**Block\_Chain\_Solidity\_App\_Setup**

**1. Install Prerequisites**

**a. Node.js**

* Download and install **Node.js (LTS version)** from [Node.js official website](https://nodejs.org/).
* Verify installation:

***node -v***

***npm -v***

**b. Git**

* Install Git from [Git official website](https://git-scm.com/).
* Verify installation:

***git --version***

**c. Visual Studio Code**

* Download and install [Visual Studio Code](https://code.visualstudio.com/).
* Install the following VS Code extensions:
  + **Solidity**: For smart contract syntax highlighting and IntelliSense.
  + **Prettier**: For code formatting.
  + **ESLint**: For JavaScript linting.

**2. Install Blockchain Development Tools**

**a. Truffle Suite (Framework for Solidity)**

* Install Truffle globally using npm:

***npm install -g truffle***

* Verify installation:

truffle version

**c. Ganache (Local Blockchain)**

* Download and install Ganache from Truffle Suite.
* Ganache simulates a blockchain for testing purposes.
* Start Ganache and use its RPC endpoint (e.g., http://127.0.0.1:7545) in your projects.

**3. Configure Solidity Development**

**a. Install Solidity Compiler**

* Install the Solidity compiler using npm:

***npm install -g solc***

* Alternatively, use the VS Code Solidity extension, which includes the compiler.

**b. Set Up a Smart Contract**

* Create a new Solidity file, e.g., SimpleStorage.sol:

solidity

***// SPDX-License-Identifier: MIT***

***pragma solidity ^0.8.0;***

***contract SimpleStorage {***

***uint public storedData;***

***function set(uint x) public {***

***storedData = x;***

***}***

***function get() public view returns (uint) {***

***return storedData;***

***}***

***}***

**c. Compile Smart Contract**

* Using Truffle:

***truffle compile***

**4. Set Up JavaScript for Blockchain Interaction**

**a. Install Web3.js or Ethers.js**

* Install Web3.js (for Ethereum interaction):

***npm install web3***

* Or Ethers.js (a modern alternative):

***npm install ethers***

**b. Example: Interact with Smart Contract**

* Create a JavaScript file, e.g., interact.js:

javascript

***const Web3 = require('web3');***

***const web3 = new Web3('http://127.0.0.1:7545'); // Ganache RPC endpoint***

***const contractABI = [***

***// Your contract's ABI***

***];***

***const contractAddress = '0xYourContractAddress';***

***const contract = new web3.eth.Contract(contractABI, contractAddress);***

***async function main() {***

***const accounts = await web3.eth.getAccounts();***

***console.log('Accounts:', accounts);***

***// Set a value***

***await contract.methods.set(42).send({ from: accounts[0] });***

***// Get the stored value***

***const value = await contract.methods.get().call();***

***console.log('Stored Value:', value);***

***}***

***main();***

* Run the script:

***node interact.js***

**5. Optional Tools**

* **Metamask**: Install the [Metamask browser extension](https://metamask.io/) to manage accounts and interact with the blockchain.
* **Infura**: Use [Infura](https://infura.io/) for connecting to the Ethereum mainnet or testnets.

**6. Teach Smart Contract Development**

1. **Explain Basics**:
   * Blockchain basics: What is Ethereum, Gas, and Smart Contracts.
   * Solidity language fundamentals.
2. **Practical Examples**:
   * Create and deploy smart contracts.
   * Use JavaScript to interact with contracts.
3. **Deploy Contracts**:
   * Use Truffle or Hardhat to deploy contracts to testnets like Rinkeby or Sepolia.
   * Example deployment command for Truffle:

***truffle migrate --network rinkeby***

1. **Advanced Topics**:
   * ERC-20 and ERC-721 tokens.
   * Security best practices in Solidity.