Database Management System Project Report: Job Portal

Team:

K Mohammad Asbar (PES1UG21CS255)

Kallesh Achar KG (PES1UG21CS266)

Overview

The Job Portal project is a comprehensive web application designed to facilitate job seekers and employers in the employment process. This report provides insights into the database management system (DBMS) aspects of the project, outlining key functionalities and the corresponding database schema.

Technologies Used

Frontend Framework: React

Form Handling: React Hook Form

Data Validation: Zod

Date Handling: date-fns

UI Components: Lucide React, Button, Form, FormControl, FormField, FormItem, FormLabel, FormMessage, Popover, PopoverContent, PopoverTrigger, Input, Calendar

HTTP Client: Axios

State Management: useState

Notification: react-hot-toast

Routing: Next.js

Cookie Handling: next-client-cookies

Short Abstract:

The Job Portal project is a web application developed using React and a robust Database Management System (DBMS) to facilitate efficient job searching and recruitment processes. Leveraging technologies such as Axios for HTTP requests, Next.js for routing, and Zod for data validation, the project encompasses various features, including job listing display, search functionality, and user authentication. The modular design allows for functionalities like creating, applying, and deleting jobs, as well as accessing user and job details. The "Create Job" functionality, implemented with React Hook Form and Zod, enables employers to seamlessly post job openings with validated data. The project aims to provide a user-friendly and comprehensive solution for both job seekers and employers in the dynamic job market landscape.

Functionalities:

1. All Employee Jobs

Purpose: Retrieve all jobs associated with a specific employee.

Implementation: The backend server retrieves job data based on the employee's credentials.

2. All Jobs

Purpose: Fetch a list of all available jobs.

Implementation: The backend server provides an API endpoint to retrieve and display all job listings.

3. Apply Jobs

Purpose: Allow users to apply for specific job positions.

Implementation: The system records user applications in the database, linking applicants to the corresponding job.

4. Create Job

Purpose: Enable employers to create new job listings.

Implementation: The system processes employer input and inserts a new job record into the database.

5. Login Employee and Login User

Purpose: Authenticate employees and regular users for secure access.

Implementation: User credentials are validated against stored records in the database.

6. Delete Job

Purpose: Remove a job listing from the database.

Implementation: Authorized users can initiate the deletion of a job record.

7. Get All Job and Get All Job for Employee

Purpose: Retrieve a list of all jobs or jobs specific to an employee.

Implementation: The backend provides API endpoints to fetch job data based on the requested criteria.

8. Get Employee Details

Purpose: Retrieve detailed information about a specific employee.

Implementation: Employee details are queried from the database based on the provided parameters.

9. Get Job and Get User Details

Purpose: Fetch detailed information about a specific job or user.

Implementation: Queries to the database return the relevant details based on the specified criteria.

10. Get User Resume

Purpose: Retrieve the resume or profile of a specific user.

Implementation: The system accesses stored resume data in the database.

11. Register Employee, Register Job, and Register User

Purpose: Facilitate the registration process for employees, job postings, and regular users.

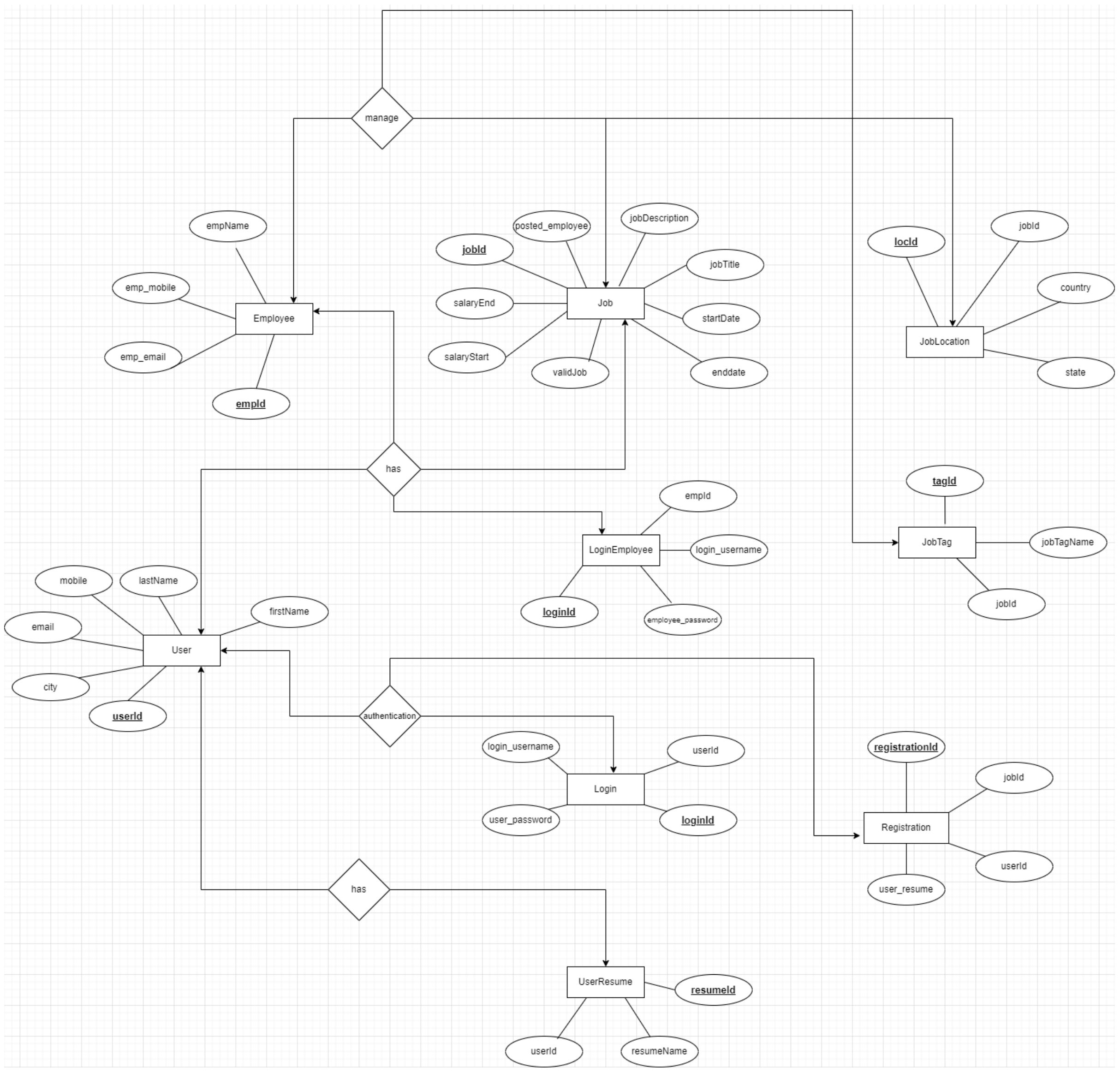
Implementation: New records are added to the database upon successful registration.

