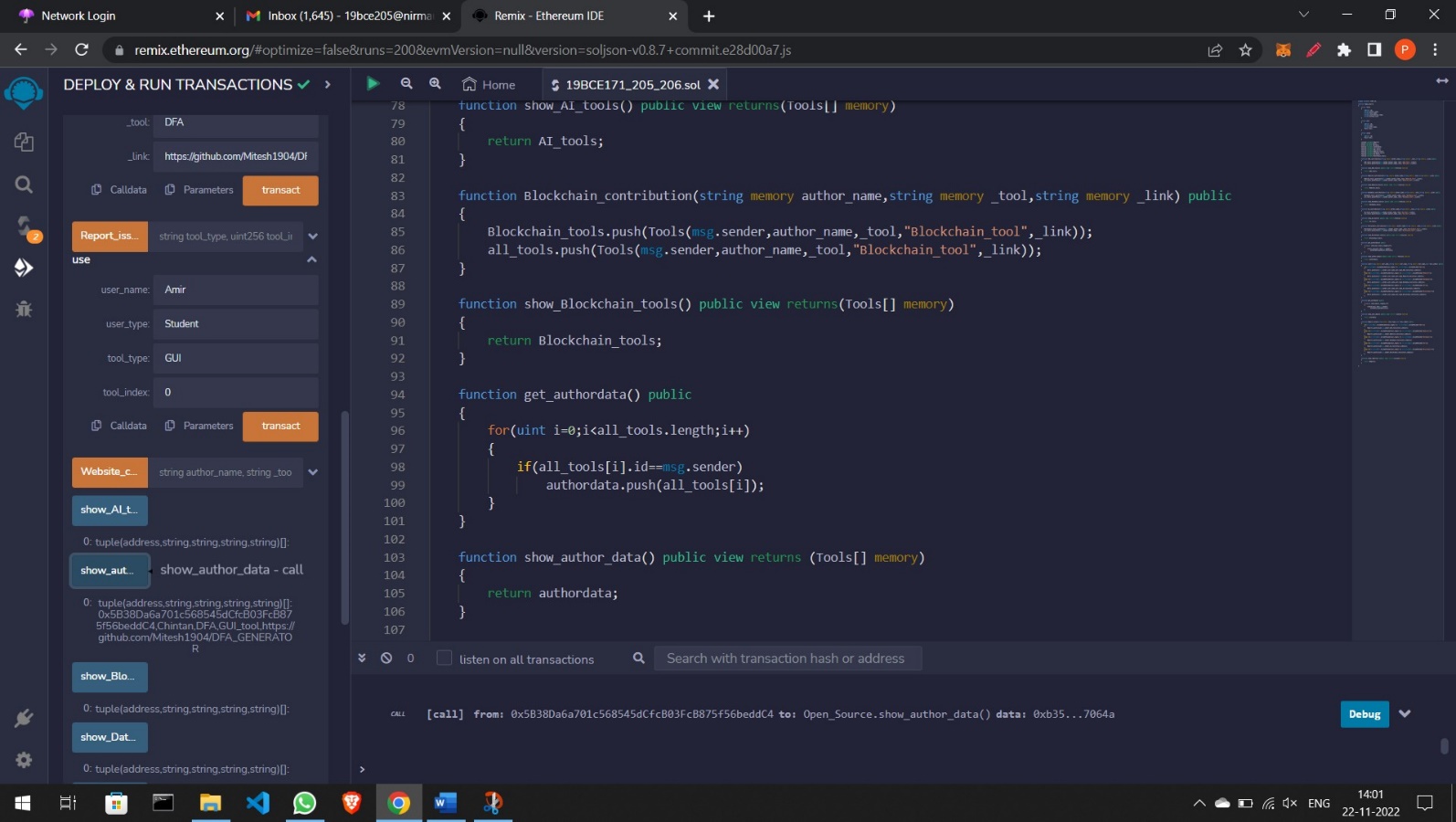
New GUI\_contribution Added



