WRITE UP

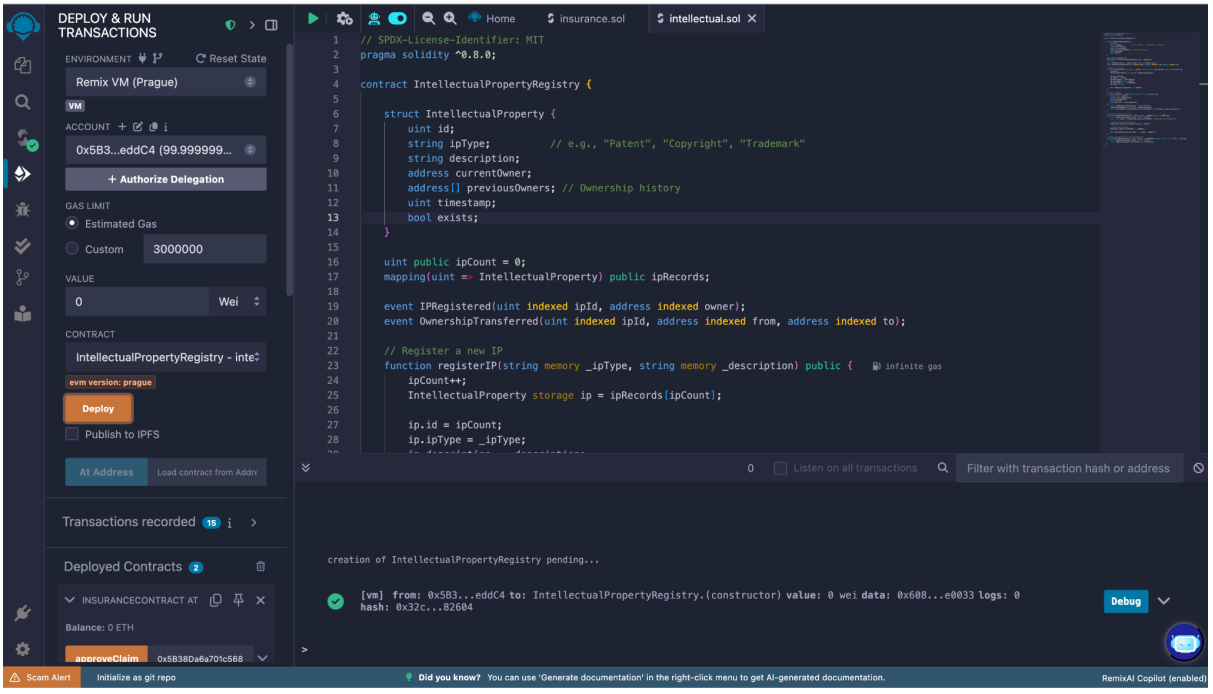
This project is a smart contract-based Intellectual Property (IP) Registry system developed in Solidity. It allows creators and innovators to securely register, verify, and transfer ownership of intellectual property rights — such as patents, trademarks, and copyrights — using the transparency and immutability of blockchain technology.