12. User Resume

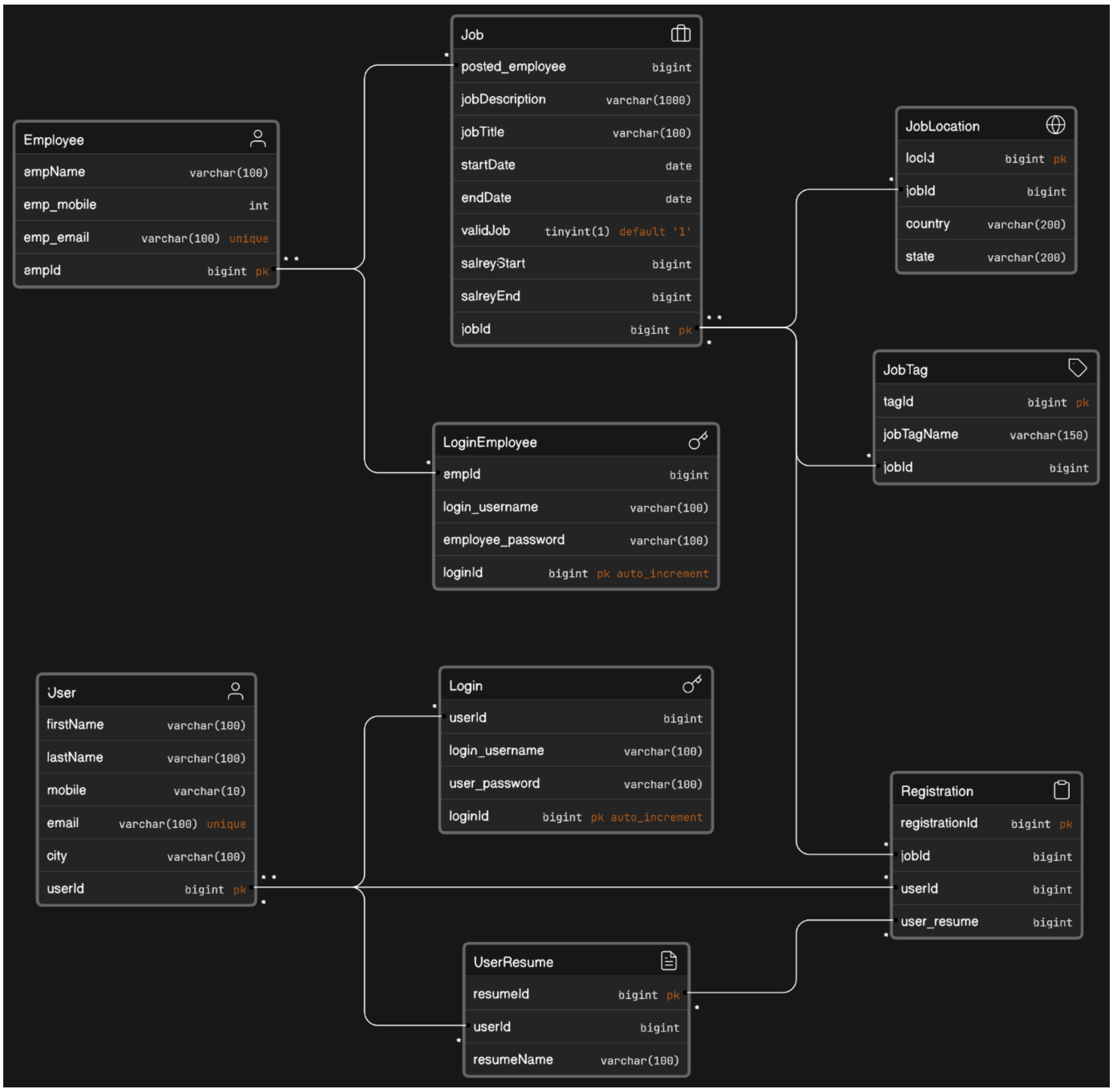
Purpose: Store and manage resumes uploaded by users.

Implementation: The system stores resume files in a secure manner, associating them with user profiles.

ER Diagram:


manage
empName
emp_mobile
Employee
emp_email
empId
posted_employee
jobDescription
jobId
jobTitle
salaryEnd
Job
startDate
salaryStart
validJob
enddate
locId
jobId
country
JobLocation
state
has
empId
LoginEmployee
login_username
loginId
employee_password
tagId
JobTag
jobTagName
jobId
mobile
lastName
firstName
email
User
city
userId
authentication
login_username
userId
Login
user_password
loginId
registrationId
jobId
Registration
userId
user_resume
has
UserResume
resumeId
userId
resumeName

Relational Schema:



DDL SQL Commands:

Create Table

Employee

```
CREATE TABLE Employee (
  empId bigint NOT NULL,
  empName varchar(100) NOT NULL,
  emp_mobile int NOT NULL,
  emp_email varchar(100) NOT NULL,
  PRIMARY KEY (empId),
  UNIQUE KEY emp_email (emp_email)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;
```

User

```
CREATE TABLE User (
  userId bigint NOT NULL,
  firstName varchar(100) NOT NULL,
  lastName varchar(100) NOT NULL,
  mobile varchar(10) DEFAULT NULL,
  email varchar(100) NOT NULL,
  city varchar(100) NOT NULL,
  PRIMARY KEY (userId),
  UNIQUE KEY email (email),
  UNIQUE KEY mobile (mobile)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;
```

Job

```
CREATE TABLE job (
  jobId bigint NOT NULL,
  posted_employee bigint NOT NULL,
  jobDescription varchar(1000) NOT NULL,
  jobTitle varchar(100) NOT NULL,
  startDate date NOT NULL,
  endDate date NOT NULL,
  validJob tinyint(1) NOT NULL DEFAULT '1',
  salreyStart bigint NOT NULL,
  salreyEnd bigint NOT NULL,
  PRIMARY KEY (jobId),
  KEY posted_employee (posted_employee),
  CONSTRAINT job_ibfk_1 FOREIGN KEY (posted_employee)
REFERENCES Employee (empId) ON DELETE CASCADE,
  CONSTRAINT endDateGtStartDate CHECK ((endDate >
startDate)),
  CONSTRAINT saleryConstraint CHECK ((salreyStart <
salreyEnd))
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;
```

JobLocation

```
CREATE TABLE jobLocation (
  locId bigint NOT NULL,
  jobId bigint NOT NULL,
  country varchar(200) NOT NULL,
  state varchar(200) NOT NULL,
  PRIMARY KEY (locId),
  KEY fk_jobLocation_jobId (jobId),
  CONSTRAINT fk_jobLocation_jobId FOREIGN KEY (jobId)
REFERENCES job (jobId) ON DELETE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;
```

JobTag

```
CREATE TABLE jobTag (
  tagId bigint NOT NULL,
  jobTagName varchar(150) NOT NULL,
  jobId bigint NOT NULL,
  PRIMARY KEY (tagId),
  KEY jobId (jobId),
  CONSTRAINT jobtag_ibfk_1 FOREIGN KEY (jobId) REFERENCES
job (jobId) ON DELETE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;
```

Login

```
CREATE TABLE Login (
  userId bigint DEFAULT NULL,
  login_username varchar(100) NOT NULL,
  user_password varchar(100) NOT NULL,
  loginId bigint NOT NULL AUTO_INCREMENT,
  PRIMARY KEY (loginId),
  KEY userId (userId),
  CONSTRAINT login_ibfk_1 FOREIGN KEY (userId) REFERENCES
User (userId) ON DELETE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=15 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;
```

