FRONTEND

Date	**20 October 2023**
Team ID	**NM2023TMID02250**
Project Name	**Project- Farmer Insurance Chain**
Maximum Marks	**4 Marks**

INTERACT WITH FRONTEND FOR ALL FUNCTIONALITIES:

To interact with the smart contract for a Farmer Insurance Chain from a frontend application, you can use web3.js or ethers.js (for Ethereum) or the appropriate library for the blockchain you're using. Here's a basic example of how to interact with the previously mentioned smart contract using web3.js and a simple HTML/JavaScript frontend:

1. **Setting Up Your HTML File**:

Create an HTML file (e.g., `index.html`) to build a basic user interface for interacting with the smart contract.

```html
<!DOCTYPE html>
<html>
<head>
    <title>Farmer Insurance Chain</title>
</head>
<body>
    <h1>Farmer Insurance Chain</h1>

    <button onclick="purchasePolicy()">Purchase Policy</button>
```

```
    <button onclick="processClaim()">Process Claim</button>
    <button onclick="withdrawFunds()">Withdraw
Funds</button>

    <div>
        <h2>Contract Info:</h2>
        <p>Owner: <span id="owner"></span></p>
        <p>Premium Amount: <span
id="premiumAmount"></span></p>
    </div>
</body>

<script src="https://cdn.jsdelivr.net/npm/web3@1.0.0-
beta.37/dist/web3.min.js"></script>
<script src="app.js"></script>
</html>
```

2. **Creating the JavaScript File (app.js)**:

In your JavaScript file (e.g., `app.js`), you can use web3.js to interact with the smart contract. Make sure you've initiated a web3 provider (e.g., MetaMask) in your browser.

