Job Portal

JobPortal.sol

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

contract JobPortal {
    struct Applicant {
        uint id;
        string name;
        string skills;
        uint rating;
        uint totalRatings;
    }

    struct Job {
        uint id;
        string title;
        string description;
        address employer;
        bool isFilled;
    }

    mapping(uint => Applicant) public applicants;
    mapping(uint => Job) public jobs;
    mapping(address => uint) public applicantIdByAddress;

    uint public applicantCount;
    uint public jobCount;

    // 1. Add a new applicant
    function addApplicant(string memory _name, string memory _skills) public {
        require(bytes(_name).length > 0, "Name cannot be empty");
        require(bytes(_skills).length > 0, "Skills cannot be empty");

        applicantCount++;
        applicants[applicantCount] = Applicant(applicantCount, _name, _skills, 0, 0);
        applicantIdByAddress[msg.sender] = applicantCount;
    }

    // 2. Get applicant details
    function getApplicant(uint _id) public view returns (string memory, string memory,
uint) {
        require(_id > 0 && _id <= applicantCount, "Invalid applicant ID");

        Applicant memory a = applicants[_id];
        return (a.name, a.skills, a.rating);
    }
```

```
    // 3. Get applicant type (You can modify the type logic as needed, here returning
skills as applicant type)
    function getApplicantType(uint _id) public view returns (string memory) {
        require(_id > 0 && _id <= applicantCount, "Invalid applicant ID");
        return applicants[_id].skills;  // Returning skills as the "type"
    }

    // 4. Add a new Job to the portal
    function addJob(string memory _title, string memory _description) public {
        require(bytes(_title).length > 0, "Job title cannot be empty");
        require(bytes(_description).length > 0, "Job description cannot be empty");

        jobCount++;
        jobs[jobCount] = Job(jobCount, _title, _description, msg.sender, false);
    }

    // 5. Get job details
    function getJob(uint _id) public view returns (string memory, string memory,
address, bool) {
        require(_id > 0 && _id <= jobCount, "Invalid job ID");

        Job memory j = jobs[_id];
        return (j.title, j.description, j.employer, j.isFilled);
    }

    // 6. Applicants apply for a job
    function applyForJob(uint _jobId) public {
        require(_jobId > 0 && _jobId <= jobCount, "Invalid job ID");
        require(!jobs[_jobId].isFilled, "Job is already filled");
        require(applicantIdByAddress[msg.sender] != 0, "Only registered applicants can
apply");

        jobs[_jobId].isFilled = true;
    }

    // 7. Provide a rating to an applicant
    function rateApplicant(uint _id, uint _rating) public {
        require(_id > 0 && _id <= applicantCount, "Invalid applicant ID");
        require(_rating >= 1 && _rating <= 5, "Rating should be between 1 and 5");

        Applicant storage applicant = applicants[_id];
        applicant.totalRatings++;
        applicant.rating = (applicant.rating * (applicant.totalRatings - 1) + _rating)
/ applicant.totalRatings;
    }

