“<Online Job Portal>”

A

Project Report

Submitted in partial fulfillment of the requirement for the award of

Degree of

Bachelor of Computer Applications

Submitted to

310 - Cimage Professional College

CIMAGE
Knowledge . Skill . Success

आर्यभट्ट ज्ञान विश्वविद्यालय
ARYABHATTA KNOWLEDGE UNIVERSITY

Submitted By
Abhijeet Kumar
Registration No- 22303310176
BCA-6th Semester
Session-2022-2025

Under Joint Guidance of
Prof. Ravi Kumar Soni &
Prof. Niraj Kumar Singh
[Asst. Professor]
Computer Science &
Applications

Cimage Professional College
Department of Computer Science & Applications
Session 2022-2025

DATE:…………………

CERTIFICATE

This is to certify that the work embodies in this project entitled, **“Online Job Portal”,** being submitted by **Abhijeet Kumar,** **(22303310176)** in partial fulfillment of the requirement for the award of “**Bachelor of Computer Applications**” to ***ARYABHATTA KNOWLEDGE UNIVERSITY, PATNA*** during the academic year 2022-25 is a record of bonafide piece of work, carried out by him/her under our/my supervision and guidance in the “**Department of Computer Science & Applications”, Cimage Professional College.**

Dr. Neeraj Agrawal	**Prof. Amit Kumar Shukla**	**Dr. Pawan Kumar Jha**
Director	Head of Department-IT	Principal

Cimage Professional College

Department of Computer Science & Applications

Session 2022-2025

Date:..................................

APPROVAL CERTIFICATE

The project report entitled **Online Job Portal** being submitted by **Abhijeet Kumar** **(22303310176)** has been examined by us and is hereby approved for the award of degree "**Bachelor of Computer Applications**", for which it has been submitted. It is understood that by this approval the undersigned do not necessarily endorse or approve any statement made, the opinion expressed or conclusion drawn therein, but approve the project only for the purpose for which it has been submitted.

(Internal Examiner) (External Examiner)

Date: Date:

Date:..................................

DECLARATION

I, **Abhijeet Kumar** hereby declare that the work, which is being presented in the project report entitled **"Online Job Portal",** partial fulfillment of the requirements for the award of degree of "**Bachelor of Computer Applications**" submitted in the department of **Computer Science & Applications** (**Cimage Professional College**) is an authentic record of my own work carried under the joint guidance of **Prof. Ravi Kumar Soni Sir and Prof. Niraj Kumar Singh Sir.** I have not submitted the matter embodied in this report for the award of any other degree.

Abhijeet Kumar

Registration No : **22303310176**

BCA 6th Semester

Session-2022-2025

ACKNOWLEDGEMENT

"A journey is easier when you travel together. Interdependence is certainly more valuable than independence."

I would like to thanks, **Prof. Ravi Kumar Soni Sir** and **Prof. Niraj Kumar Singh Sir**, for providing regular guidance and insight into my project work. I also thank them for all advice he has givenme in the past year, and for always having time for me, whenever I needed.

I give special thanks to **Prof. Amit Kumar Shukla Sir, Prof. Raju Upadhyay Sir,** and **Prof. Anjesh Sir** for always being willing to help find solutions to any problems I had with my work.

"The completion of any project depends upon the cooperation, coordination, and combined efforts of several resources of knowledge, inspiration, and energy".

I also extend my deepest gratitude to **Director**, CIMAGE GROUP OF INSTITUITIONS, PATNA, **Dr. Neeraj Agrawal Sir, Dean** CIMAGE GROUP OF INSTITUITIONS, PATNA, **Dr. Neeraj Kumar Sir** and **Centre Head Mrs. Megha Agrawal Ma'am** for providing all the necessary facilities and true encouraging environment to bring out the best in my endeavors.

I express my gratitude and thanks to all the staff members of Computer Science departments for their sincere cooperation in furnishing relevant information to complete this Project well in time successfully.

I extend a special word to my friends, who have been a constant source of inspiration throughout my project work.

Lastly but not least I must express my cordial thank to my parent and family members who gave me the moral support without that it is impossible to complete my project work. With this note, I thank everyone for the support.

Abhijeet Kumar

Registration No : **22303310176**

BCA 6th Semester

Session-2022-2025

CONTENTS

INTRODUCTION

The “Online Job Portal ” has been design with the help of two powerful programming languages “React JS”, “JAVA Spring Boot” and “MySQL”. The “React JS” being in the foreground and the “ JAVA Spring Boot” and “MySQL” in the background respectively.

The foreground language is favorite today, due to his interactive user interface and supporting powerful tool for development of interactive software.

The background language is favorite many of the programmers because it offers the much wanted structured query facility. The MySQL provides high security, as the database is stored in the logical area called “DB Engine”. As “DB Engine” is created in the virtual memory and end user cannot access that directly it.

With these thoughts the “SCHOOL MANAGEMENT SYSTEM” has been developed. The “JAVA” gives the interactive look and “MySQL” gives high security.

OBJECTIVE

The motive of the project entitled “**Online Job Searching**” is to maintain all the record and activities taken place in the Recuriter department of the organisation.

The objectives of this system:

- Maintain the information about Employee and Employer
- Maintain the details of Employers.
- Maintain the Software Employee details
- Record the various kinds of Job details.
- Maintain the back-up o of software.
- Provide various reports like fee report, About the Job report etc.

PROJECT CATEGORY JAVA

The Java is a programming language originally developed by James Gosling at Sun Microsystems (which is now a subsidiary of Oracle Corporation) and released in 1995 as a core component of Sun Microsystems' Java platform. The language derives much of its syntax from C and C++ but has a simpler object model and fewer low-level facilities. Java applications are typically compiled to bytecode (class file) that can run on any Java Virtual Machine (JVM) regardless of computer architecture. Java is general-purpose, concurrent, class-based, and object-oriented, and is specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere". Java is considered by many as one of the most influential programming languages of the 20th century, and widely used from application software to web application.[9][10]

The original and reference implementation Java compilers, virtual machines, and class libraries were developed by Sun from 1995. As of May 2007, in compliance with the specifications of the Java Community Process, Sun relicensed most of their Java technologies under the GNU General Public License. Others have also developed alternative implementations of these Sun technologies, such as the GNU Compiler for Java and GNUClasspath.

ADVANTAGES OF JAVA

1. **Java is simple:** Java was designed to be easy to use and is therefore easy to write, compile, debug, and learn than other programming languages. The reason that why Java is much simpler than C++ is because Java uses automatic memory allocation and garbage collection where else C++ requires the programmer to allocate memory and to collect garbage.

2. **Java is object-oriented:** Java is object-oriented because programming in Java is centered on creating objects, manipulating objects, and making objects work together. This allows you to create modular programs and reusable code.

3. **Java is platform-independent:** One of the most significant advantages of Java is its ability to move easily from one computer system to another.

