Vivekanand Education Society's Institute of Technology

Department of Computer Engineering

Subject: Blockchain

Semester:7 Class: D17B

Roll No: 31	**Name:Tithi Jhamnani**
Practical No: 6	**Title: Creating Smart Contracts and performing transactions using Solidity and Remix IDE**
DOP:	**DOS:**
Grades:	**LOs Mapped: LO2**
Signature:	

AIM: Creating Smart Contracts and performing transactions using Solidity and Remix IDE
Lab Objectives: To explore Blockchain concepts.
Lab Outcomes (LO): Design Smart Contract using Solidity (LO2)

Task to be performed:

1. Based on the topic selected for the Mini Project draft at least 2 smart contracts that will ensure the working of the proposed system.
2. Deploy the contract and check the working of the same.

Theory:

1. **What is a Smart Contract?**

   A smart contract is a self-executing, self-enforcing contract with the terms of the agreement directly written into code. It runs on a blockchain, a distributed and decentralised ledger technology. Smart contracts are designed to automatically execute and enforce the terms of an agreement or contract when predefined conditions are met. These contracts are tamper-proof and immutable, meaning that once they are deployed on a blockchain, their code and execution cannot be altered or tampered with.

   Key characteristics and concepts related to smart contracts:

   1. Code-Based: Smart contracts are written in programming languages specific to the blockchain platform on which they are deployed, such as Solidity for Ethereum.

   2. Decentralised: Smart contracts are executed on a blockchain network, which is decentralised and maintained by a distributed network of nodes (computers). This eliminates the need for intermediaries like banks or legal authorities.

   3. Trustless: Smart contracts aim to eliminate the need for trust between parties. Trust is placed in the code and the blockchain network's consensus mechanism, ensuring transparency and fairness.

   4. Automation: Once deployed, smart contracts execute automatically when predetermined conditions are met. This automation reduces the risk of human error and ensures timely execution.

   5. Immutable: The code and state of a smart contract are immutable, meaning they cannot be altered once deployed. This ensures the integrity and reliability of the contract.

   6. Transparency: Smart contract code and transaction history are typically visible to all participants on the blockchain, providing transparency and auditability.

7. Security: While smart contracts are designed to be secure, vulnerabilities can exist in their code, leading to potential exploits. Careful code review and testing are essential to ensure the security of smart contracts.

Smart contracts have a wide range of applications, including but not limited to:

- Decentralised finance (DeFi): Smart contracts enable lending, borrowing, trading, and yield farming without the need for traditional financial intermediaries.
- Supply chain management: They can be used to track and verify the authenticity of products throughout the supply chain.
- Voting systems: Smart contracts can be employed to create secure and transparent voting systems.
- Legal agreements: They can automate and enforce legal contracts, such as property transactions or insurance policies.

Ethereum is one of the most well-known blockchain platforms that supports smart contracts, but other blockchain networks like Binance Smart Chain, Cardano, and Polka Dot also offer smart contract capabilities.

2. **Significance of smart Contracts in Ethereum Blockchain**

Smart contracts play a significant role in the Ethereum blockchain ecosystem, and their significance can be understood from several perspectives:

1. Decentralised Applications (DApps): Ethereum is a platform designed for building decentralised applications. Smart contracts serve as the backbone of these applications, enabling developers to create DApps with various functionalities, such as decentralised finance (DeFi), non-fungible tokens (NFTs), decentralized exchanges (DEXs), and more. These applications can operate without central control, censorship, or reliance on intermediaries.

2. Decentralised Finance (DeFi): Ethereum is a central hub for DeFi applications, which have grown significantly in popularity. Smart contracts within DeFi handle functions like lending, borrowing, trading, and yield farming. These financial services are accessible to anyone with an internet connection, regardless of their geographical location, and they often provide higher returns and lower fees compared to traditional financial systems.

3. Tokenization: Ethereum's smart contracts enable the creation of tokens that represent various assets, from cryptocurrencies to real-world assets like real estate or artwork. These tokens can be freely traded on Ethereum's blockchain and have facilitated the emergence of the NFT market.

4. Automation and Trustlessness: Smart contracts eliminate the need for intermediaries, such as banks, lawyers, or escrow services. Transactions are automated and executed based on predefined conditions, reducing the risk of fraud and the reliance on trust between parties.

5. Global Reach: Ethereum is a global blockchain platform. Smart contracts are accessible to anyone with an internet connection, making it possible for individuals and businesses worldwide to participate in its ecosystem.

