Experiment No-4

Title: Development and Deployment of a Smart Contract using Solidity.

Aim: To understand the fundamentals of blockchain and smart contracts, and to develop and deploy a simple smart contract on the Ethereum blockchain using the Solidity programming language..

Theory:

- ### Smart Contracts:

A smart contract is a self-executing contract with the terms of the agreement directly written into code. These contracts are deployed on a blockchain network, and once deployed, they are immutable, meaning they cannot be altered. Smart contracts run automatically when the specified conditions are met, ensuring trustless and transparent transactions.

- ### Solidity:

Solidity is the most widely used programming language for developing smart contracts on the Ethereum blockchain. It is a statically typed language designed specifically for writing smart contracts. Solidity code is compiled into bytecode that can be deployed and executed on the Ethereum Virtual Machine (EVM).

- ### Ethereum Blockchain:

Ethereum is a decentralized platform that runs smart contracts and decentralized applications (dApps). Ethereum smart contracts use Ether (ETH) as the internal cryptocurrency for transaction fees and rewards.

Key Concepts in Solidity:

Smart Contract Deployment: The process of deploying the contract to the blockchain.

State Variables: Variables that hold data and are stored on the blockchain.

Functions: Used to define the actions or logic in a smart contract.

Gas: Gas is the computational fee required to execute transactions and operations on the Ethereum network.

Public and Private Functions: Public functions can be called externally, while private ones are restricted to the contract itself.

Steps of Execution:

- Set up development environment:
- Install and configure tools like Node.js, Truffle Suite, and Ganache for local blockchain simulation.
- Use Remix IDE for smart contract development and testing in the browser.
- Write the smart contract in Solidity.
- Compile and deploy the contract to a local Ethereum network using Truffle or directly on Remix.

- Interact with the deployed contract using web interfaces (JavaScript or Truffle scripts).
- Test the contract on the Ethereum test network (e.g., Rinkeby or Ropsten) for real-world simulation before deploying to the main network.

Stepwise Procedure:

1. Setting Up Development Environment:

- Install Node.js and npm (Node Package Manager) from Node.js official site
- Install Truffle by running the command npm install -g truffle.
- Install Ganache to simulate a local Ethereum blockchain from Ganache official site.
- Optionally, use Remix IDE for quick contract development directly in the browser (Remix IDE).

2. Writing the Smart Contract:

Create a new folder for the project and initialize it with Truffle:

```
mkdir SmartContractExample
cd SmartContractExample
truffle init
```

Create a new Solidity file in the contracts/ directory (e.g., SimpleStorage.sol).

3. Writing the Solidity Code (Smart Contract):

Example: SimpleStorage.sol:

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
contract SimpleStorage {
    uint256 storedData;
    // Event to log data changes
    event DataStored(uint256 data);
    // Store a new value
    function set(uint256 x) public {
        storedData = x;
        emit DataStored(storedData);
    }
    // Retrieve the stored value
    function get() public view returns (uint256) {
        return storedData;
    }
}
```

4. Compile the Contract:

Using Truffle:

truffle compile

5. Deploy the Contract:

Create a migration script under the migrations/ folder, such as 2_deploy_contracts.js:

```
const SimpleStorage = artifacts.require("SimpleStorage");
module.exports = function (deployer) {
  deployer.deploy(SimpleStorage);
};
```

Deploy to the local Ganache network:

truffle migrate

6. Interacting with the Deployed Contract:

Use Truffle's console to interact with the deployed contract:

truffle console

In the console, interact with the contract:

```
let instance = await SimpleStorage.deployed();
await instance.set(100);  // Store value 100
let value = await instance.get();  // Retrieve stored value
console.log(value.toString());  // Should output 100
```

7. Deploy to Test Network:

1. Configure Truffle for test networks (e.g., Rinkeby) by editing truffle-config.js.
2. Deploy using:

truffle migrate --network rinkeby

Key Points :

- Solidity is the most commonly used language for developing smart contracts on Ethereum.
- A smart contract is immutable once deployed on the blockchain.
- Gas fees are associated with every transaction (execution of smart contracts).
- Truffle Suite is a popular framework for developing, testing, and deploying smart contracts.
- A migration script is used in Truffle to deploy smart contracts to the blockchain.
- Events can be used to log and track changes to smart contract state.

Conclusion :

This lab successfully demonstrated how to develop and deploy a smart contract using Solidity and the Truffle framework. We created a simple storage contract that allows storing and retrieving values on the Ethereum blockchain. The simulation included both local deployment using Ganache and real deployment to an Ethereum test network. Smart contracts offer immense potential in blockchain applications due to their transparency, security, and automation features.

Viva Questions:

1. What is a smart contract, and how does it differ from a traditional contract?
2. Explain the role of pragma in Solidity.
3. What are state variables in Solidity?
4. What is the purpose of gas in Ethereum?
5. What is the purpose of the event keyword in Solidity?
6. Explain the difference between public, private, and internal functions in Solidity.
7. What is the role of deployer.deploy() in the Truffle migration script?
8. How do you test a smart contract before deploying it to the mainnet?
9. What is the Ethereum Virtual Machine (EVM)?
10. What is a migration script in Truffle?