4. **Java is distributed**: Distributed computing involves several computers on a network working together. Java is designed to make distributed computing easy with the networking capability that is inherently integrated into it.

5. **Java is interpreted**: An interpreter is needed in order to run Java programs. The programs are compiled into Java Virtual Machine code called bytecode.

6. **Java is secure:** Java is one of the first programming languages to consider security as part of its design. The Java language, compiler, interpreter, and runtime environment were each developed with security in mind.

7. **Java is robust:** Robust means reliable and no programming language can really assure reliability. Java puts a lot of emphasis on early checking for possible errors, as Java compilers are able to detectmany problems that would first show up during execution time in other languages.

8. **Java is multithreaded:** Multithreaded is the capability for a program to perform several tasks simultaneously within a program.In Java, multithreaded programming has been smoothly integrated into it, while in other languages, operating system-specific procedures have to be called in order to enable multithreading.

TOOLS/PLATFORM REQUIRED

Operating System

- Windows XP ,Windows 7,8,10,11

HARDWARE & SOFTWARE REQUIREMENTS

Hardware Requirements:

- Intel i5 Processor
- 8 GB RAM
- 512 GB Solid State Drive

Software Requirements:

- JDK 17.0
- MySQL
- Intellij IDEA/Editor
- Spring Boot 3.5.3

PROBLEM DEFINITION

To develop an Online Job Portal that maintains records of Employee , Employers, job postings, and applications in order to simplify and automate the hiring process for both employers and candidates.

REQUIREMENT SPECIFICATIONS

A Software Requirements Specification (SRS) is a complete description of the behavior of the system to be developed. It includes a set of use cases that describe all the interactions the users will have with the software. Use cases are also known as functional requirements. In addition to use cases, the SRS also contains non-functional (or supplementary) requirements. Non-functional requirements are requirements which impose constraints on the design or implementation (such as performance engineering requirements, quality standards, or design constraints).

The general requirements of the project are:

- **To maintain the registration details of employees and employers.**
- **To keep track of job postings and application submissions by employees.**
- **To record employee profile details, including resumes, skills, and experience.**
- **To manage employer accounts and their posted job vacancies.**
- **To enable employees to search, filter, and apply for suitable jobs.**

PROJECT PLANNING & SCHEDULING

Project planning is part of project management, which relates to the use of schedules such as Gantt charts to plan and subsequently report progress within the project environment.

Initially, the project scope is defined and the appropriate methods for completing the project are determined. Following this step, the durations for the various tasks necessary to complete the work are listed and groupedinto a work breakdown structure. The logical dependencies between tasks are defined using an activity network diagram that enables identification ofthe critical path. Float or slack time in the schedule can be calculated using software. Then the necessary resources can be estimated and costs for each activity can be allocated to each resource, giving the total project cost. At this stage, the project plan may be optimized to achieve the appropriate balance between resource usage and project duration to comply with the project objectives. Once established and agreed, the plan becomes what is known as the baseline. Progress will be measured against the baseline throughout the life of the project. Analyzing progress compared to the baseline is known as earned value management.

The PROJECT SCHEDULE is a calendar that links the tasks to be done with the resources that will do them. Before a project schedule can be created, the project manager must have a work breakdown structure (WBS), an effort estimate for each task, and a resource list with availability for each resource. If these are not yet available, it may be possible to create something that looks like a schedule, but it will essentially be a work of fiction. A project manager's time is better spent on working with the team to create a WBS and estimates (using a consensus-driven estimation method like Wideband Delphi than on trying to build a project schedule without them.

GANTT CHART

Gantt charts (developed by Henry L. Gantt) are a project control technique that can be used for several purposes including scheduling, budgeting and resource planning. A Gantt chart is a bar chart with each bar representing an activity.

The bars are drawn against a time line. The length of each bar is proportional to the length of time planned for the activity. Gantt chart can take different phase depending on their intended use. The Gantt chart of **Online Job Portal** is drawn for the job management. The Gantt chart is drawn below:

Gantt Chart

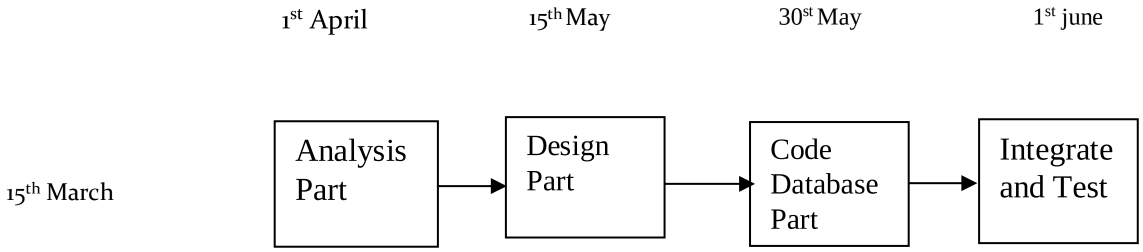
Task Name	Duration	1 April				May				1 June			
		Week 1	Week 2	Week 3	Week 4	Week 1	Week 2	Week 3	Week 4	Week 1	Week 2	Week 3	Week 4
Project Kickoff	1 Week	✓											
Requirement Gathering	2 Weeks		✓	✓									
Frontend Design (React)	3 Weeks				✓	✓	✓						
Backend Dev (Spring Boot)	3 Weeks						✓	✓	✓				
Integration (React + Spring)	2 Weeks									✓	✓		
Testing & Debugging	2 Weeks											✓	✓
Report & Final Submission	1 Week												✓

PERT CHART

PERT (Project Evaluation and Review Technique) charts consist of a network of boxes and arrows. The boxes represent activities and the arrows represent task dependencies.

PERT is organized by events and activities or tasks. PERT have more advantages and they are likely to be used for more complex projects. Through PERT chart the various task paths are defined. PERT enables the calculation of critical path. Each path consists of combination of tasks, which must be completed. The time and the cost associated with each task along a path are calculated, and the path that requires the greatest amount of elapsed time is the critical path.

PERT controls time and cost during the project and also facilitate finding the right balance between completing a project on time and completing it within the budget. There are thus not one but many critical paths, depending on the permutations of the estimates for each task. This makes analysis of critical path in PERT charts very complex. The PERT chart representation of the **Online Job Portal** is shown below.



SYSTEM ANALYSIS

IDENTIFICATION OF NEED

To Important Indication for software we use the Principles of Requirement Engineering. Requirement engineering provides the appropriate mechanism for understanding what the customer wants, analyzing need, assessing feasibility, negotiating a reasonable solution, specifying the solution unambiguously, validating the specification and managing the requirement as they are transformed into an operational system. The requirement engineering process can be described in five distinct steps:

Requirement elicitation.

Requirement analysis & negotiation.

Requirement specification.

System Modeling.

Requirement validation.

