FarmSupplyChain.sol

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

contract FarmProductSupplyChain {
    struct FarmProduct {
        string productId;
        string productDescription;
        string producerName;
        string producerAddress;
        string harvestDate;
        string distributerName;
        string distributerAddress;
        string prodToDistDate;
        string retailerName;
        string retailerAddress;
        string distToRetaDate;
    }

    mapping(string => FarmProduct) private farmProducts;
    event ProductAdded(string productId);
    event OwnershipTransferred(string productId, string newOwner);

    function addNewCa(
        string memory id,
        string memory description,
        string memory producerName,
        string memory producerAddress,
        string memory harvestDate
    ) public {
        require(bytes(farmProducts[id].productId).length == 0, "Product ID already
exists");

        farmProducts[id] = FarmProduct(
            id,
            description,
            producerName,
            producerAddress,
            harvestDate,
            "", "", "", "", "", ""
        );

        emit ProductAdded(id);
    }

    function changeCarOwnership(
        string memory id,
        string memory newOwnerName,
        string memory newOwnerAddress,
        string memory transferDate,
```

```
        bool toDistributor
    ) public {
        require(bytes(farmProducts[id].productId).length != 0, "Product does not
exist");

        if (toDistributor) {
            farmProducts[id].distributerName = newOwnerName;
            farmProducts[id].distributerAddress = newOwnerAddress;
            farmProducts[id].prodToDistDate = transferDate;
        } else {
            farmProducts[id].retailerName = newOwnerName;
            farmProducts[id].retailerAddress = newOwnerAddress;
            farmProducts[id].distToRetaDate = transferDate;
        }

        emit OwnershipTransferred(id, newOwnerName);
    }

    function queryCarById(string memory id) public view returns (FarmProduct memory) {
        require(bytes(farmProducts[id].productId).length != 0, "Product not found");
        return farmProducts[id];
    }
}
```

FarmSupplyChain.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Farm Product Supply Chain</title>
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/web3/1.7.5/web3.min.js"></script>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 20px;
            padding: 20px;
            background-color: #f4f4f4;
        }
        .container {
            max-width: 600px;
            margin: auto;
            padding: 20px;
            background: white;
            border-radius: 8px;
            box-shadow: 0px 0px 10px rgba(0, 0, 0, 0.1);
        }
        input, button {
            width: 100%;
            padding: 10px;
            margin: 5px 0;
            border: 1px solid #ccc;
            border-radius: 5px;
        }
        button {
            background-color: #28a745;
            color: white;
            cursor: pointer;
        }
    </style>
</head>
<body>
    <div class="container">
        <h2>Farm Product Supply Chain</h2>
        <h3>Add New Product</h3>
        <input type="text" id="productId" placeholder="Product ID">
        <input type="text" id="description" placeholder="Description">
        <input type="text" id="producerName" placeholder="Producer Name">
        <input type="text" id="producerAddress" placeholder="Producer Address">
        <input type="text" id="harvestDate" placeholder="Harvest Date">
        <button onclick="addNewProduct()">Add Product</button>
```

```
        <h3>Transfer Ownership</h3>
        <input type="text" id="transferId" placeholder="Product ID">
        <input type="text" id="newOwnerName" placeholder="New Owner Name">
        <input type="text" id="newOwnerAddress" placeholder="New Owner Address">
        <input type="text" id="transferDate" placeholder="Transfer Date">
        <select id="ownerType">
            <option value="distributor">To Distributor</option>
            <option value="retailer">To Retailer</option>
        </select>
        <button onclick="transferOwnership()">Transfer Ownership</button>

        <h3>Query Product</h3>
        <input type="text" id="queryId" placeholder="Product ID">
        <button onclick="queryProduct()">Query</button>
        <p id="result"></p>
    </div>

    <script>
        const contractAddress = "0x94B29A512810629D0dcc5c1EF524fCbb357ED515";
        const contractABI = [
    {
        "inputs": [
            {
                "internalType": "string",
                "name": "id",
                "type": "string"
            },
            {
                "internalType": "string",
                "name": "description",
                "type": "string"
            },
            {
                "internalType": "string",
                "name": "producerName",
                "type": "string"
            },
            {
                "internalType": "string",
                "name": "producerAddress",
                "type": "string"
            },
            {
                "internalType": "string",
                "name": "harvestDate",
                "type": "string"
            }
        ],
        "name": "addNewCa",
        "outputs": [],
        "stateMutability": "nonpayable",
        "type": "function"
    },
```