LoginEmployee

```
CREATE TABLE LoginEmployee (
  empId bigint DEFAULT NULL,
  login_username varchar(100) NOT NULL,
  employee_password varchar(100) NOT NULL,
  loginId bigint NOT NULL AUTO_INCREMENT,
  PRIMARY KEY (loginId),
  KEY empId (empId),
  CONSTRAINT loginemployee_ibfk_1 FOREIGN KEY (empId)
REFERENCES Employee (empId) ON DELETE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=7 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;
```

Registration

```
CREATE TABLE Registration (
  registrationId bigint NOT NULL,
  jobId bigint NOT NULL,
  userId bigint NOT NULL,
  user_resume bigint NOT NULL,
  PRIMARY KEY (registrationId),
  UNIQUE KEY repetative_user (jobId, userId),
  KEY userId (userId),
  KEY user_resume (user_resume),
  CONSTRAINT registration_ibfk_1 FOREIGN KEY (jobId)
REFERENCES job (jobId) ON DELETE CASCADE,
  CONSTRAINT registration_ibfk_2 FOREIGN KEY (userId)
REFERENCES User (userId) ON DELETE CASCADE,
  CONSTRAINT registration_ibfk_3 FOREIGN KEY (user_resume)
REFERENCES UserResume (resumeId) ON DELETE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;
```

UserResume

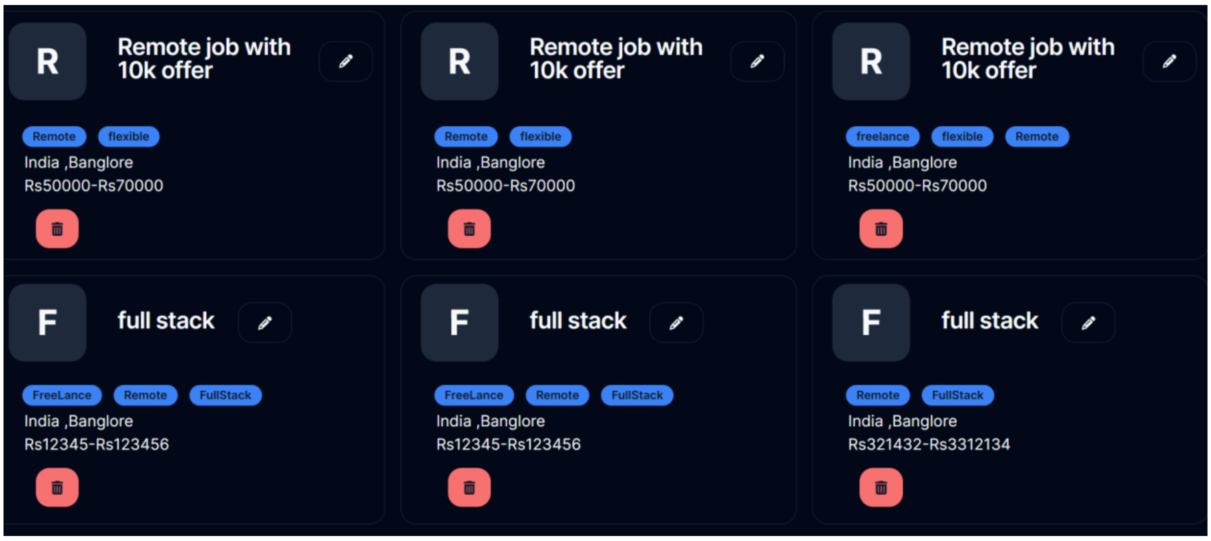
```
CREATE TABLE UserResume (
  resumeId bigint NOT NULL,
  userId bigint NOT NULL,
  resumeName varchar(100) DEFAULT NULL,
  PRIMARY KEY (resumeId),
  KEY userId (userId),
  CONSTRAINT userresume_ibfk_1 FOREIGN KEY (userId)
REFERENCES User (userId) ON DELETE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;
```

CRUD Operations Screenshots:

Select

```
SELECT Job.*,country,state,jobTagName FROM Job INNER JOIN
(SELECT jobTag.jobId,state,country,jobTagName FROM jobTag
INNER JOIN jobLocation ON jobTag.jobId=jobLocation.jobId) AS
mergeTable ON Job.jobId=mergeTable.jobId WHERE Job.validJob
= 1"
```

Before update



R Remote job with 10k offer
Remote flexible
India ,Banglore
Rs50000-Rs70000

R Remote job with 10k offer
Remote flexible
India ,Banglore
Rs50000-Rs70000

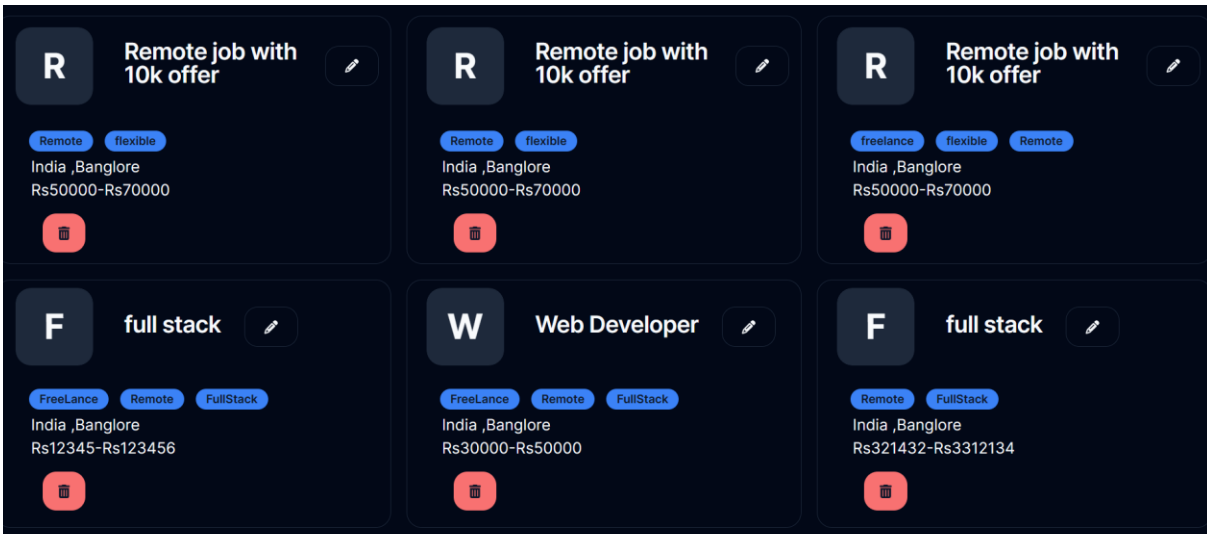
R Remote job with 10k offer
freelance flexible Remote
India ,Banglore
Rs50000-Rs70000