Requirement Management.

In other words we can say that requirement analysis is a software task that bridges the gap between system level requirement engineering and software design.

Requirement analysis allows the software engineering to refine the software allocation and build models of the data, functional and behavioral domains that will be treated by software. Requirement analysis provides the software designer with a representation of information, function and behavior that can be translated into data, architectural, interface and component level design; finally the requirement specification provides the developer and the customer with the means to assess quality once software is built.

The most commonly used requirement elicitation technique is to conduct a meeting or interview. The first meeting between a software engineer and customer can be likened to the awareness of a first date between to adolescents. Neither person knows what to

say or ask; Both are worried that what they do say will bemisinterpreted; both are thinking about where it might lead (Both likely have radically different expectation here); Both want to get the think over with, but at the same time, both want it to be a success.

Here according to this principle the analyst starts by asking context-free-questions. That is a set of question that will lead to a basic understanding of the problem, the people who want a solution, the nature of solution that is desired, and the effectiveness of the first encounter itself. The first set of Context-free question focuses on the customer, the overall goals, and the benefits. For example, the analyst might ask: -

Who is behind the request for this Work?

Who will use the solution?

What will be economic benefit of a successful solution?

Is there another source for the solution that you need?

The next set of questions enables the software engineer to gain a better Understanding of the problem and the customer to voice his or her perceptions about a solution: -

How would you characterize “good” output that would be generated by a successful solution?

What problem(s) will this solution address?

Can you show me (or describe) the environment in which the solution will be used?

Will special performance issues or constraints affect the way the solution is approached?

The final set of question focuses on the effectiveness of the meeting:

Are you the right person to answer these questions? Are your answers? Official”?

Are my questions relevant to the problem that you have?

Am I asking too many questions?

Can anyone else provide additional information?

Should I be asking you anything else?

According to the above concepts I went to institution and met its staffs, management, and some persons related with the organizational work, which advised me related with my project such as details of agent, policy related information will be cater this system.

Preliminary Investigation: -

The first step in the system development life cycle is the preliminary investigation to determine the feasibility of the System. The purpose of the preliminary investigation is to evaluate projectrequests. It is not a design study nor does it includes the collection of details top describe the business system in all respect. Rather, it is collecting of information that helps committee members to evaluate the merits of the project request and make an informed judgment about the feasibility of the proposed project.

Here for the Ngo Management System, I have worked on the preliminary investigation that accomplished the following objectives: -

Clarify and understand the project request.

Determine the size of the project.

Assess costs and benefits of alternative approaches.

Determine the technical and operational; feasibility of alternative approaches.

Report the findings to management, with recommendation outlining the acceptances or rejection of the proposal.

Considering above criteria I also keep in mind that the requirements are clearly understandable when the clarification of project request is enquired. The data of the Ngo Management System, which are collected by me during preliminary investigation, are through Reviewing organization Documents Onsite observation and

Conducting interviews.

FEASIBILITY STUDY

Not everything imaginable is feasible, not even in software, even as it may appear to outsiders. Feasibility is the determination of whether or not a project is wrong doing. On the contrary, software feasibility has seven solid dimensions as below: -

- **Technical feasibility.**
- **Operational feasibility.**
- **Economic feasibility.**
- **Social feasibility.**
- **Management feasibility.**
- **Legal feasibility.**
- **Time feasibility.**

The process followed in making this determination is called a feasibility study. The type of study determines if a project can and should be taken. Once it has been determined that a project is feasible, that analyst can go ahead and prepare the project specification which finalizes project requirements. Generally, feasibility studies are undertaken within tight time constraints and normally culminate in a written and oral feasibility report. The contents and recommendation of such a study will be used as a sound basis for deciding whether to proceed, postpone or cancel the project. Thus, since the feasibility study may lead to the commitment of large resources, it becomes necessary that it should be conducted competently and that no fundamental errors of judgment are made.

In other words we can say that a feasibility study is conducted to select the best system that meets performance requirement. This contains and identification description, an evaluation of candidate system and the selection of best system for the job. The system required a statement of constraints; the identification of specific system objective and a description of outputs define performance.

TECHNICAL FEASIBILITY: -

The technical feasibility of Course4Job assesses the availability and capabilities of the required technologies, software, and infrastructure. Given the advancements in Spring Boot and React Js, the project is technically feasible, promising the development of a robust and scalable platform for efficient job seeking and recruitment.

ECONOMICAL FEASIBILITY: -

The development of the Online Job Portal System is **economically feasible** as it requires minimal investment in terms of hardware and software. The system will be developed using open-source technologies like **React.js, Spring Boot, and MySQL**, which eliminates licensing costs. Since most of the development and deployment can be done using freely available resources and platforms, the overall cost remains low.

Moreover, by automating the hiring process for employers and employees, the portal reduces the time, effort, and expenses involved in traditional recruitment methods. The long-term benefits—such as reduced paperwork, quicker hiring, and improved job matching—make the system cost-effective and a worthy investment.

OPERATIONAL FEASIBILITY: -

The Online Job Portal System is operationally feasible as it is user-friendly, efficient, and designed to meet the needs of both employees and employers. The system provides a smooth interface for users to register, post jobs, apply for jobs, and manage their profiles with ease.

Both employees and employers can operate the system with minimal technical knowledge. With simple navigation, secure login, and clear instructions, users can perform tasks without requiring much training or support.

Additionally, since the system is web-based, it can be accessed from any device with internet connectivity, making it highly practical and accessible. The automation of job matching and application tracking improves overall productivity and saves time for all users.

In nutshell we can say that it has following operational features:

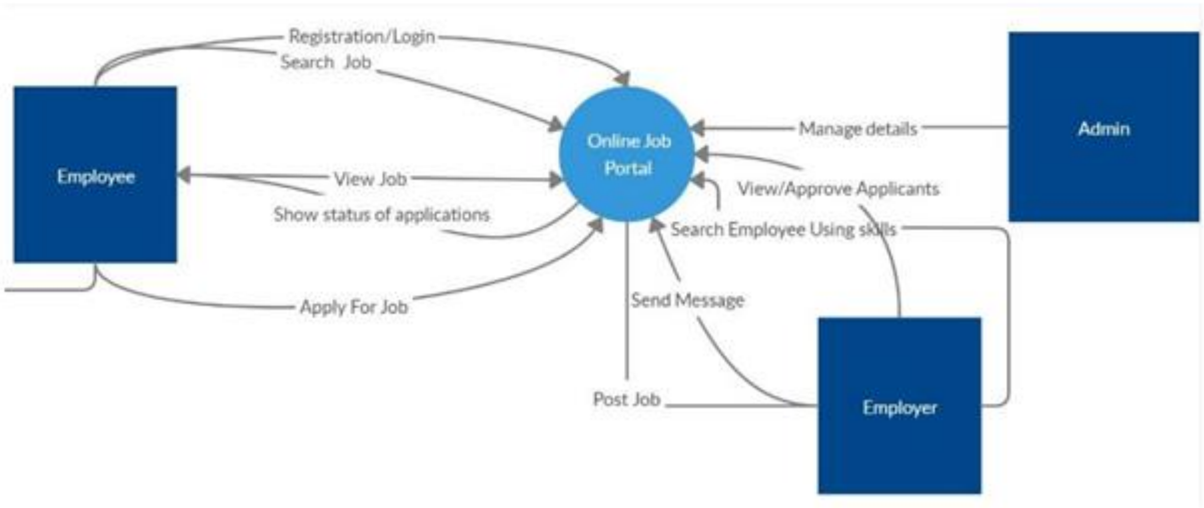
1. It is User-Friendly.
2. It is having less paperwork.
3. Efficient tractability.
4. Query can be generated.
5. Various Reports and Forms can be generated.
6. Fully protected by Password and User name for unauthorized access.