    // 8. Fetch applicant rating
    function getApplicantRating(uint _id) public view returns (uint) {
        require(_id > 0 && _id <= applicantCount, "Invalid applicant ID");
        return applicants[_id].rating;
    }
}
```

JobPortal.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Job Portal</title>
    <script src="https://cdn.jsdelivr.net/npm/web3/dist/web3.min.js"></script>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #f4f4f4;
            text-align: center;
            padding: 20px;
        }
        h2, h3 {
            color: #333;
        }
        button {
            background-color: #007bff;
            color: white;
            border: none;
            padding: 10px 15px;
            margin: 10px;
            cursor: pointer;
            border-radius: 5px;
            font-size: 16px;
        }
        button:hover {
            background-color: #0056b3;
        }
        input {
            padding: 8px;
            margin: 5px;
            width: 250px;
            border: 1px solid #ccc;
            border-radius: 5px;
        }
        .container {
            background: white;
            padding: 20px;
            border-radius: 10px;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
            max-width: 400px;
            margin: auto;
        }
        #walletAddress {
            font-weight: bold;
            color: #007bff;
        }
```

```
    </style>
</head>
<body>
    <h2>Job Portal</h2>
    <button onclick="connectWallet()">Connect Wallet</button>
    <p id="walletAddress"></p>

    <div class="container">
        <h3>Register as Applicant</h3>
        <input type="text" id="applicantName" placeholder="Name">
        <input type="text" id="applicantSkills" placeholder="Skills">
        <button onclick="addApplicant()">Register</button>
    </div>

    <div class="container">
        <h3>Post a Job</h3>
        <input type="text" id="jobTitle" placeholder="Job Title">
        <input type="text" id="jobDescription" placeholder="Job Description">
        <button onclick="addJob()">Post Job</button>
    </div>

    <div class="container">
        <h3>Apply for a Job</h3>
        <input type="number" id="jobId" placeholder="Job ID">
        <button onclick="applyForJob()">Apply</button>
    </div>

    <div class="container">
        <h3>Rate an Applicant</h3>
        <input type="number" id="applicantId" placeholder="Applicant ID">
        <input type="number" id="rating" placeholder="Rating (1-5)">
        <button onclick="rateApplicant()">Rate</button>
    </div>

    <div class="container">
        <h3>Get Applicant Details</h3>
        <input type="number" id="getApplicantId" placeholder="Applicant ID">
        <button onclick="getApplicantDetails()">Get Details</button>
        <p id="applicantDetails"></p>
    </div>

    <div class="container">
        <h3>Get Applicant Type</h3>
        <input type="number" id="getApplicantTypeId" placeholder="Applicant ID">
        <button onclick="getApplicantType()">Get Type</button>
        <p id="applicantType"></p>
    </div>

    <div class="container">
        <h3>Get Applicant Rating</h3>
        <input type="number" id="getApplicantRatingId" placeholder="Applicant ID">
        <button onclick="getApplicantRating()">Get Rating</button>
        <p id="applicantRating"></p>
```

```
    </div>

    <script>
        let contract;
        let account;
        const contractAddress = "0xda07d58a2358f0a8a501ED92C87Fdb0b2395d9a7"; //
Replace with your contract address
        const abi = [
    {
        "inputs": [
            {
                "internalType": "string",
                "name": "_name",
                "type": "string"
            },
            {
                "internalType": "string",
                "name": "_skills",
                "type": "string"
            }
        ],
        "name": "addApplicant",
        "outputs": [],
        "stateMutability": "nonpayable",
        "type": "function"
    },
    {
        "inputs": [
            {
                "internalType": "string",
                "name": "_title",
                "type": "string"
            },
            {
                "internalType": "string",
                "name": "_description",
                "type": "string"
            }
        ],
        "name": "addJob",
        "outputs": [],
        "stateMutability": "nonpayable",
        "type": "function"
    },
    {
        "inputs": [
            {
                "internalType": "uint256",
                "name": "_jobId",
                "type": "uint256"
            }
        ],
        "name": "applyForJob",
```

```
        "outputs": [],
        "stateMutability": "nonpayable",
        "type": "function"
    },
    {
        "inputs": [
            {
                "internalType": "uint256",
                "name": "_id",
                "type": "uint256"
            },
            {
                "internalType": "uint256",
                "name": "_rating",
                "type": "uint256"
            }
        ],
        "name": "rateApplicant",
        "outputs": [],
        "stateMutability": "nonpayable",
        "type": "function"
    },
    {
        "inputs": [],
        "name": "applicantCount",
        "outputs": [
            {
                "internalType": "uint256",
                "name": "",
                "type": "uint256"
            }
        ],
        "stateMutability": "view",
        "type": "function"
    },
    {
        "inputs": [
            {
                "internalType": "address",
                "name": "",
                "type": "address"
            }
        ],
        "name": "applicantIdByAddress",
        "outputs": [
            {
                "internalType": "uint256",
                "name": "",
                "type": "uint256"
            }
        ],
        "stateMutability": "view",
        "type": "function"
```

```
    },
    {
        "inputs": [
            {
                "internalType": "uint256",
                "name": "",
                "type": "uint256"
            }
        ],
        "name": "applicants",
        "outputs": [
            {
                "internalType": "uint256",
                "name": "id",
                "type": "uint256"
            },
            {
                "internalType": "string",
                "name": "name",
                "type": "string"
            },
            {
                "internalType": "string",
                "name": "skills",
                "type": "string"
            },
            {
                "internalType": "uint256",
                "name": "rating",
                "type": "uint256"
            },
            {
                "internalType": "uint256",
                "name": "totalRatings",
                "type": "uint256"
            }
        ],
        "stateMutability": "view",
        "type": "function"
    },
    {
        "inputs": [
            {
                "internalType": "uint256",
                "name": "_id",
                "type": "uint256"
            }
        ],
        "name": "getApplicant",
        "outputs": [
            {
                "internalType": "string",
                "name": "",
```