PROBLEM STATEMENT

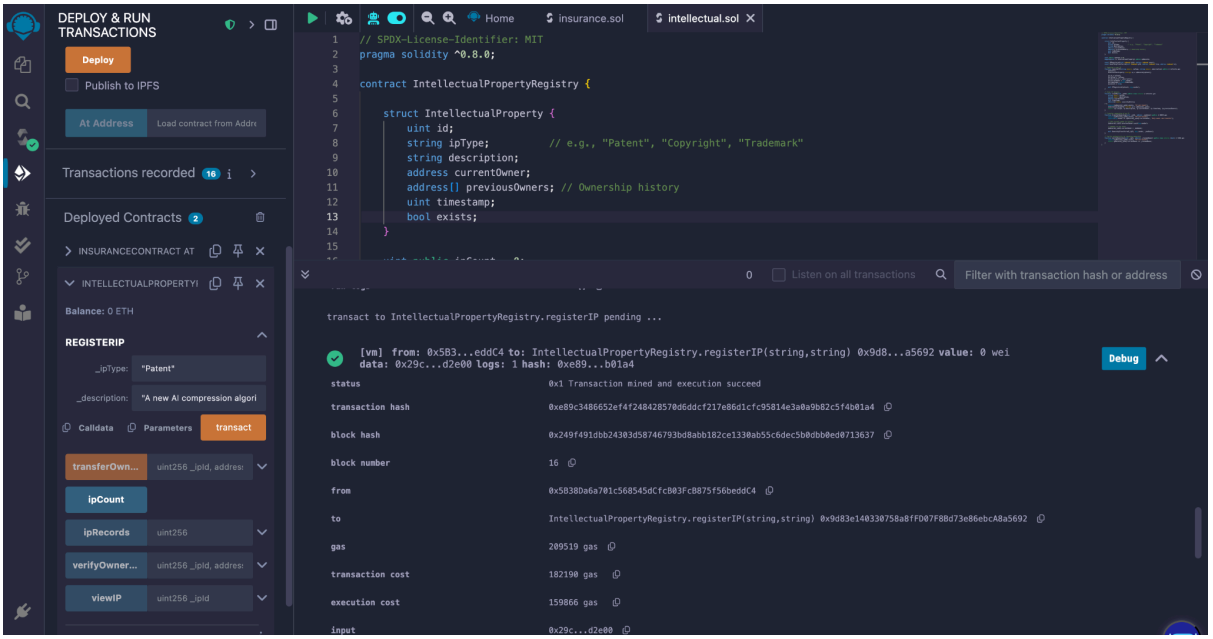
Intellectual property (IP) rights are critical for creators and innovators to protect their inventions, creations, and designs. However, the current systems for registering and managing IP rights often face challenges related to proof of ownership, preventing unauthorized use, and ensuring transparency in transactions. Blockchain technology offers a decentralized and secure platform to address these issues effectively.

Screenshots:

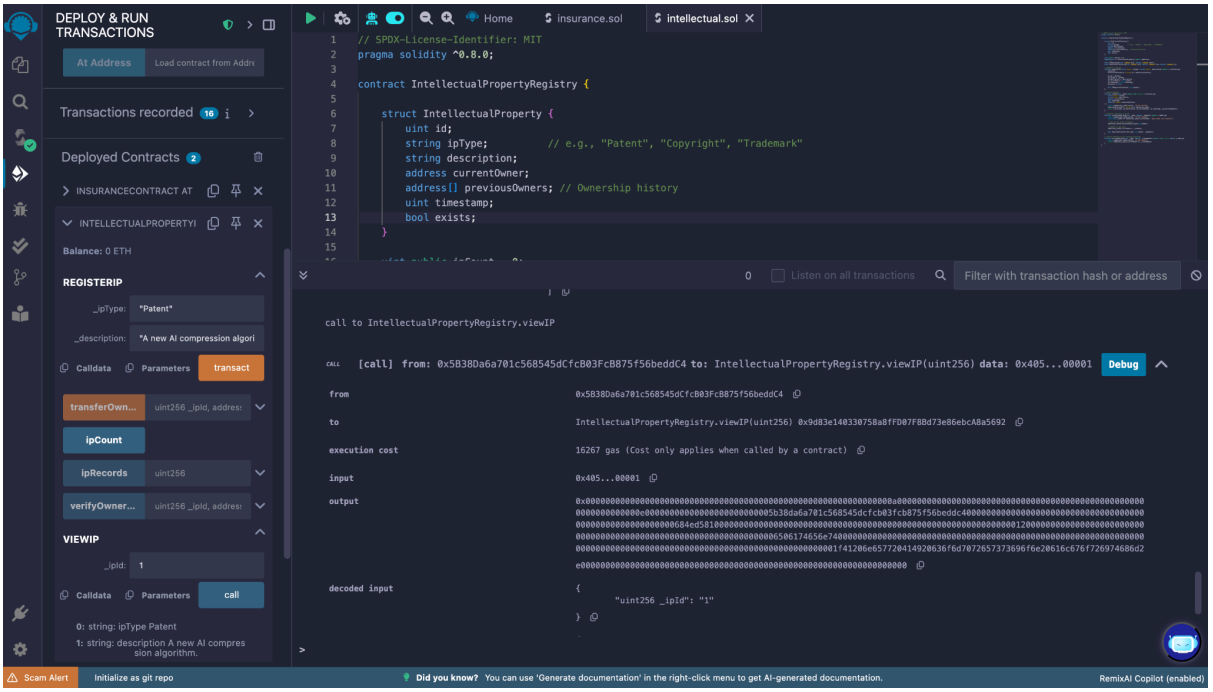
1.CODE DEPLOYEMENT



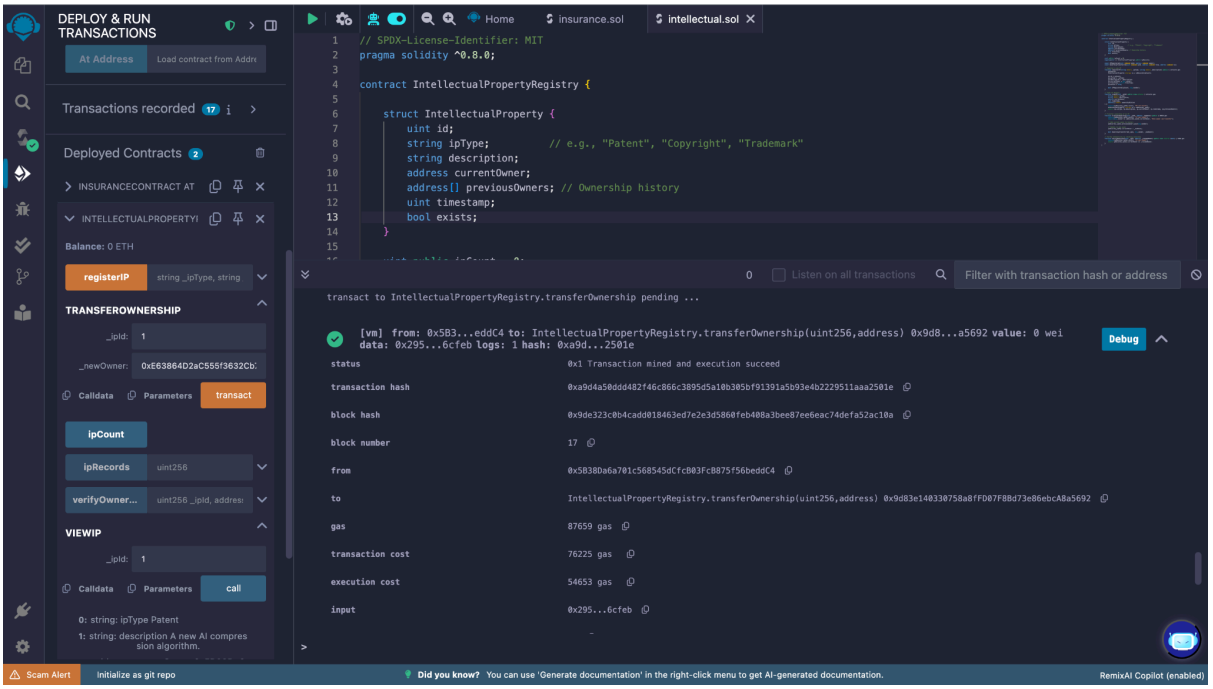
2.Register IP



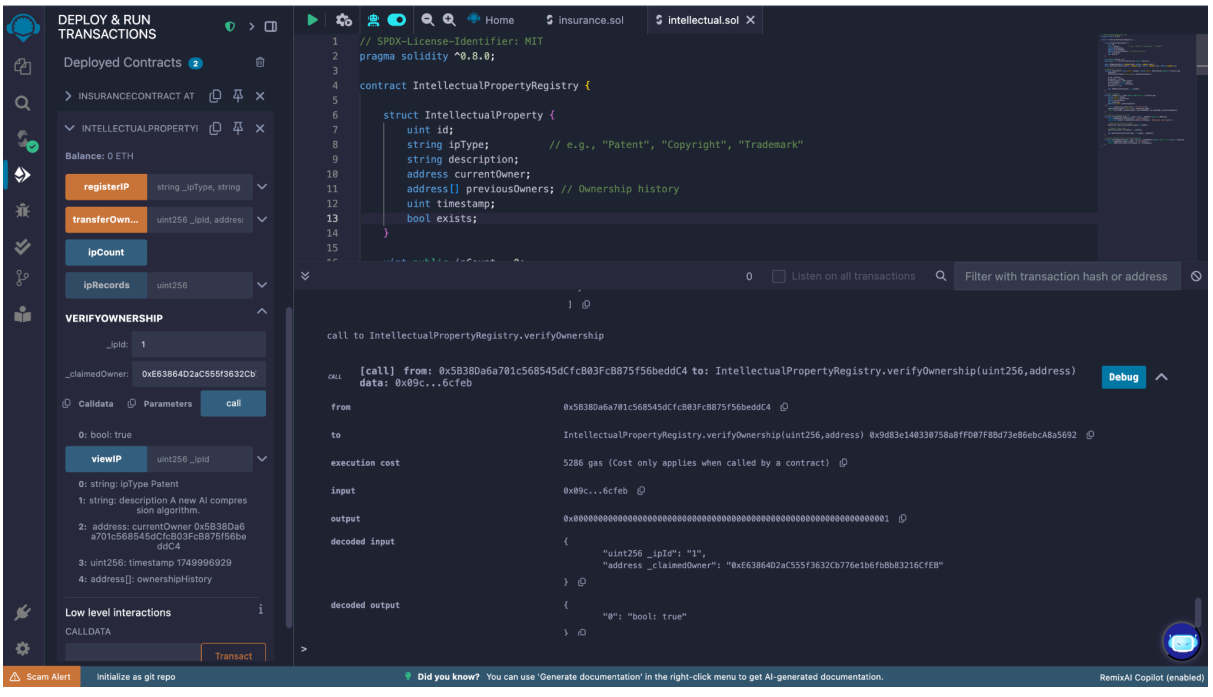
3. View IP



4. Transfer Ownership



5. Verify Ownership



SOURCE CODE:

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


contract IntellectualPropertyRegistry {
```

```
    struct IntellectualProperty {
        uint id;
        string ipType;          // e.g., "Patent", "Copyright", "Trademark"
        string description;
        address currentOwner;
        address[] previousOwners; // Ownership history
        uint timestamp;
        bool exists;
    }

    uint public ipCount = 0;
    mapping(uint => IntellectualProperty) public ipRecords;

    event IPRegistered(uint indexed ipId, address indexed owner);
    event OwnershipTransferred(uint indexed ipId, address indexed from, address
indexed to);

    // Register a new IP
    function registerIP(string memory _ipType, string memory _description) public {
        ipCount++;
        IntellectualProperty storage ip = ipRecords[ipCount];

        ip.id = ipCount;
        ip.ipType = _ipType;
        ip.description = _description;
        ip.currentOwner = msg.sender;
        ip.timestamp = block.timestamp;
        ip.exists = true;