ANALYSIS

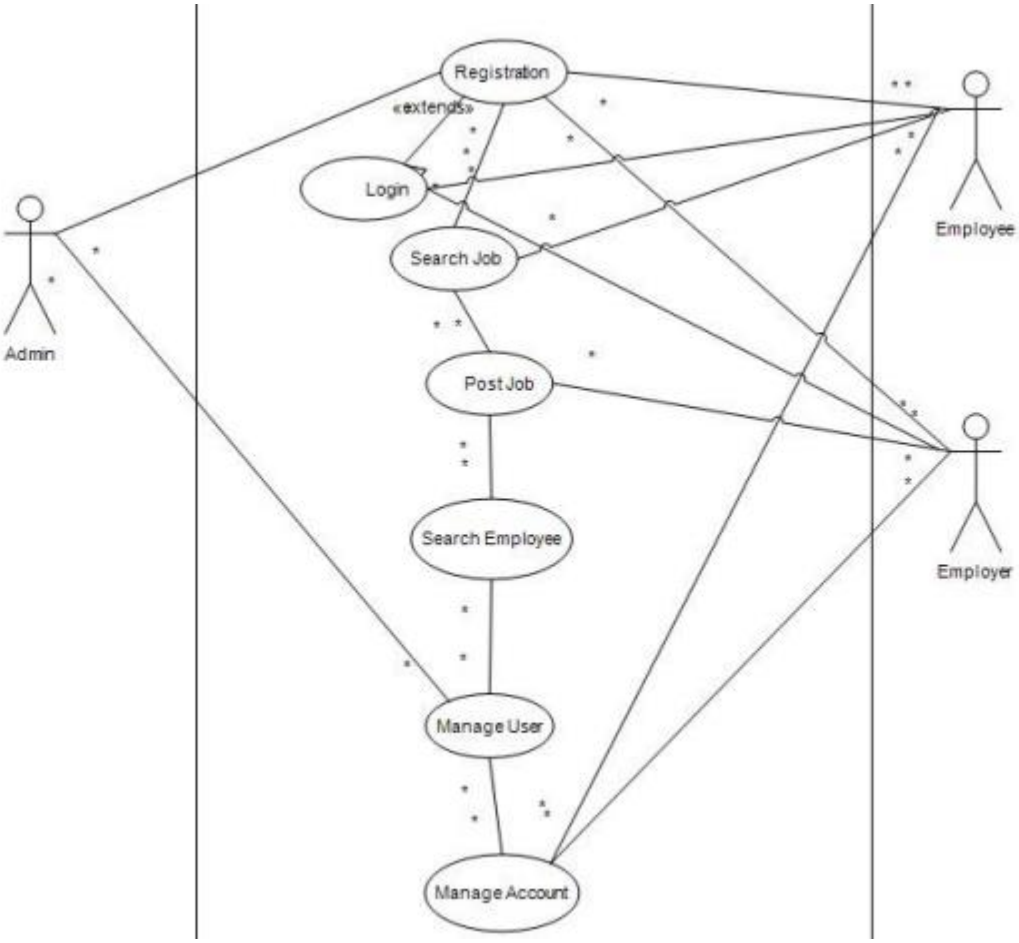
System Analysis refers into the process of examining a situation with the intent of improving it through better procedures and methods. System Analysis is the process of planning a new System to either replace or complement an existing system.

DFD

The DFD is an excellent communication tool for analysts to model processes and functional requirements. Used effectively, it is a useful and easy to understand modeling tool. It has broad application



DFD : MAINTAIN STUDENT RECORD



MODULE DESCRIPTION

These are the following Modules in the project:-

1. **EMPLOYEE**:- This module is used to manage information about employees (job seekers). It stores their personal details such as name, address, qualifications, skills, experience, and resume. Employees can register, update their profile, search for jobs, apply to job postings, and track their application status from this module.

2. **EMPLOYER**:- This module stores the details of employers who post jobs on the portal. It includes organization name, recruiter details, and login credentials. Employers can post new job vacancies, view applications received, filter suitable candidates, and shortlist or reject applications.

3. **JOB MANAGEMENT**: - This module handles all job-related data. Employers can add, update, or remove job postings. Each job post includes details like job title, description, location, qualifications, experience required, and application deadline.