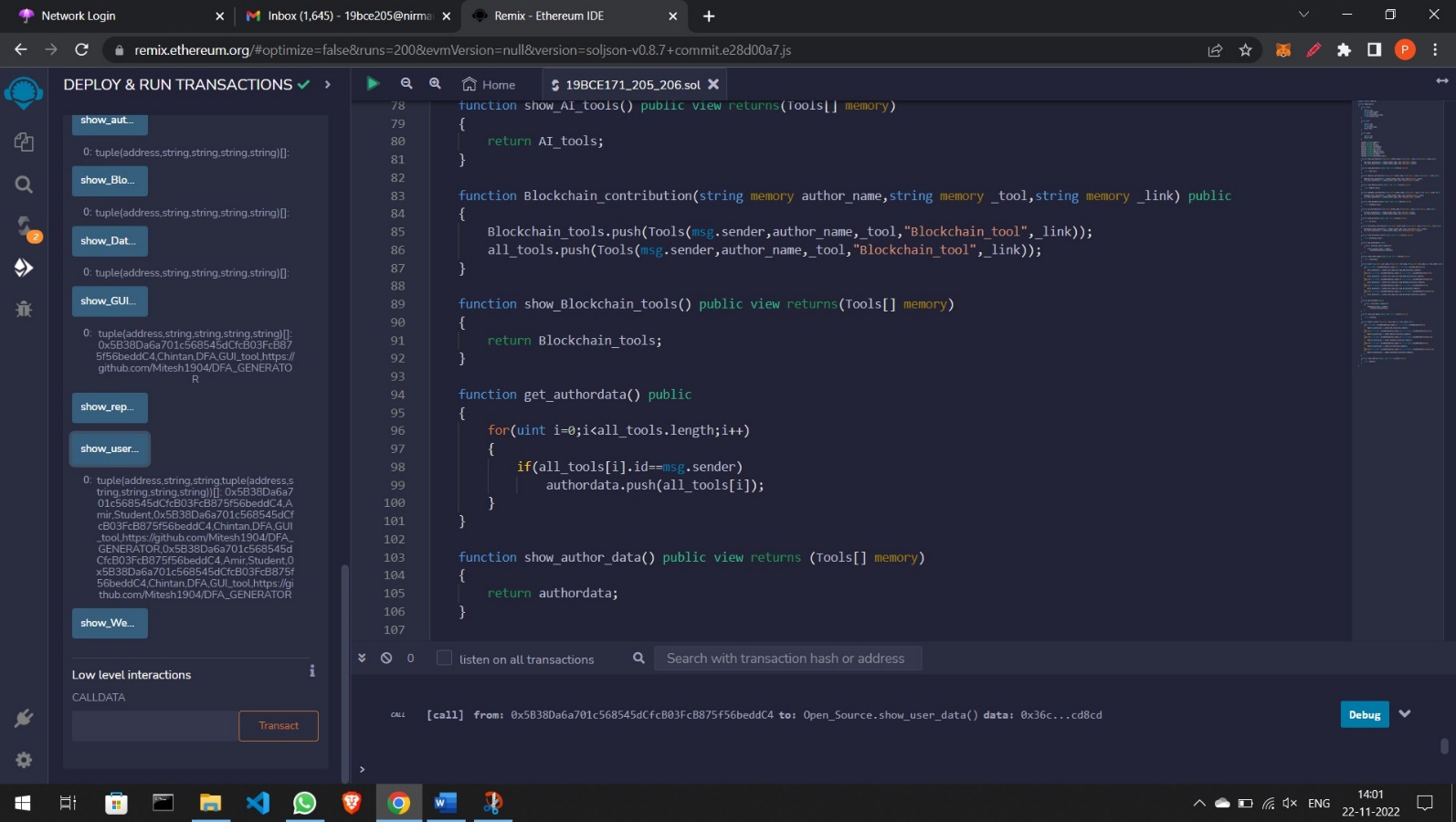
Show GUI Contribution:



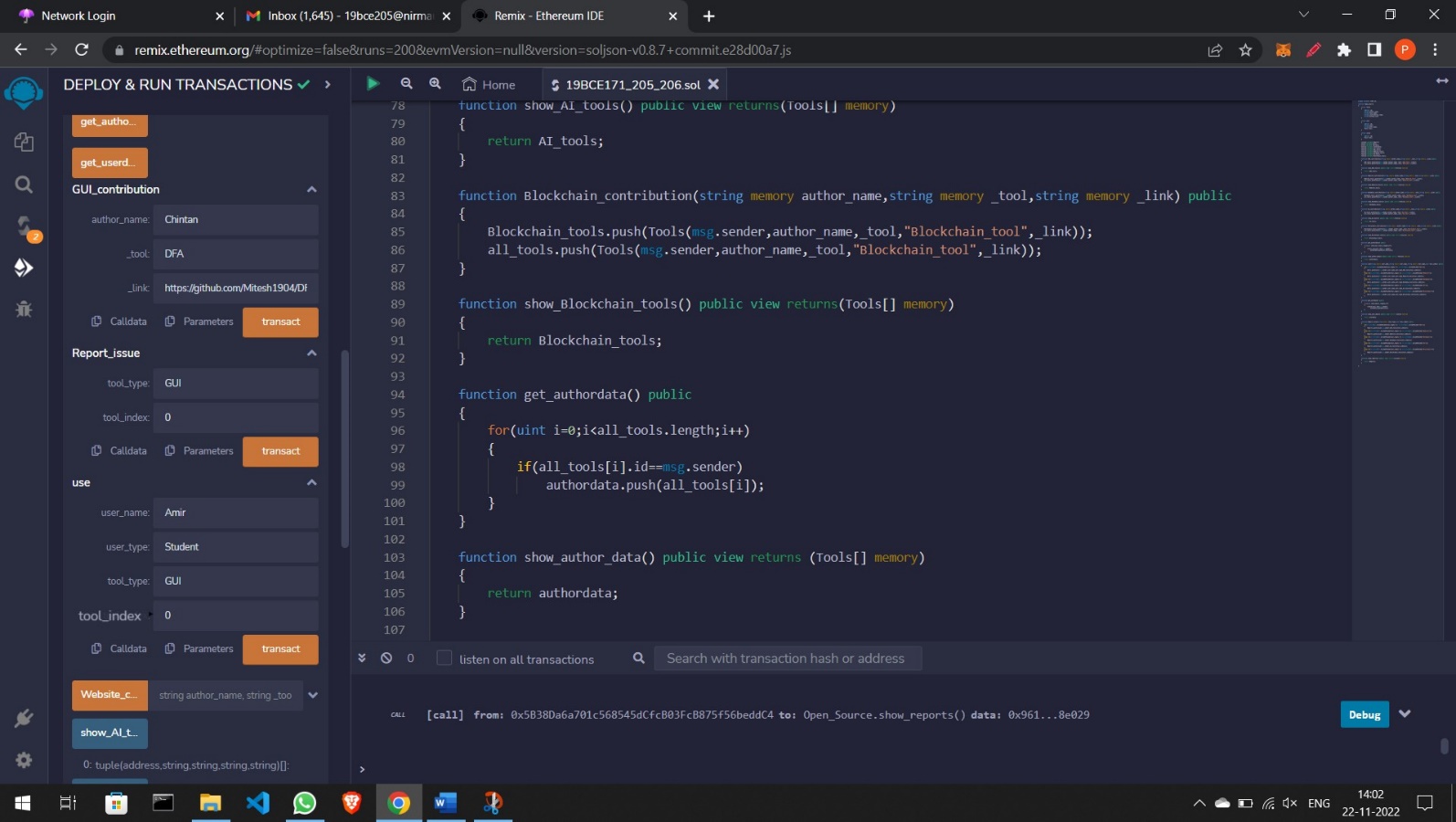
New User added:



Show user:



**Report Issue:**



**Show Report Issue:**