F full stack
FreeLance Remote FullStack
India ,Banglore
Rs12345-Rs123456

F full stack
FreeLance Remote FullStack
India ,Banglore
Rs12345-Rs123456

F full stack
Remote FullStack
India ,Banglore
Rs321432-Rs3312134

After update



R Remote job with 10k offer
Remote flexible
India ,Banglore
Rs50000-Rs70000

R Remote job with 10k offer
Remote flexible
India ,Banglore
Rs50000-Rs70000

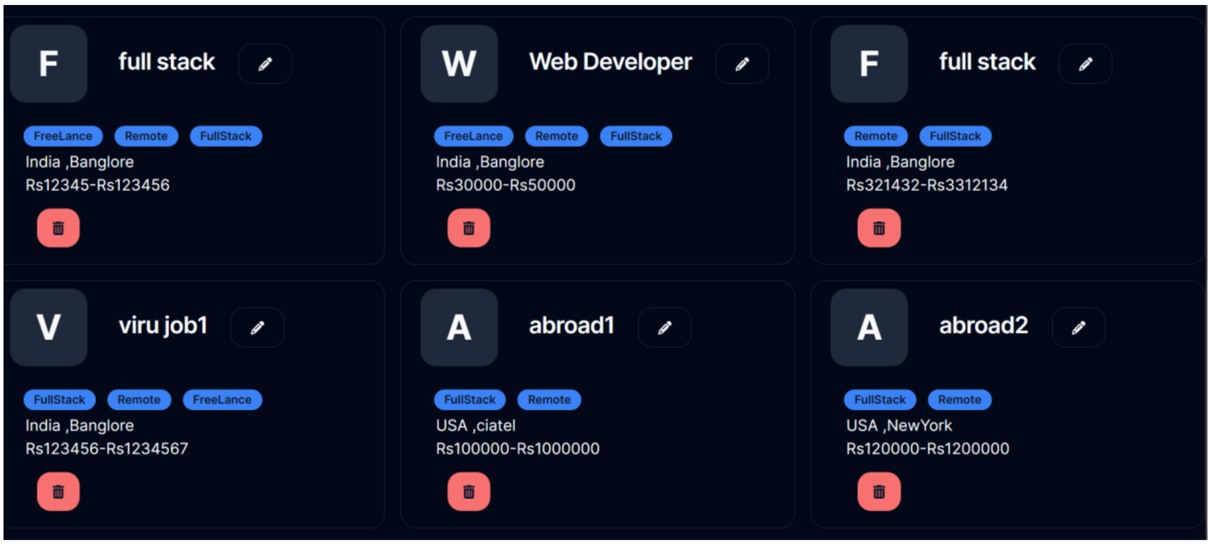
R Remote job with 10k offer
freelance flexible Remote
India ,Banglore
Rs50000-Rs70000

F full stack
FreeLance Remote FullStack
India ,Banglore
Rs12345-Rs123456

W Web Developer
FreeLance Remote FullStack
India ,Banglore
Rs30000-Rs50000

F full stack
Remote FullStack
India ,Banglore
Rs321432-Rs3312134

Before deleting


F full stack
FreeLance Remote FullStack
India ,Banglore
Rs12345-Rs123456

W Web Developer
FreeLance Remote FullStack
India ,Banglore
Rs30000-Rs50000

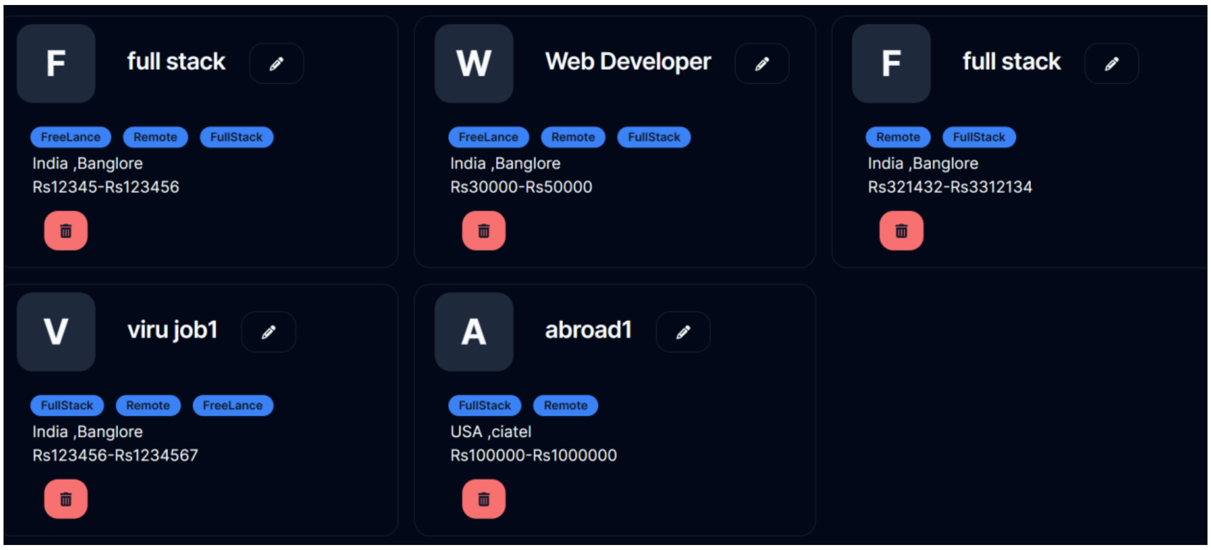
F full stack
Remote FullStack
India ,Banglore
Rs321432-Rs3312134