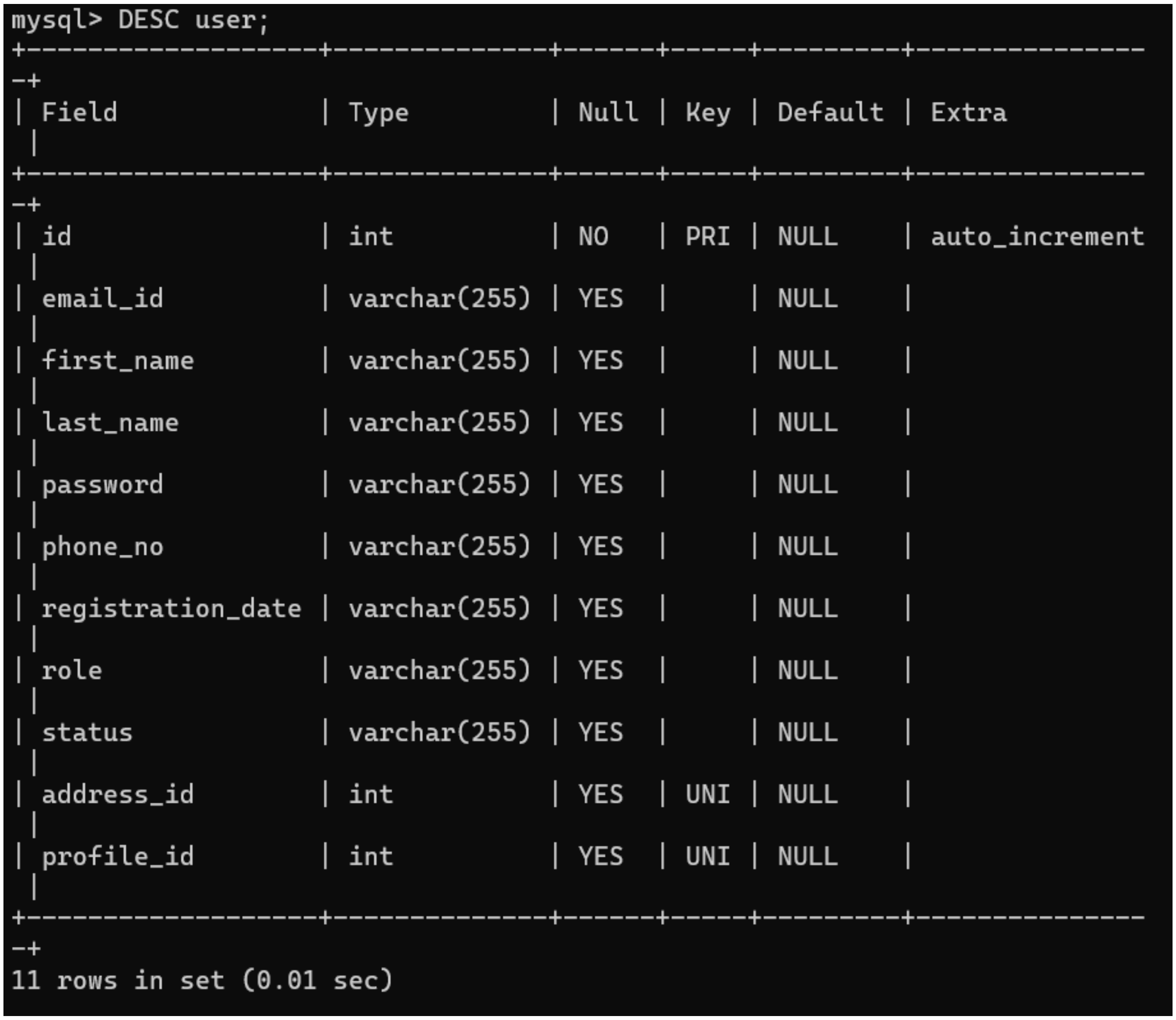
4. **APPLICATION TRACKING**:- This module keeps track of all job applications submitted by employees. It allows employers to update the status of each application (e.g., Under Review, Shortlisted, Rejected), and employees can view the status of their applied jobs..

5. **SEARCH AND FILTER**:- This module allows employees to search for jobs using different criteria like job title, company name, location, category, and salary range. It helps in finding the most relevant jobs easily.

DATA STRUCTURE SNAP SHOT

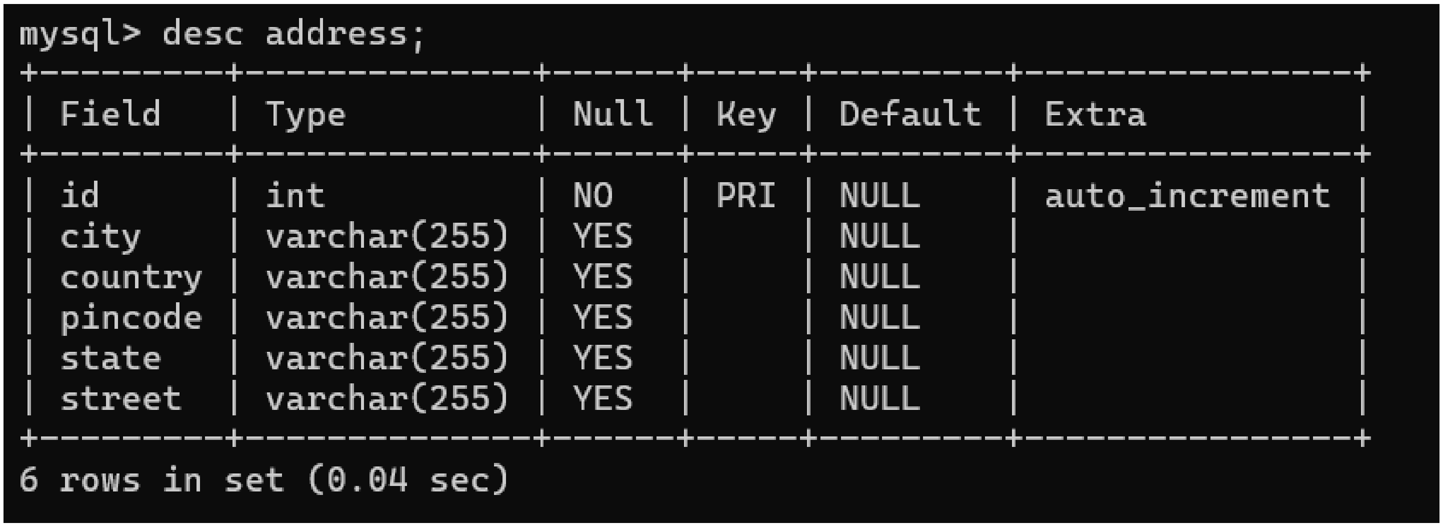
DATA DICTIONERY:- The data dictionary stores description of data items and structures, as well as systems processes. Data dictionary used for database design in this project is mentioned. Care was taken during the database design for future expansion of school's need.