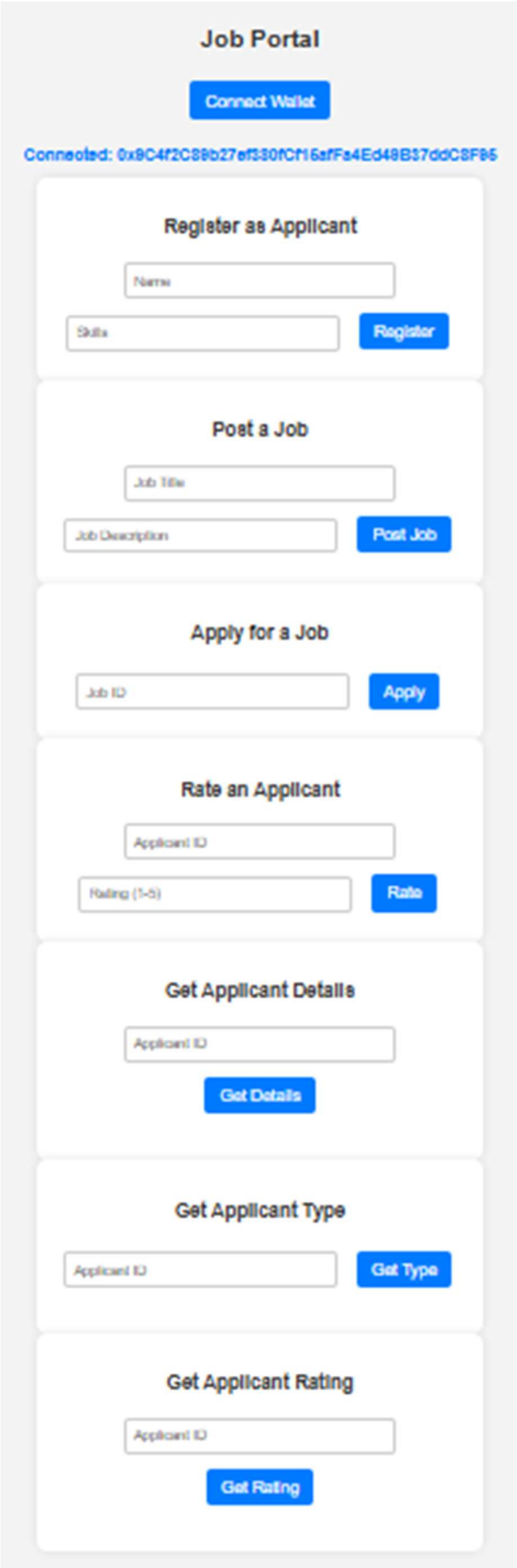
```
                "type": "string"
            },
            {
                "internalType": "string",
                "name": "",
                "type": "string"
            },
            {
                "internalType": "uint256",
                "name": "",
                "type": "uint256"
            }
        ],
        "stateMutability": "view",
        "type": "function"
    },
    {
        "inputs": [
            {
                "internalType": "uint256",
                "name": "_id",
                "type": "uint256"
            }
        ],
        "name": "getApplicantRating",
        "outputs": [
            {
                "internalType": "uint256",
                "name": "",
                "type": "uint256"
            }
        ],
        "stateMutability": "view",
        "type": "function"
    },
    {
        "inputs": [
            {
                "internalType": "uint256",
                "name": "_id",
                "type": "uint256"
            }
        ],
        "name": "getApplicantType",
        "outputs": [
            {
                "internalType": "string",
                "name": "",
                "type": "string"
            }
        ],
        "stateMutability": "view",
        "type": "function"
```

```
    },
    {
        "inputs": [
            {
                "internalType": "uint256",
                "name": "_id",
                "type": "uint256"
            }
        ],
        "name": "getJob",
        "outputs": [
            {
                "internalType": "string",
                "name": "",
                "type": "string"
            },
            {
                "internalType": "string",
                "name": "",
                "type": "string"
            },
            {
                "internalType": "address",
                "name": "",
                "type": "address"
            },
            {
                "internalType": "bool",
                "name": "",
                "type": "bool"
            }
        ],
        "stateMutability": "view",
        "type": "function"
    },
    {
        "inputs": [],
        "name": "jobCount",
        "outputs": [
            {
                "internalType": "uint256",
                "name": "",
                "type": "uint256"
            }
        ],
        "stateMutability": "view",
        "type": "function"
    },
    {
        "inputs": [
            {
                "internalType": "uint256",
                "name": "",
```

```
                "type": "uint256"
            }
        ],
        "name": "jobs",
        "outputs": [
            {
                "internalType": "uint256",
                "name": "id",
                "type": "uint256"
            },
            {
                "internalType": "string",
                "name": "title",
                "type": "string"
            },
            {
                "internalType": "string",
                "name": "description",
                "type": "string"
            },
            {
                "internalType": "address",
                "name": "employer",
                "type": "address"
            },
            {
                "internalType": "bool",
                "name": "isFilled",
                "type": "bool"
            }
        ],