V viru job1
FullStack Remote FreeLance
India ,Banglore
Rs123456-Rs1234567

A abroad1
FullStack Remote
USA ,ciatel
Rs100000-Rs1000000

A abroad2
FullStack Remote
USA ,NewYork
Rs120000-Rs1200000

After deleting


F full stack
FreeLance Remote FullStack
India ,Banglore
Rs12345-Rs123456

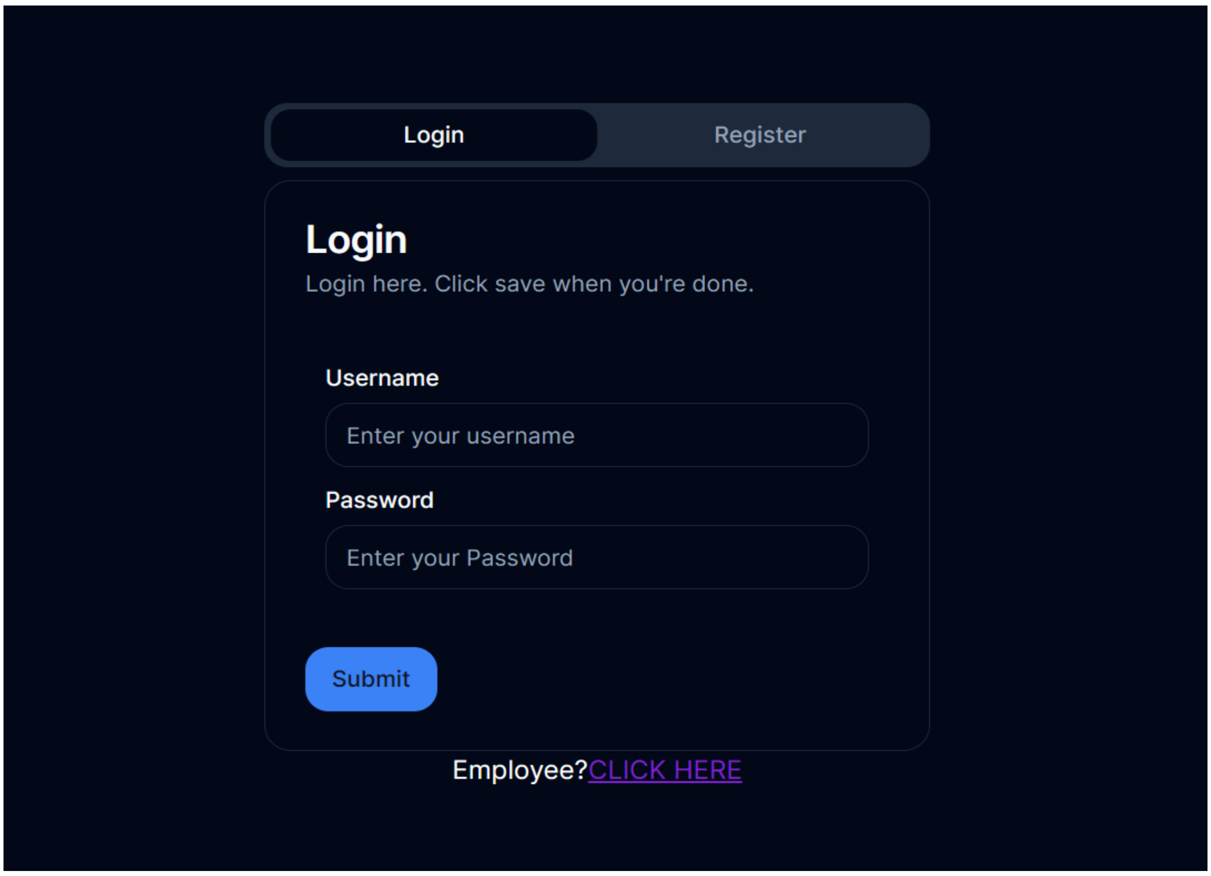
W Web Developer
FreeLance Remote FullStack
India ,Banglore
Rs30000-Rs50000

F full stack
Remote FullStack
India ,Banglore
Rs321432-Rs3312134

V viru job1
FullStack Remote FreeLance
India ,Banglore
Rs123456-Rs1234567

A abroad1
FullStack Remote
USA ,ciatel
Rs100000-Rs1000000

List of Functionalities and its associated Query screenshots from Front End:



Login

Login here. Click save when you're done.