User_ Details



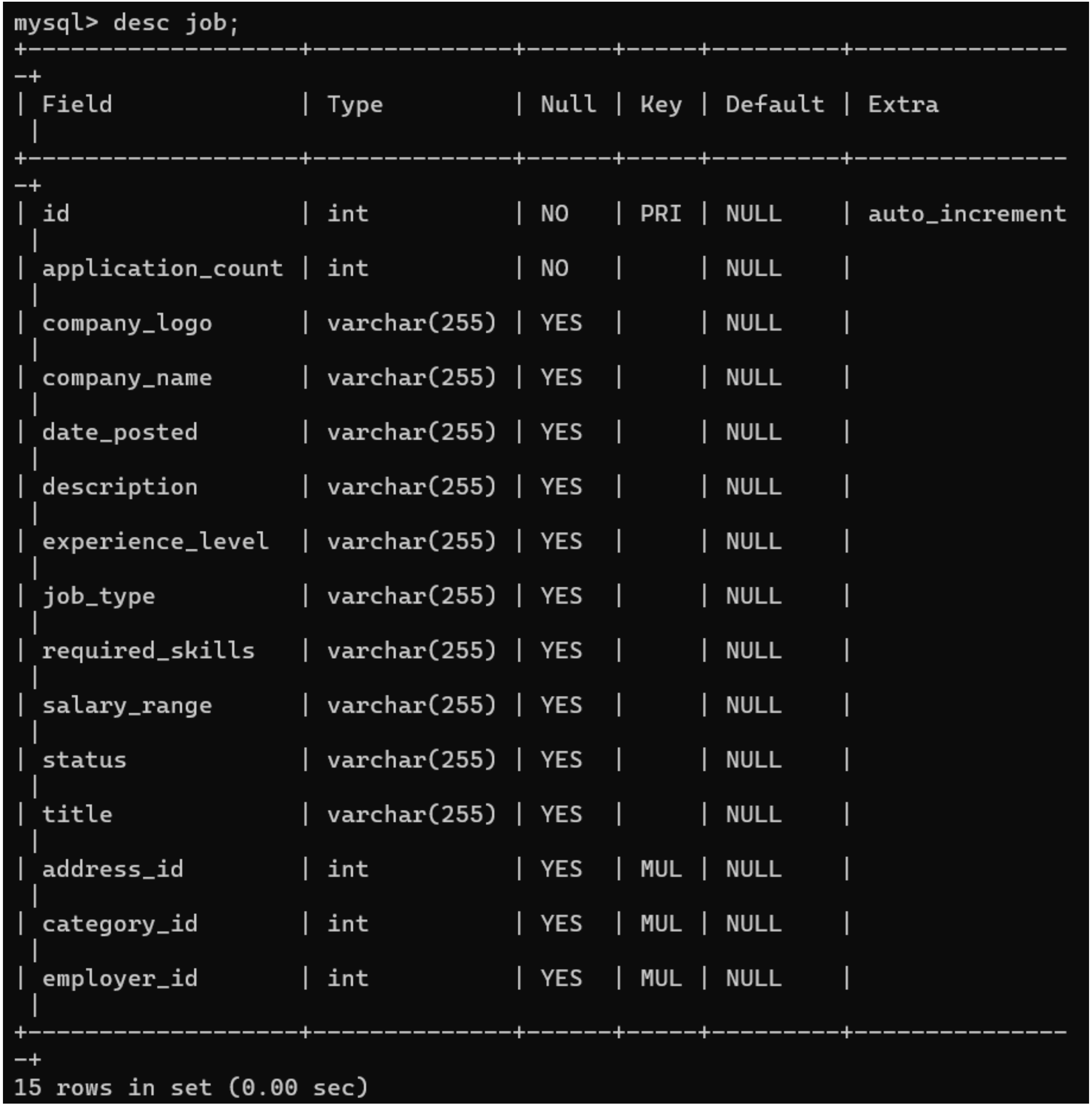
```
mysql> DESC user;
+-------------------+--------------+------+-----+---------+---------------
-+
| Field             | Type         | Null | Key | Default | Extra
 |
+-------------------+--------------+------+-----+---------+---------------
-+
| id                | int          | NO   | PRI | NULL    | auto_increment
 |
| email_id          | varchar(255) | YES  |     | NULL    |
 |
| first_name        | varchar(255) | YES  |     | NULL    |
 |
| last_name         | varchar(255) | YES  |     | NULL    |
 |
| password          | varchar(255) | YES  |     | NULL    |
 |
| phone_no          | varchar(255) | YES  |     | NULL    |
 |
| registration_date | varchar(255) | YES  |     | NULL    |
 |
| role              | varchar(255) | YES  |     | NULL    |
 |
| status            | varchar(255) | YES  |     | NULL    |
 |
| address_id        | int          | YES  | UNI | NULL    |
 |
| profile_id        | int          | YES  | UNI | NULL    |
 |
+-------------------+--------------+------+-----+---------+---------------
-+
11 rows in set (0.01 sec)
```

Address Details (add_address):



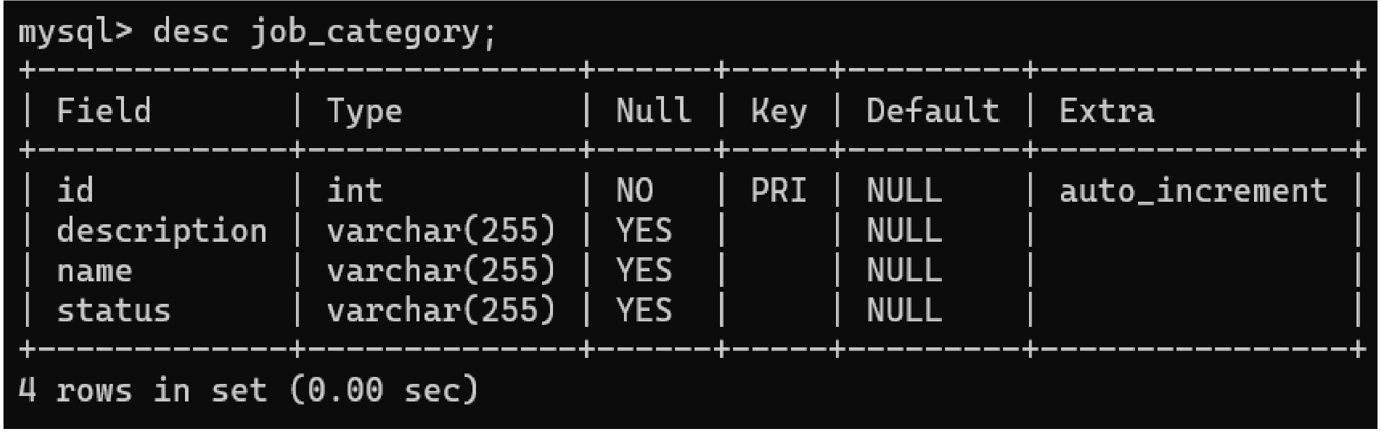
```
mysql> desc address;
+---------+--------------+------+-----+---------+----------------+
| Field   | Type         | Null | Key | Default | Extra          |
+---------+--------------+------+-----+---------+----------------+
| id      | int          | NO   | PRI | NULL    | auto_increment |
| city    | varchar(255) | YES  |     | NULL    |                |
| country | varchar(255) | YES  |     | NULL    |                |
| pincode | varchar(255) | YES  |     | NULL    |                |
| state   | varchar(255) | YES  |     | NULL    |                |
| street  | varchar(255) | YES  |     | NULL    |                |
+---------+--------------+------+-----+---------+----------------+
6 rows in set (0.04 sec)
```