        "stateMutability": "view",
        "type": "function"
    }
];

        async function connectWallet() {
            if (window.ethereum) {
                const web3 = new Web3(window.ethereum);
                await window.ethereum.request({ method: "eth_requestAccounts" });
                const accounts = await web3.eth.getAccounts();
                account = accounts[0];
                document.getElementById("walletAddress").innerText = "Connected: " +
account;
                contract = new web3.eth.Contract(abi, contractAddress);
            } else {
                alert("Please install MetaMask!");
            }
        }

        async function addApplicant() {
            const name = document.getElementById("applicantName").value;
            const skills = document.getElementById("applicantSkills").value;
```

```
            await contract.methods.addApplicant(name, skills).send({ from: account });
            alert("Applicant registered successfully!");
        }

        async function addJob() {
            const title = document.getElementById("jobTitle").value;
            const description = document.getElementById("jobDescription").value;
            await contract.methods.addJob(title, description).send({ from: account });
            alert("Job posted successfully!");
        }

        async function applyForJob() {
            const jobId = document.getElementById("jobId").value;
            await contract.methods.applyForJob(jobId).send({ from: account });
            alert("Applied for job successfully!");
        }

        async function rateApplicant() {
            const applicantId = document.getElementById("applicantId").value;
            const rating = document.getElementById("rating").value;
            await contract.methods.rateApplicant(applicantId, rating).send({ from:
account });
            alert("Applicant rated successfully!");
        }

        async function getApplicantDetails() {
            const applicantId = document.getElementById("getApplicantId").value;
            const details = await contract.methods.getApplicant(applicantId).call();
            document.getElementById("applicantDetails").innerText = `Name:
${details[0]}, Skills: ${details[1]}, Rating: ${details[2]}`;
        }

        async function getApplicantType() {
            const applicantId = document.getElementById("getApplicantTypeId").value;
            const applicantType = await
contract.methods.getApplicantType(applicantId).call();
            document.getElementById("applicantType").innerText = `Applicant Type:
${applicantType}`;
        }

        async function getApplicantRating() {
            const applicantId = document.getElementById("getApplicantRatingId").value;
            const rating = await
contract.methods.getApplicantRating(applicantId).call();
            document.getElementById("applicantRating").innerText = `Rating: ${rating}`;
        }
    </script>
</body>
</html>
```

Output:


Job Portal
Connect Wallet
Connected: 0x8C4f2C88b27ef330fCf16afFa4Ed48B37ddC8F85
Register as Applicant
Name
Skills
Register
Post a Job
Job Title
Job Description
Post Job
Apply for a Job
Job ID
Apply
Rate an Applicant
Applicant ID
Rating (1-5)
Rate
Get Applicant Details
Applicant ID
Get Details
Get Applicant Type
Applicant ID
Get Type
Get Applicant Rating
Applicant ID
Get Rating