Username

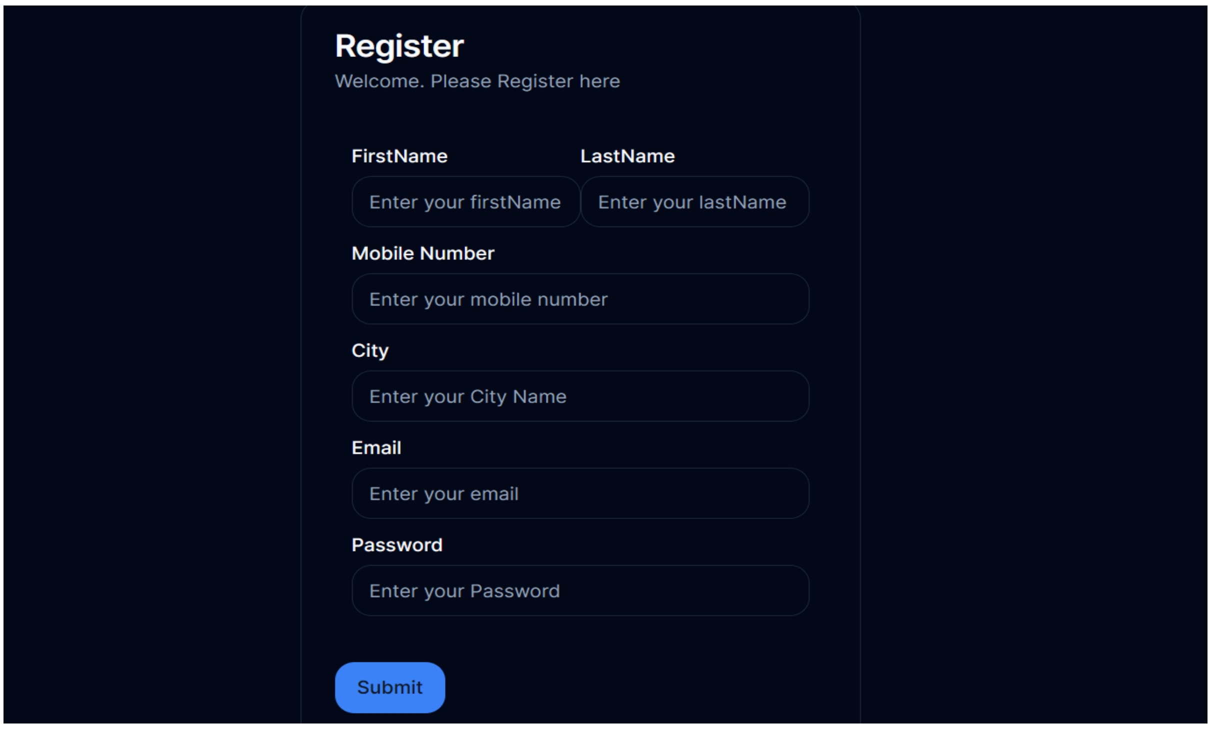
Enter your username

Password

Enter your Password

Submit

Employee?CLICK HERE



Register

Welcome. Please Register here

FirstName

Enter your firstName

LastName

Enter your lastName

Mobile Number

Enter your mobile number

City

Enter your City Name

Email

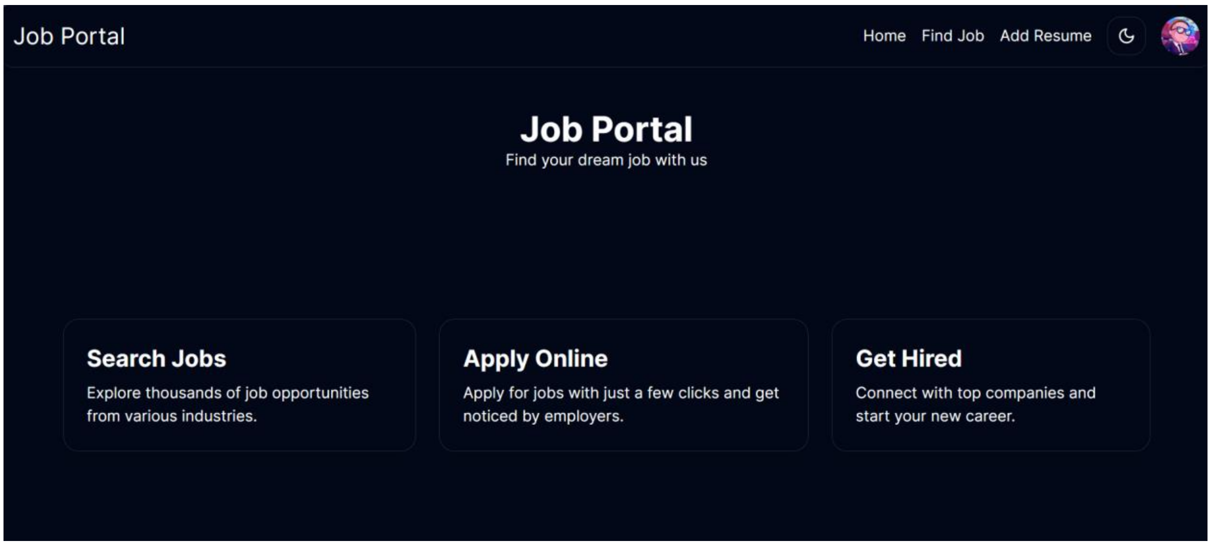
Enter your email

Password

Enter your Password

Submit

User



Job Portal

Home Find Job Add Resume

Job Portal

Find your dream job with us

Search Jobs

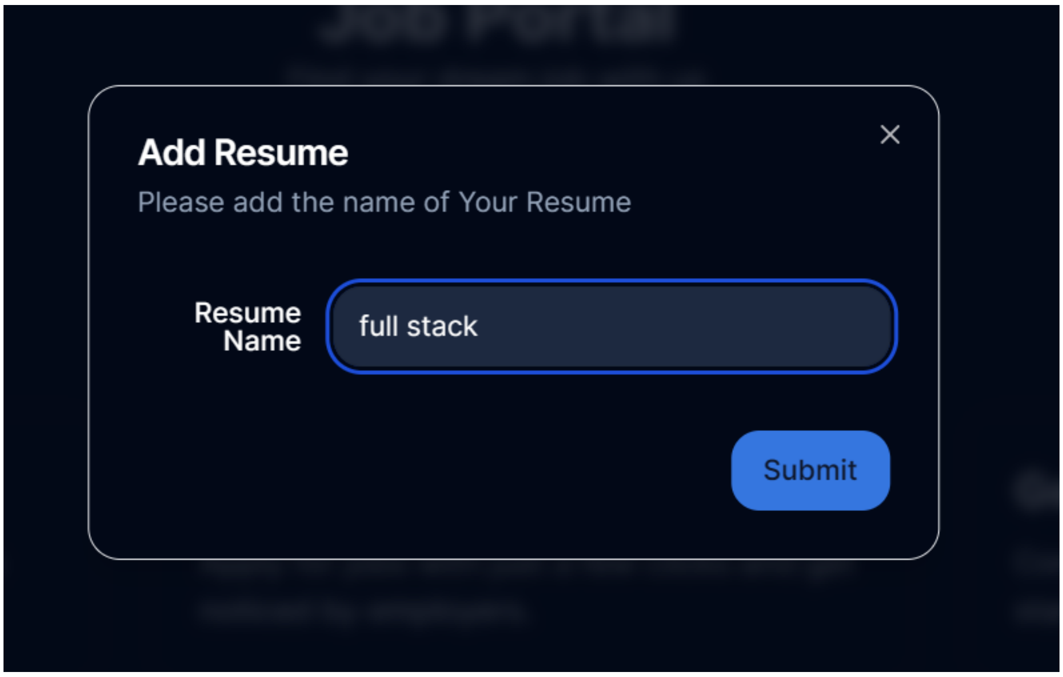
Explore thousands of job opportunities from various industries.

Apply Online

Apply for jobs with just a few clicks and get noticed by employers.

Get Hired

Connect with top companies and start your new career.



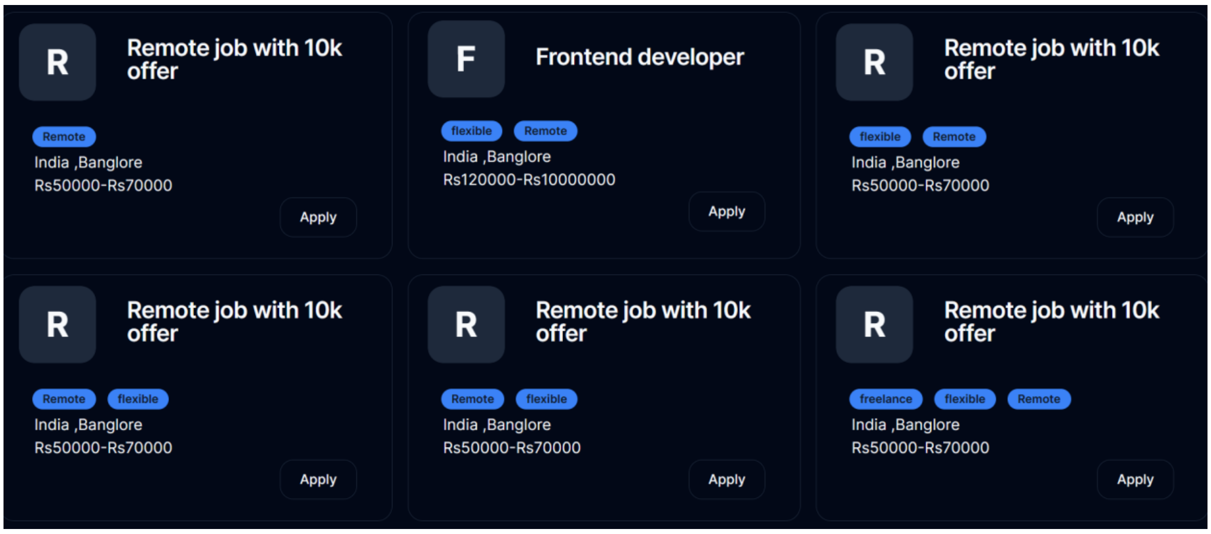
Add Resume

Please add the name of Your Resume

Resume Name: full stack

Submit

Find jobs



R
Remote job with 10k offer
Remote
India ,Banglore
Rs50000-Rs70000
Apply

F
Frontend developer
flexible Remote
India ,Banglore
Rs120000-Rs10000000
Apply

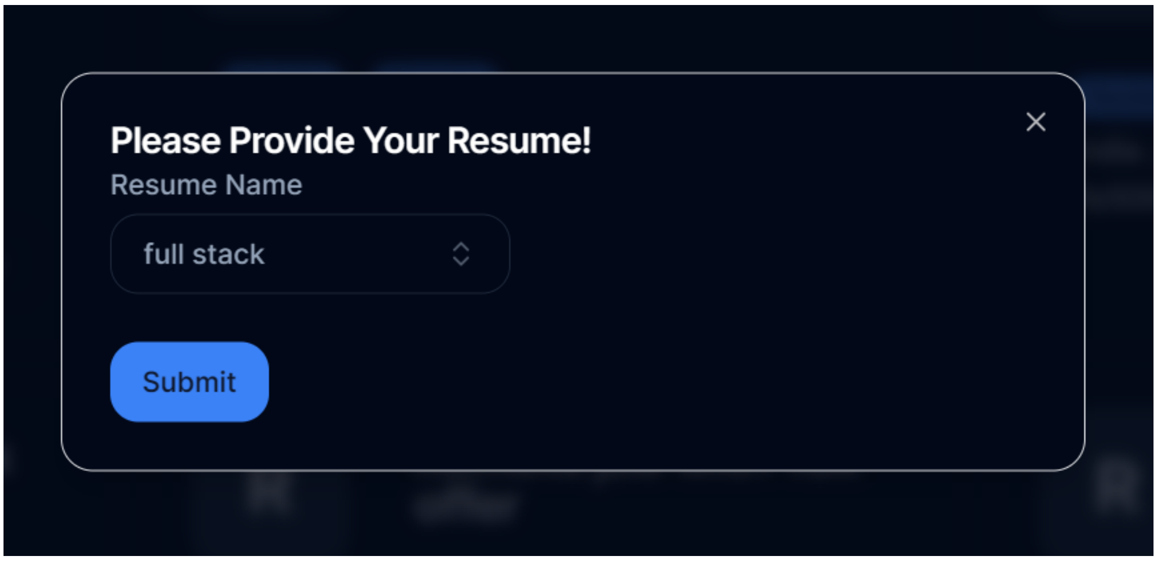
R
Remote job with 10k offer
flexible Remote
India ,Banglore
Rs50000-Rs70000
Apply