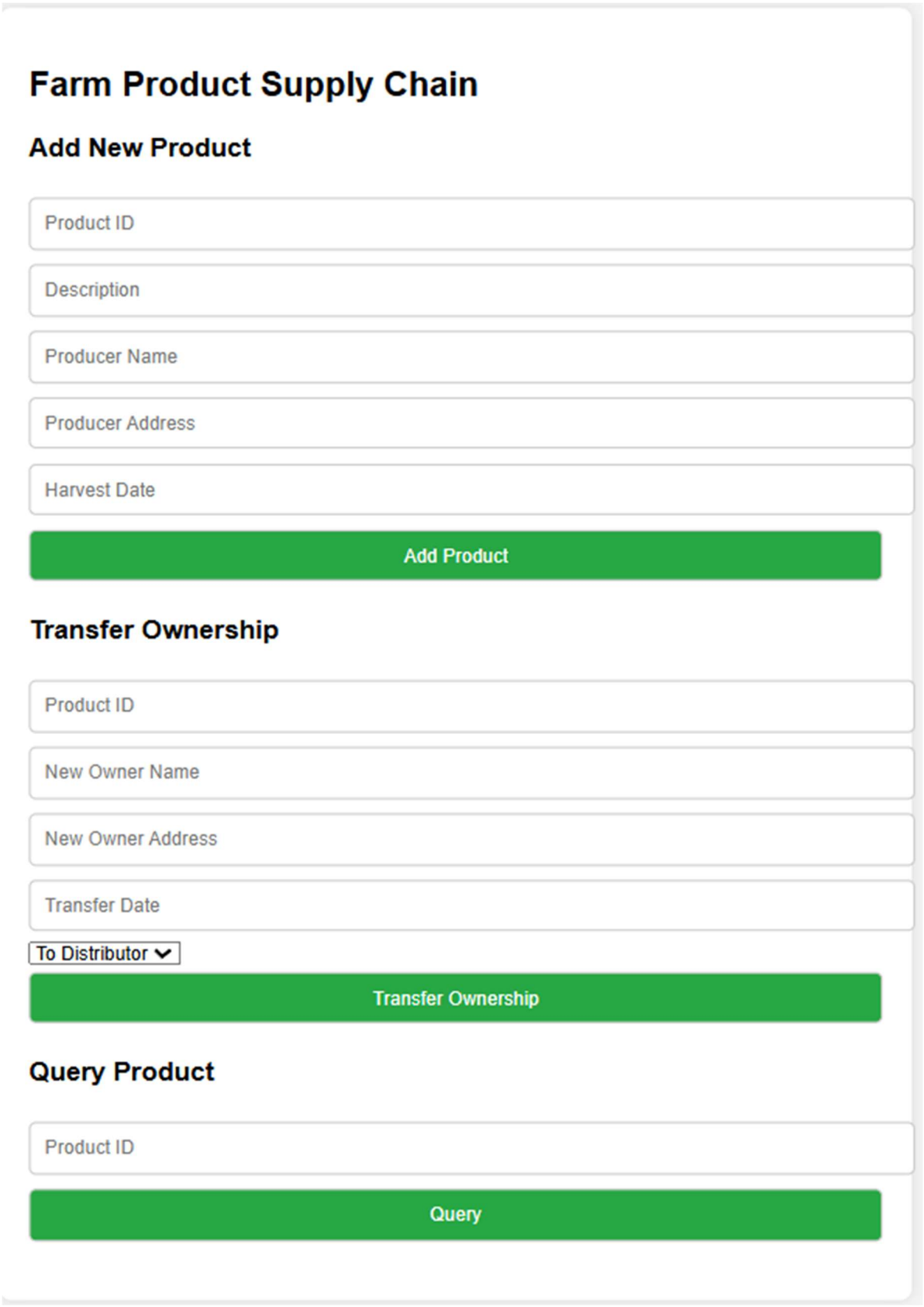
```
    {
        "inputs": [
            {
                "internalType": "string",
                "name": "id",
                "type": "string"
            },
            {
                "internalType": "string",
                "name": "newOwnerName",
                "type": "string"
            },
            {
                "internalType": "string",
                "name": "newOwnerAddress",
                "type": "string"
            },
            {
                "internalType": "string",
                "name": "transferDate",
                "type": "string"
            },
            {
                "internalType": "bool",
                "name": "toDistributor",
                "type": "bool"
            }
        ],
        "name": "changeCarOwnership",
        "outputs": [],
        "stateMutability": "nonpayable",
        "type": "function"
    },
    {
        "anonymous": false,
        "inputs": [
            {
                "indexed": false,
                "internalType": "string",
                "name": "productId",
                "type": "string"
            },
            {
                "indexed": false,
                "internalType": "string",
                "name": "newOwner",
                "type": "string"
            }
        ],
        "name": "OwnershipTransferred",
        "type": "event"
    },
    {
```

```
        "anonymous": false,
        "inputs": [
            {
                "indexed": false,
                "internalType": "string",
                "name": "productId",
                "type": "string"
            }
        ],
        "name": "ProductAdded",
        "type": "event"
    },
    {
        "inputs": [
            {
                "internalType": "string",
                "name": "id",
                "type": "string"
            }
        ],
        "name": "queryCarById",
        "outputs": [
            {
                "components": [
                    {
                        "internalType": "string",
                        "name": "productId",
                        "type": "string"
                    },
                    {
                        "internalType": "string",
                        "name": "productDescription",
                        "type": "string"
                    },
                    {
                        "internalType": "string",
                        "name": "producerName",
                        "type": "string"
                    },
                    {
                        "internalType": "string",
                        "name": "producerAddress",
                        "type": "string"
                    },
                    {
                        "internalType": "string",
                        "name": "harvestDate",
                        "type": "string"
                    },
                    {
                        "internalType": "string",
                        "name": "distributerName",
                        "type": "string"
```

```
                    },
                    {
                        "internalType": "string",
                        "name": "distributerAddress",
                        "type": "string"
                    },
                    {