6. Transparency and Security: Ethereum's blockchain provides transparency as all smart contract code and transaction history are publicly viewable. Additionally, the blockchain's security features, such as immutability and cryptographic verification, enhance trust and reliability.

7. Innovation: Ethereum's flexibility and programmability allow developers to create innovative applications and experiment with new use cases for blockchain technology. This has led to a continuous stream of new projects and ideas within the Ethereum ecosystem.

8. Ethereum Improvement Proposals (EIPs): The Ethereum community can propose and implement changes to the Ethereum network through EIPs. Some EIPs are designed to improve the functionality and efficiency of smart contracts, ensuring that Ethereum remains a dynamic and evolving platform.

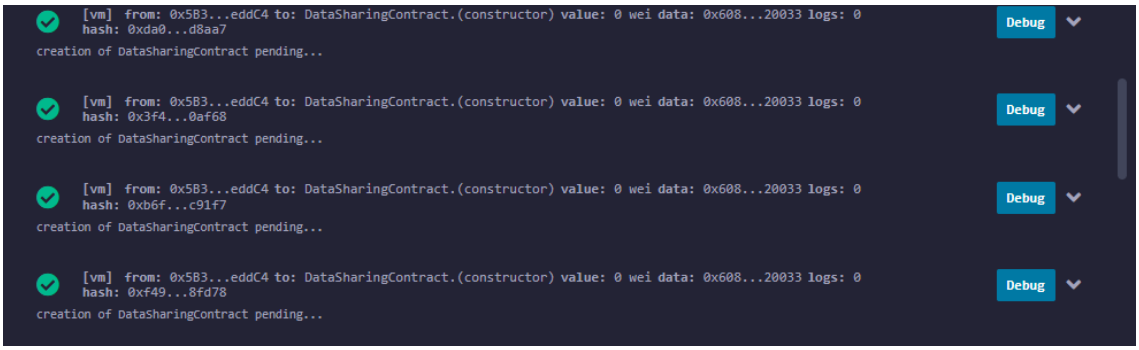
In summary, smart contracts are the foundation of Ethereum's versatility and have enabled the platform to become a pioneering force in the blockchain space. They have revolutionised various industries and continue to drive innovation, making Ethereum a focal point for blockchain development and adoption.

Conclusion:

In conclusion, smart contracts are a revolutionary concept in the blockchain world, and they hold immense significance, particularly within the Ethereum blockchain ecosystem. These self-executing and trustless contracts, encoded in blockchain-based code, have transformed how we conduct transactions, execute agreements, and build decentralised applications. They have ushered in a new era of transparency, security, and automation, enabling the creation of decentralised finance systems, NFTs, and a wide array of innovative applications. Ethereum, as a pioneer in smart contract technology, has played a central role in this transformation, offering a global platform for developers and users to participate in a borderless, decentralised economy. The impact of smart contracts extends far beyond the blockchain realm, promising to reshape various industries and redefine the way we interact with digital and physical assets in the years to come.

1. Data Sharing Smart Contract:

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

contract DataSharingContract {
    address public owner;
    mapping(address => bool) public authorizedResearchers;
    mapping(bytes32 => bool) public sharedData;

    event DataShared(address indexed researcher, bytes32 dataHash);

    constructor() {
        owner = msg.sender;
    }

    modifier onlyOwner() {
        require(msg.sender == owner, "Only the owner can perform this action");
        _;
    }

    modifier onlyAuthorized() {
        require(authorizedResearchers[msg.sender], "You are not authorized to share data");
        _;
    }

    function authorizeResearcher(address researcher) public onlyOwner {
        authorizedResearchers[researcher] = true;
    }

    function revokeAuthorization(address researcher) public onlyOwner {
        authorizedResearchers[researcher] = false;
    }