R
Remote job with 10k offer
Remote flexible
India ,Banglore
Rs50000-Rs70000
Apply

R
Remote job with 10k offer
Remote flexible
India ,Banglore
Rs50000-Rs70000
Apply

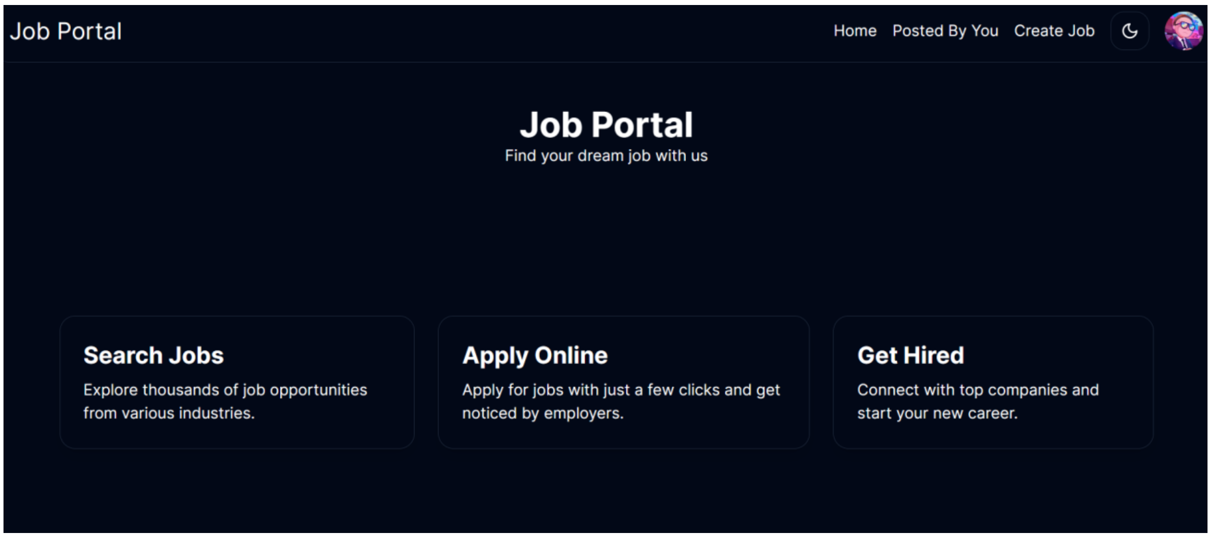
R
Remote job with 10k offer
freelance flexible Remote
India ,Banglore
Rs50000-Rs70000
Apply

Apply by adding the resume



Please Provide Your Resume!
Resume Name
full stack
Submit

Employee



Job Portal

Home Posted By You Create Job

Job Portal

Find your dream job with us

Search Jobs

Explore thousands of job opportunities from various industries.

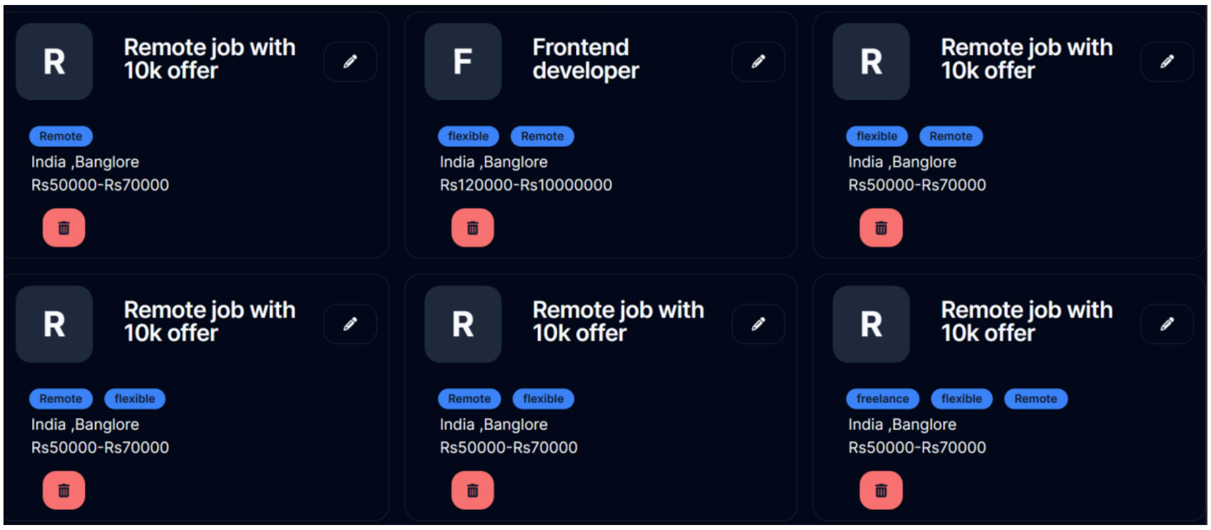
Apply Online

Apply for jobs with just a few clicks and get noticed by employers.

Get Hired

Connect with top companies and start your new career.