                        "internalType": "string",
                        "name": "prodToDistDate",
                        "type": "string"
                    },
                    {
                        "internalType": "string",
                        "name": "retailerName",
                        "type": "string"
                    },
                    {
                        "internalType": "string",
                        "name": "retailerAddress",
                        "type": "string"
                    },
                    {
                        "internalType": "string",
                        "name": "distToRetaDate",
                        "type": "string"
                    }
                ],
                "internalType": "struct FarmProductSupplyChain.FarmProduct",
                "name": "",
                "type": "tuple"
            }
        ],
        "stateMutability": "view",
        "type": "function"
    }
];

        let web3;
        let contract;

        window.addEventListener("load", async () => {
            if (window.ethereum) {
                web3 = new Web3(window.ethereum);
                await window.ethereum.request({ method: "eth_requestAccounts" });
                const accounts = await web3.eth.getAccounts();
                contract = new web3.eth.Contract(contractABI, contractAddress, { from:
accounts[0] });
            } else {
                alert("Please install MetaMask!");
            }
        });

        async function addNewProduct() {
```

```
            const id = document.getElementById("productId").value;
            const description = document.getElementById("description").value;
            const producerName = document.getElementById("producerName").value;
            const producerAddress = document.getElementById("producerAddress").value;
            const harvestDate = document.getElementById("harvestDate").value;
            await contract.methods.addNewCa(id, description, producerName,
producerAddress, harvestDate).send();
            alert("Product added successfully");
        }

        async function transferOwnership() {
            const id = document.getElementById("transferId").value;
            const newOwnerName = document.getElementById("newOwnerName").value;
            const newOwnerAddress = document.getElementById("newOwnerAddress").value;
            const transferDate = document.getElementById("transferDate").value;
            const toDistributor = document.getElementById("ownerType").value ===
"distributor";
            await contract.methods.changeCarOwnership(id, newOwnerName,
newOwnerAddress, transferDate, toDistributor).send();
            alert("Ownership transferred successfully");
        }

        async function queryProduct() {
            const id = document.getElementById("queryId").value;
            const result = await contract.methods.queryCarById(id).call();
            document.getElementById("result").innerText = JSON.stringify(result, null,
2);
        }
    </script>
</body>
</html>
```

Output:


Farm Product Supply Chain
Add New Product
Product ID
Description
Producer Name
Producer Address
Harvest Date
Add Product
Transfer Ownership
Product ID
New Owner Name
New Owner Address
Transfer Date
To Distributor
Transfer Ownership
Query Product
Product ID
Query