**Job Portal Source Code:**

pragma solidity ^ 0.8.0;

contract JobPortal {

    //Variables

    struct applicant {

        uint candidate\_id;

        string candidate\_name;

        uint age;

        string emailid;

        string experience;

        string applicanttype;

        string skills;

        //address applicantaddress;

        uint rating;

       }

    struct job {

        uint jobid;

        string title;

        string description;

        uint salary;

        bool isOpen;

        }

    mapping (address => applicant) ApplicantDetails;

    mapping (uint => job) JobDetails;

    event JobApplication(uint  jobid, address  applicantaddress);

    address admin;

    // Functions

    modifier onlyApplicant {

        require(bytes(ApplicantDetails[msg.sender].applicanttype).length != 0,"Only Applicants can apply for a job");

        \_;

    }

    /\*Checks if the keccak hash of applicant type of the caller address is same

    as keccak hash of employerso that only employers can call certain functions\*/

    modifier onlyEmployer {

        require(keccak256(bytes(ApplicantDetails[msg.sender].applicanttype)) == keccak256(bytes("employer")),"Only Employers can perform this action");

        \_;

    }

    /\*Applicant details are added and user has to provide rating as zero for newly added

    applicants whose rating will be provided by employers\*/

    function addApplicant (uint \_id,string memory \_name,

    uint \_age,string memory \_emailid,string memory \_experience,string memory \_applicantype,string memory \_skills,uint \_rating) public {

        require(\_rating == 0,"Please provide Applicant rating as 0 when adding applicant");

        applicant memory AD = applicant (\_id,\_name,\_age,\_emailid,\_experience,\_applicantype,\_skills,\_rating);

        ApplicantDetails[msg.sender] = AD;

        }

    //Applicant details are fetched from mapping based on caller address

    function getApplicantDetails () public view returns (applicant memory) {

        return ApplicantDetails[msg.sender];

        }

    //Fetching the applicant type of the applicant based on address

    function applicantType () public view returns (string memory) {

        return ApplicantDetails[msg.sender].applicanttype;

    }

    //Adding job details only employers can perform this action due to modifier

    function addJob (uint \_jobid,string memory \_title,string memory \_description,uint \_salary, bool \_isOpen) public onlyEmployer {

        job memory JB = job (\_jobid,\_title,\_description,\_salary,\_isOpen);

        JobDetails[\_jobid] = JB;

    }

    //Fetching jonb details using job id

    function getJobDetails (uint \_jobid) public view returns (job memory) {

        return  JobDetails[\_jobid];

    }

    /\*Applying for a job when it is open only applicants can perform this action and

     an event will be triggered so that front end knows

     that an applicant has applied for a particular jobid\*/

    function applyJob (uint \_jobid) public onlyApplicant {

        require(JobDetails[\_jobid].isOpen , "Job is not open for application");

        emit JobApplication(\_jobid, msg.sender);

    }

    //To rate an Applicant

    function rateApplicant (uint \_rating) public  {

        require(\_rating >= 1 && \_rating <= 5, "Rating must be between 1 and 5");

        ApplicantDetails[msg.sender].rating = \_rating;

    }

    //Fetching Applicant rating

    function getApplicantrating () public view returns (uint) {

        return  ApplicantDetails[msg.sender].rating;

    }

}