Jobs Posted by employee



R **Remote job with 10k offer**
Remote
India ,Banglore
Rs50000-Rs70000

F **Frontend developer**
flexible Remote
India ,Banglore
Rs120000-Rs10000000

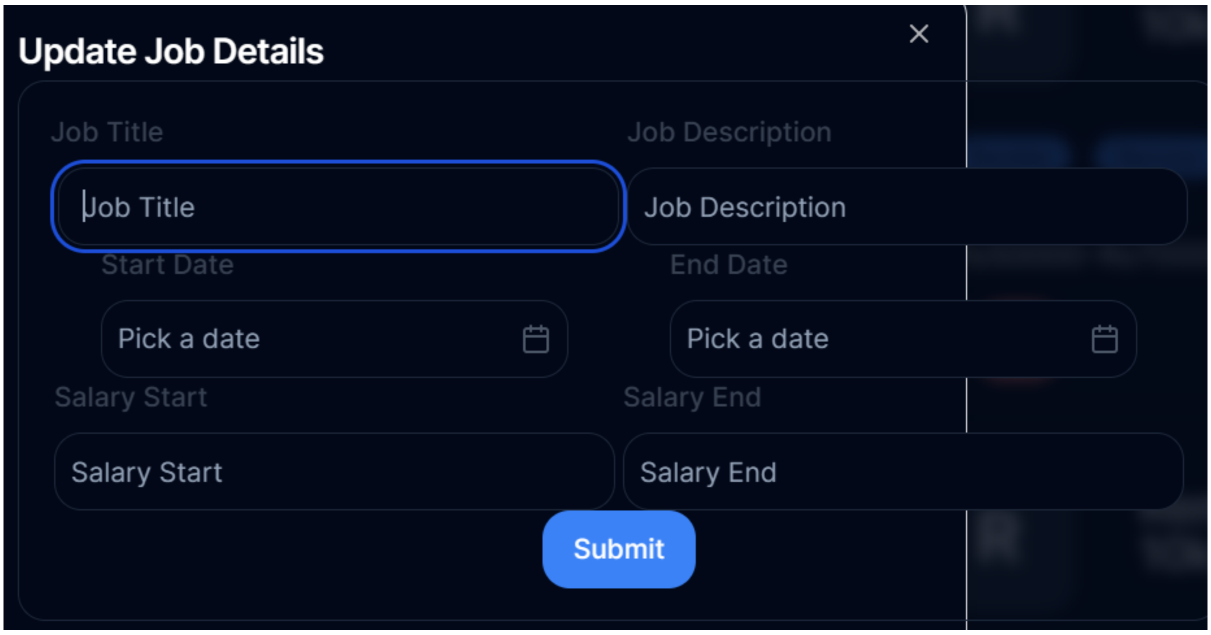
R **Remote job with 10k offer**
flexible Remote
India ,Banglore
Rs50000-Rs70000

R **Remote job with 10k offer**
Remote flexible
India ,Banglore
Rs50000-Rs70000

R **Remote job with 10k offer**
Remote flexible
India ,Banglore
Rs50000-Rs70000

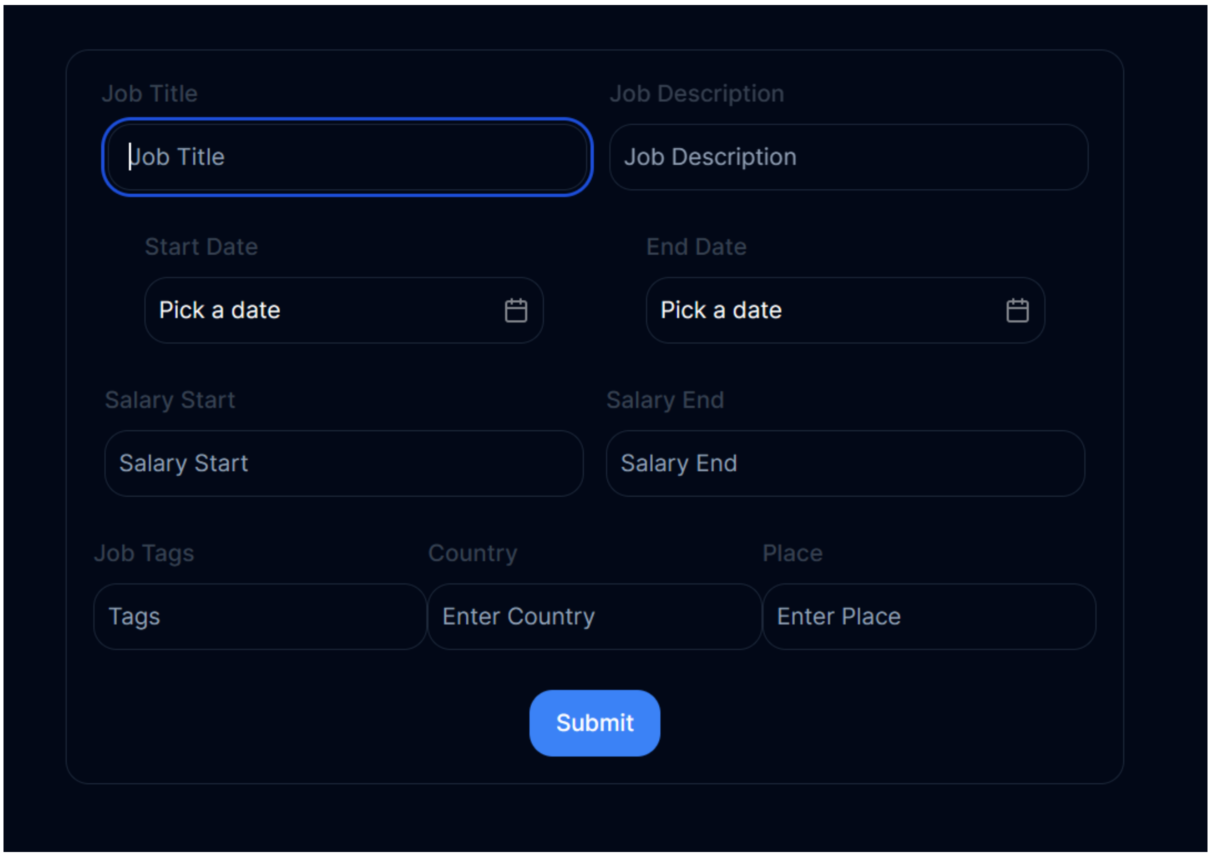
R **Remote job with 10k offer**
freelance flexible Remote
India ,Banglore
Rs50000-Rs70000

Edit job


Update Job Details

Job Title: Job Title
Job Description: Job Description
Start Date: Pick a date
End Date: Pick a date
Salary Start: Salary Start
Salary End: Salary End
Submit

Create job


Job Title: Job Title
Job Description: Job Description
Start Date: Pick a date
End Date: Pick a date
Salary Start: Salary Start
Salary End: Salary End
Job Tags: Tags
Country: Enter Country
Place: Enter Place
Submit

Procedures/Functions/Triggers and their Code snippets for invoking them:

Procedure

```
CREATE PROCEDURE GetJobsByEmployeeId(IN p_empId BIGINT)
BEGIN
    SELECT Job.*,country,state,jobTagName FROM Job INNER
JOIN (SELECT jobTag.jobId,state,country,jobTagName FROM
jobTag INNER JOIN jobLocation ON
jobTag.jobId=jobLocation.jobId) AS mergeTable ON
Job.jobId=mergeTable.jobId WHERE Job.posted_employee =
p_empId;
END
```

->The stored procedure retrieves job information, including details from the Job table and additional information related to job tags, state, and country from the jobTag and jobLocation tables. The results are filtered based on the provided employee ID, ensuring that only jobs posted by the specified employee are returned.

You can call this stored procedure by providing an employee ID as an argument, and it will return a result set with job information that matches the specified criteria.

Trigger

```
CREATE TRIGGER CheckSalaryRange BEFORE INSERT ON job FOR
EACH ROW BEGIN
    IF NEW.salreyStart > NEW.salreyEnd THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Error: The start salary must be
less than or equal to the end salary.';
    END IF;
END
```

-> The salary start (salaryStart) must be less than or equal to the salary end (salaryEnd). If this condition is not met, the trigger raises an exception with a custom error message, preventing the insertion of the new row.

->This trigger is useful for maintaining data integrity by enforcing a business rule that the start salary should always be less than or equal to the end salary. If this condition is violated during an insertion operation, an error will be raised, and the insertion will be aborted.