Smart Contract Viva Questions & Detailed Answers

BT 1: Bank Account Smart Contract (Deposit, Withdraw, Show Balance)

1. What is the purpose of creating a Bank Account Smart Contract?

A Bank Account Smart Contract simulates basic banking operations on the blockchain. It allows users to deposit, withdraw, and check balance without depending on any third-party bank. This ensures transparency, security, and decentralization. All transactions are recorded permanently on the Ethereum blockchain.

2. Which data type is used to store the balance and why?

The 'uint' (unsigned integer) data type is used for storing balance. Since balance cannot be negative, 'uint' is suitable and also optimizes storage on the blockchain.

3. Explain the deposit function in your contract.

The 'deposit()' function is defined as 'function deposit() public payable'. The 'payable' keyword allows the function to accept Ether from the sender. When a user calls this function with some Ether, the value is added to their account balance stored in the mapping.

4. Explain the withdraw function and how it ensures security.

The 'withdraw()' function first checks whether the user has enough balance using the 'require' statement. If true, it transfers Ether back to the user's address using 'payable(msg.sender).transfer(amount)'. This prevents unauthorized or excess withdrawals.

5. What is the purpose of the 'payable' keyword in Solidity?

The 'payable' keyword allows a function or an address to receive Ether. Without marking a function as 'payable', it cannot handle incoming Ether transactions.

6. How are multiple customer accounts managed in your contract?

Multiple customers are handled using a mapping: 'mapping(address => uint) balances;'. Each customer's Ethereum address acts as a key, mapping to their balance. This makes account management unique and secure.

7. What happens when a transaction fails?

If a transaction fails due to conditions in 'require' or 'assert', it reverts. All state changes are rolled back, but the gas used up to that point is consumed.

8. How did you deploy the smart contract?

The smart contract was written in Solidity using Remix IDE. MetaMask wallet was connected to Remix, and the contract was deployed on a test network like Goerli or Sepolia using test Ether.

9. What is the function of EVM (Ethereum Virtual Machine)?

The Ethereum Virtual Machine (EVM) executes the smart contract bytecode in a decentralized environment. It ensures deterministic execution and consistency across all Ethereum nodes.

10. What is the difference between msg.sender and tx.origin?

'msg.sender' is the immediate account (user or contract) that calls a function, while 'tx.origin' refers to the original external account that initiated the entire transaction chain.

BT 2: Student Data Smart Contract (Structures and Arrays)

1. What is the role of structures in Solidity?

Structures allow grouping of related variables into a single data type. In this contract, the 'Student' structure holds student details like roll number, name, and marks. It improves readability and organizes data effectively.

2. How are arrays used with structures?

An array of 'Student' structures is used to store multiple student records. For example: 'Student[] public students;'. This allows dynamic addition and retrieval of student data.

3. How can you add and retrieve student data?

Adding: 'students.push(Student(rollNo, name, marks));'. Retrieving: a function returns details based on index. Example: 'function getStudent(uint index) public view returns (...)'.

4. What is the difference between dynamic and fixed arrays?

Dynamic arrays can grow or shrink in size at runtime, while fixed arrays have a predefined length. For student records, dynamic arrays are preferred because the number of students can vary.

5. How do you ensure unique student entries?

By using a unique identifier such as 'rollNo' or maintaining a mapping 'mapping(uint => Student)' to avoid duplicate records.

6. How is memory managed in Solidity?

Solidity has 'memory' for temporary data within functions and 'storage' for permanent blockchain data. Student records are stored in 'storage' since they need to persist after execution.

7. How is gas consumption affected in your contract?

Every addition or modification consumes gas. The more data stored or complex operations used, the higher the gas consumption. Simpler data structures reduce gas cost.

8. How can only the teacher add student records?

By restricting access using modifiers. Example: 'modifier onlyOwner { require(msg.sender == owner, 'Not Authorized'); _; }'. This ensures only the deployer (teacher) can add students.

9. How did you verify deployment and gas fees?

After deploying on a test network like Sepolia, the transaction hash was checked on the Sepolia Etherscan. It showed gas used, transaction fee, and status.

10. What are the benefits of using blockchain for student data management?

Blockchain ensures immutability, transparency, and prevents data tampering. Each record is permanently stored and can be verified anytime without intermediaries.

Common Questions for Both Assignments

1. What is a Smart Contract?

A Smart Contract is a self-executing digital agreement written in Solidity that runs on a blockchain. It contains predefined conditions and automatically executes actions once those conditions are met.

2. What is the role of MetaMask?

MetaMask acts as a bridge between the Ethereum blockchain and your browser. It manages private keys, allows sending transactions, and connects to decentralized apps (DApps).

3. What is gas and why is it important?

Gas represents the computational cost of executing operations on Ethereum. It prevents infinite loops and rewards miners for validating transactions. The total fee = Gas Used × Gas Price.

4. What is the difference between Mainnet and Testnet?

Mainnet is the live Ethereum network using real Ether. Testnets (Goerli, Sepolia) are for testing purposes using free test Ether, allowing developers to deploy safely.

5. What is ABI and its use?

ABI (Application Binary Interface) defines how to interact with a smart contract's functions and data. It's required for frontend or other systems to call contract functions.

6. How do you ensure security in smart contracts?

Security is ensured by using access control (modifiers), validating inputs, preventing reentrancy, and avoiding untrusted external calls.

7. How does blockchain immutability help your project?

Once data or contracts are stored on the blockchain, they cannot be modified. This provides transparency and ensures that no one can alter the transaction history.

8. How can you upgrade a contract?

Smart contracts are immutable once deployed. Upgrades can be done by deploying a new version or using proxy contracts to separate logic and data.