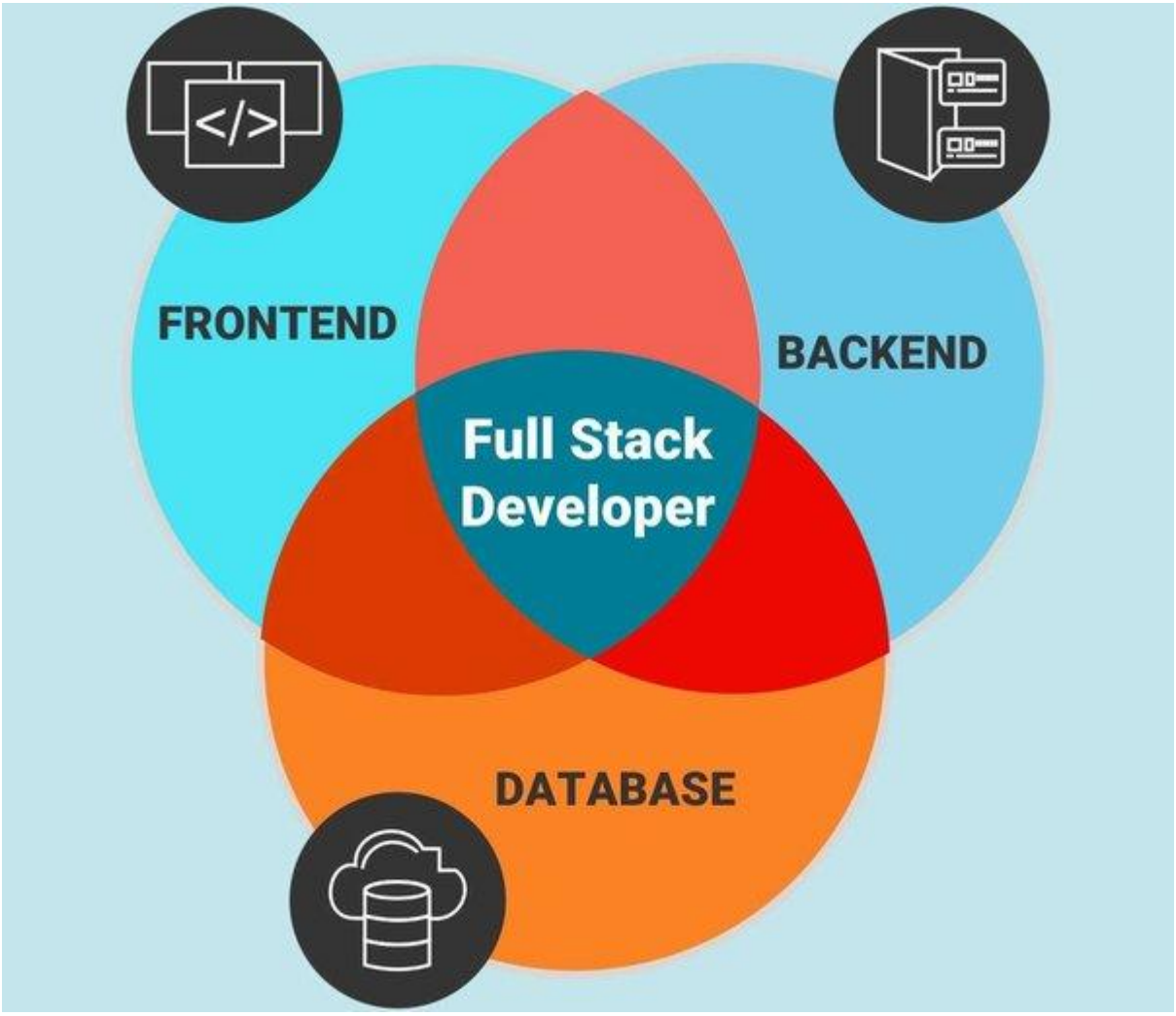
```javascript
// Connect to the smart contract
const contractAddress = 'YOUR_CONTRACT_ADDRESS'; //
Replace with your contract's address
const ABI = [ /* Your contract's ABI */ ];
const web3 = new Web3(Web3.givenProvider);

const farmerInsuranceContract = new web3.eth.Contract(ABI,
contractAddress);

// Retrieve contract data
async function getContractData() {
```

```
    const owner = await
farmerInsuranceContract.methods.owner().call();
    const premiumAmount = await
farmerInsuranceContract.methods.premiumAmount().call();

    document.getElementById('owner').innerText = owner;
    document.getElementById('premiumAmount').innerText =
premiumAmount;
}

getContractData();

// Purchase Policy
async function purchasePolicy() {
    try {
        const accounts = await web3.eth.getAccounts();
        await
farmerInsuranceContract.methods.purchasePolicy().send({ from:
accounts[0], value: web3.utils.toWei('1', 'ether') });
        alert('Policy purchased successfully!');
    } catch (error) {
        console.error(error);
    }
}


// Process Claim
async function processClaim() {
    try {
        const accounts = await web3.eth.getAccounts();
        await
farmerInsuranceContract.methods.processClaim().send({ from:
accounts[0] });
        alert('Claim processed successfully!');
    } catch (error) {
        console.error(error);
    }
}
```

```
// Withdraw Funds (Owner only)
async function withdrawFunds() {
    try {
        const accounts = await web3.eth.getAccounts();
        await
farmerInsuranceContract.methods.withdrawFunds().send({ from:
accounts[0 });
        alert('Funds withdrawn successfully!');
    } catch (error) {
        console.error(error);
    }
}
```

Remember to replace `''YOUR_CONTRACT_ADDRESS''` with the actual address of your deployed Farmer Insurance Chain smart contract, and `''YOUR_CONTRACT_ABI''` with your contract's ABI.

This is a basic example and does not include error handling, security measures, or production-ready features. In a real-world application, you would implement more robust front-end and security measures, such as checking for contract interactions and user authentication. Additionally, ensure that users have a web3 provider (like MetaMask) installed and configured in their browsers.


FRONTEND
BACKEND
Full Stack
Developer
DATABASE