    function shareData(bytes32 dataHash) public onlyAuthorized {
        sharedData[dataHash] = true;
        emit DataShared(msg.sender, dataHash);
    }
}
```

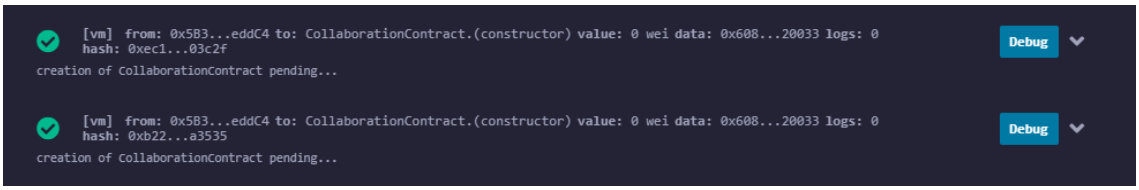

[vm] from: 0x5B3...eddC4 to: DataSharingContract.(constructor) value: 0 wei data: 0x608...20033 logs: 0
hash: 0xda0...d8aa7
Debug
creation of DataSharingContract pending...
[vm] from: 0x5B3...eddC4 to: DataSharingContract.(constructor) value: 0 wei data: 0x608...20033 logs: 0
hash: 0x3f4...0af68
Debug
creation of DataSharingContract pending...
[vm] from: 0x5B3...eddC4 to: DataSharingContract.(constructor) value: 0 wei data: 0x608...20033 logs: 0
hash: 0xb6f...c91f7
Debug
creation of DataSharingContract pending...
[vm] from: 0x5B3...eddC4 to: DataSharingContract.(constructor) value: 0 wei data: 0x608...20033 logs: 0
hash: 0xf49...8fd78
Debug
creation of DataSharingContract pending...

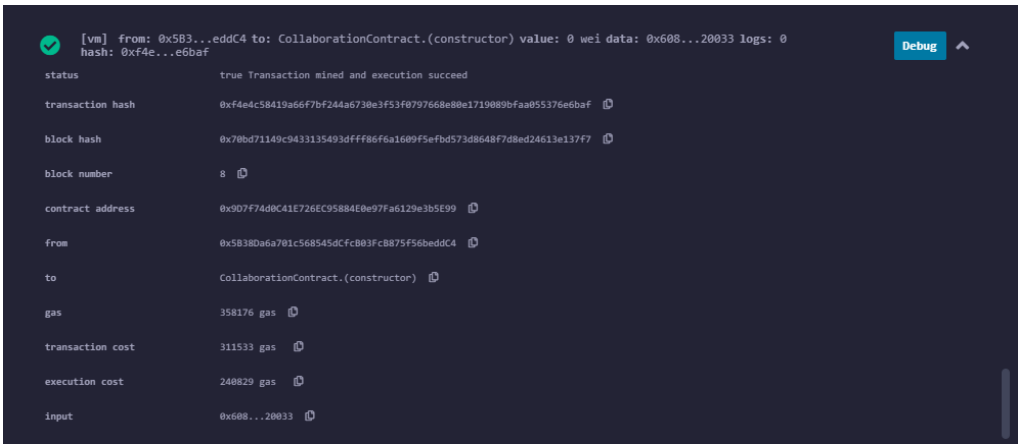

[vm] from: 0x5B3...eddC4 to: DataSharingContract.(constructor) value: 0 wei data: 0x608...20033 logs: 0
hash: 0xf9e...ea4a4
status true Transaction mined and execution succeed
transaction hash 0xf9eee5df25a390b36ef6b5a341f937cfd8ed5afccaaa4066afcbdcd410bea4a4
block hash 0x768355e35e7413333e84e51b93f0eed2f32c3ce1ce27efc88680273b1df5ce2f
block number 5
contract address 0x7EF2e0048f5bAeDe046f6BF797943daF4ED8CB47
from 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
to DataSharingContract.(constructor)
gas 554265 gas
transaction cost 482093 gas
execution cost 399977 gas
input 0x608...20033
decoded input {}

2. Collaboration Smart Contract:

solidity

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

contract CollaborationContract {
    address public owner;
    mapping(bytes32 => uint256) public collaborationProposals;

    event ProposalCreated(bytes32 proposalId, uint256 proposalValue);

    constructor() {
        owner = msg.sender;
    }

    modifier onlyOwner() {
        require(msg.sender == owner, "Only the owner can perform this action");
        _;
    }

    function createProposal(bytes32 proposalId, uint256 proposalValue) public onlyOwner {
        collaborationProposals[proposalId] = proposalValue;
        emit ProposalCreated(proposalId, proposalValue);
    }
}
```





These contracts are simplistic and lack many important features and security checks. In a real-world scenario, you would need to consider:

- Access control and permission management.
- Data encryption and privacy measures for medical data.
- Mechanisms for researchers to request access to data.
- Handling collaboration proposals with more complexity.
- Compliance with data privacy and healthcare regulations.

It's highly recommended to consult with blockchain and smart contract developers, legal experts, and domain specialists when designing and implementing smart contracts for medical research data sharing and collaboration to ensure security and compliance with regulations.