        emit IPRegistered(ipCount, msg.sender);
    }

    // View IP details
    function viewIP(uint _ipId) public view returns (
        string memory ipType,
        string memory description,
        address currentOwner,
        uint timestamp,
        address[] memory ownershipHistory
    ) {
        require(ipRecords[_ipId].exists, "IP not found");
        IntellectualProperty storage ip = ipRecords[_ipId];
        return (ip.ipType, ip.description, ip.currentOwner, ip.timestamp,
ip.previousOwners);
```

```
    }

    // Transfer ownership of an IP
    function transferOwnership(uint _ipId, address _newOwner) public {
        require(ipRecords[_ipId].exists, "IP not found");
        require(msg.sender == ipRecords[_ipId].currentOwner, "Only owner can
transfer");

        // Add current owner to history
        ipRecords[_ipId].previousOwners.push(msg.sender);

        // Update to new owner
        ipRecords[_ipId].currentOwner = _newOwner;

        emit OwnershipTransferred(_ipId, msg.sender, _newOwner);
    }

    // Verify authenticity by checking ownership
    function verifyOwnership(uint _ipId, address _claimedOwner) public view returns
(bool) {
        require(ipRecords[_ipId].exists, "IP not found");
        return ipRecords[_ipId].currentOwner == _claimedOwner;
    }
}
```