Job Details:-



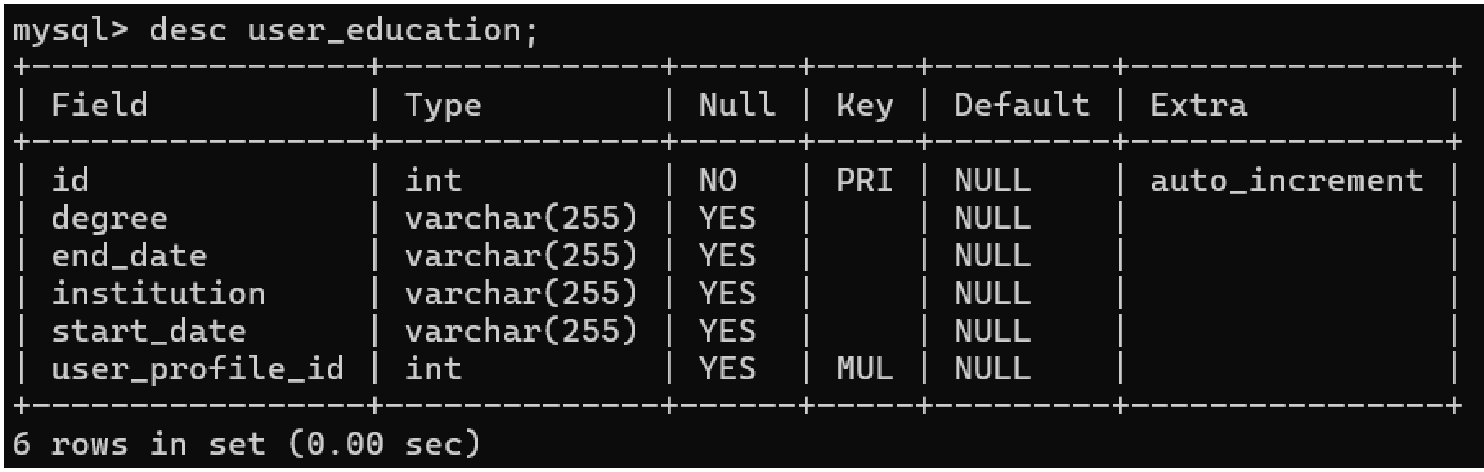
```
mysql> desc job;
+-------------------+--------------+------+-----+---------+---------------
-+
| Field             | Type         | Null | Key | Default | Extra
 |
+-------------------+--------------+------+-----+---------+---------------
-+
| id                | int          | NO   | PRI | NULL    | auto_increment
 |
| application_count | int          | NO   |     | NULL    |
 |
| company_logo      | varchar(255) | YES  |     | NULL    |
 |
| company_name      | varchar(255) | YES  |     | NULL    |
 |
| date_posted       | varchar(255) | YES  |     | NULL    |
 |
| description       | varchar(255) | YES  |     | NULL    |
 |
| experience_level  | varchar(255) | YES  |     | NULL    |
 |
| job_type          | varchar(255) | YES  |     | NULL    |
 |
| required_skills   | varchar(255) | YES  |     | NULL    |
 |
| salary_range      | varchar(255) | YES  |     | NULL    |
 |
| status            | varchar(255) | YES  |     | NULL    |
 |
| title             | varchar(255) | YES  |     | NULL    |
 |
| address_id        | int          | YES  | MUL | NULL    |
 |
| category_id       | int          | YES  | MUL | NULL    |
 |
| employer_id       | int          | YES  | MUL | NULL    |
 |
+-------------------+--------------+------+-----+---------+---------------
-+
15 rows in set (0.00 sec)
```

Job Category



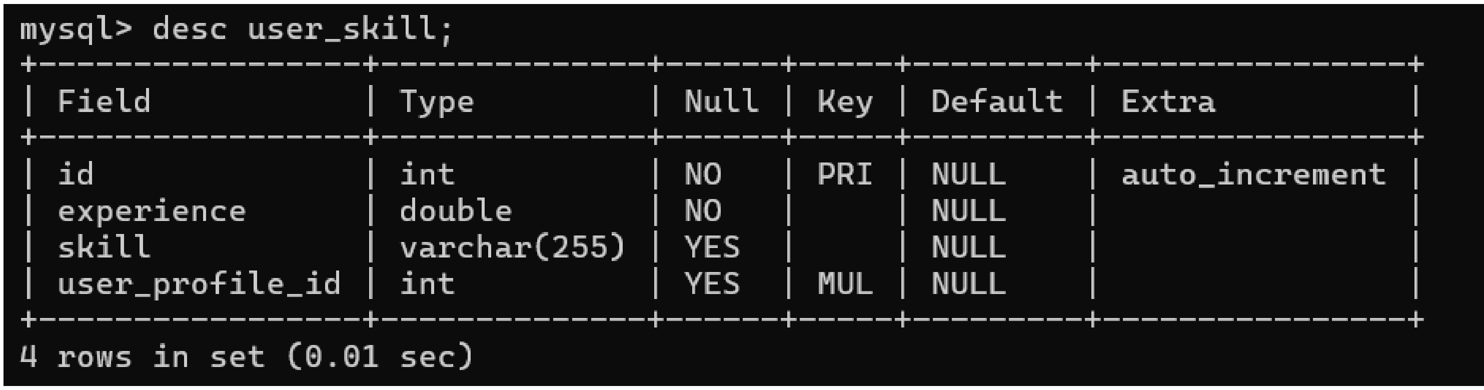
```
mysql> desc job_category;
+-------------+--------------+------+-----+---------+----------------+
| Field       | Type         | Null | Key | Default | Extra          |
+-------------+--------------+------+-----+---------+----------------+
| id          | int          | NO   | PRI | NULL    | auto_increment |
| description | varchar(255) | YES  |     | NULL    |                |
| name        | varchar(255) | YES  |     | NULL    |                |
| status      | varchar(255) | YES  |     | NULL    |                |
+-------------+--------------+------+-----+---------+----------------+
4 rows in set (0.00 sec)
```

User_education:



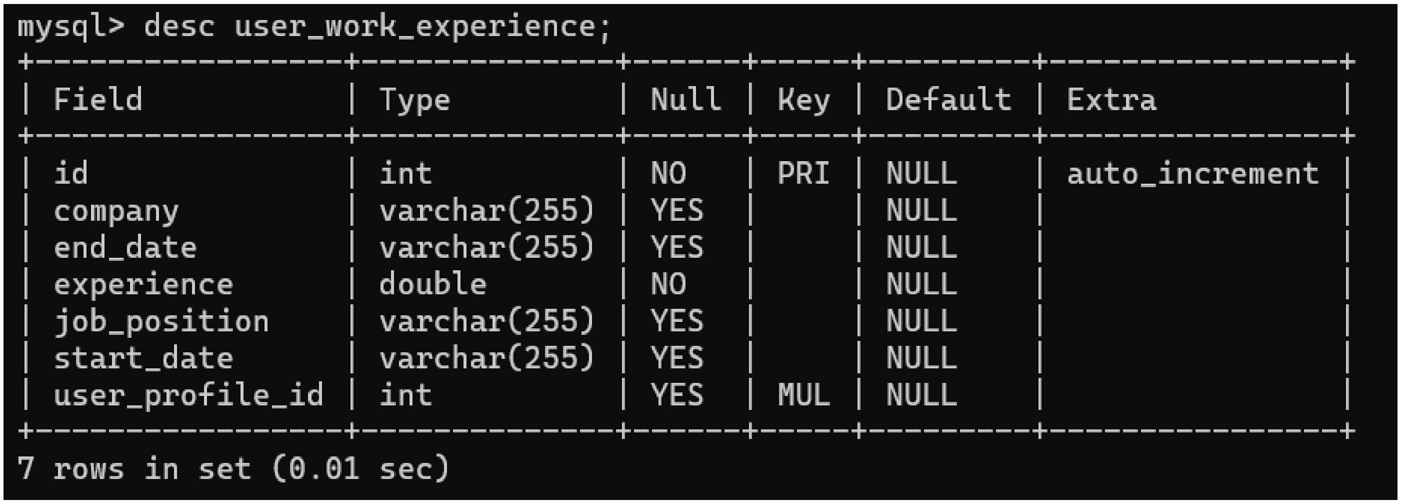
```
mysql> desc user_education;
+-----------------+--------------+------+-----+---------+----------------+
| Field           | Type         | Null | Key | Default | Extra          |
+-----------------+--------------+------+-----+---------+----------------+
| id              | int          | NO   | PRI | NULL    | auto_increment |
| degree          | varchar(255) | YES  |     | NULL    |                |
| end_date        | varchar(255) | YES  |     | NULL    |                |
| institution     | varchar(255) | YES  |     | NULL    |                |
| start_date      | varchar(255) | YES  |     | NULL    |                |
| user_profile_id | int          | YES  | MUL | NULL    |                |
+-----------------+--------------+------+-----+---------+----------------+
6 rows in set (0.00 sec)
```

User Skill



```
mysql> desc user_skill;
+-----------------+--------------+------+-----+---------+----------------+
| Field           | Type         | Null | Key | Default | Extra          |
+-----------------+--------------+------+-----+---------+----------------+
| id              | int          | NO   | PRI | NULL    | auto_increment |
| experience      | double       | NO   |     | NULL    |                |
| skill           | varchar(255) | YES  |     | NULL    |                |
| user_profile_id | int          | YES  | MUL | NULL    |                |
+-----------------+--------------+------+-----+---------+----------------+
4 rows in set (0.01 sec)
```

Work Experience



```
mysql> desc user_work_experience;
+-----------------+--------------+------+-----+---------+----------------+
| Field           | Type         | Null | Key | Default | Extra          |
+-----------------+--------------+------+-----+---------+----------------+
| id              | int          | NO   | PRI | NULL    | auto_increment |
| company         | varchar(255) | YES  |     | NULL    |                |
| end_date        | varchar(255) | YES  |     | NULL    |                |
| experience      | double       | NO   |     | NULL    |                |
| job_position    | varchar(255) | YES  |     | NULL    |                |
| start_date      | varchar(255) | YES  |     | NULL    |                |
| user_profile_id | int          | YES  | MUL | NULL    |                |
+-----------------+--------------+------+-----+---------+----------------+
7 rows in set (0.01 sec)
```

LIST OF REPORTS

Employee Registration Report- Shows the list of all registered employees with their profiles, skills, and registration dates.

Employer Registration Report- Displays all registered employers along with company details and contact information.

Job Posting Report- Contains all job vacancies posted by employers including job title, type, location, and posted date.

Application Report- Lists all job applications submitted by employees with application date, job title, and current status.

Job-wise Application Report- Shows the number of applications received for each job post and their status (Applied, Shortlisted, Rejected, etc.).

Employee-wise Application Report- Displays the list of jobs applied by a specific employee and their current status.

Employer-wise Job Report- Lists all the jobs posted by a particular employer along with posting and deadline dates.

SCOPE AND FUTURE ENHANCEMENT

The Online Job Portal System aims to provide a centralized digital platform that bridges the gap between employees (job seekers) and employers (recruiters). It simplifies the recruitment process by enabling employers to post job vacancies and employees to apply for jobs through an interactive web interface. The system supports user registration, job management, resume handling, application tracking, and notification services. It is designed to be scalable, user-friendly, and accessible from any device with internet connectivity.

- Interview Scheduling System

- Chat and Messaging System
- Job Recommendation Engine
- Video Resume Upload
- Integration with LinkedIn/Other Platforms

IMPLEMENTATION METHODOLOGY

Implementation of a tested system is done on the actual location with the actual hardware, Software and manpower platforms or limitation. To implement of this software, the followings are the necessary:

This project run on internet or intranet , so I have to selectedinternet supportive software. For implementing this software a web- server is required . I have to install IIS(internet information service). Know this project is web-based so it will need a web-browser which will act as its interface, so for this project install a web-browser capableof providing proper interface to this project such as internet Explorer/mozilla Firefox/opera e.t.c and many other browser.and for back-end operation ,I have to install ORACLE 9i which handle all the work related to database query and updation as required by the sostware. In future also use better databases.we know that this project will be developed on heavy platform ASPJAVA with c# , so there is requirement of advance processer and proper RAM and other advance Hardware configuration.

- ✓ Create separate user for this application: A user named as “admin” with the password “pass” is to be created for storing the data in a separate place. Storing data under separate user will protect the data. Oracle Navigator can be used for creating the said username and password.
- ✓ Create required tables: The structure of tables required for this project (Detail given in the data base design section) is to be created in the said user. The SQL statement CREATE can be used for creating such tables.

- ✓ Create Distributable set of files: An installation package of the software is to be built for implementation of the project at desired location.

TESTING

Software testing consists of the following stages.

a) TEST PLAN DEVELOPMENT:-

The importance of software testing and its implications need not be emphasized. Software testing is a critical element of software quality assurance and represents the ultimate review of specifications, design and coding. The main objective of testing is to uncover a host oferrors, systematically, with minimum effort and time. It improves the reliability of the software.

b) TESTING STRATEGY:-

A testing strategy explains how to conduct a test to uncover all the errors with minimum possible time and effort. It should be flexible enough to promote customization that may be necessary in due course of development process.

c) UNIT TESTING:-

Unit testing focuses the verification effort on the smallest unit of software modules using detailed design specifications. It is performed to uncover errors within the boundaries of individual modules. All major modules like Employee Registration, Employer Login, Job

Posting, and Application Submission were tested independently to ensure that each component functions correctly.

INTEGRATION TESTING:-

Integration testing is performed after unit testing, where different modules are combined and tested as a group. The objective is to check the data flow between modules. For example, after posting a job (Employer module), it should correctly reflect in the Job Search section of the Employee module.

VALIDATION TESTING:-

Validation Testing is performed to ensure that the developed system meets all the specified functional and non-functional requirements. It confirms that the right system has been built for the intended users.

Validation testing checks whether the **Online Job Portal System** behaves as expected when provided with valid and invalid inputs. Each form and module in the system is validated to avoid incorrect or incomplete data entries.

TEST CASE DESIGN

The techniques that are used in deriving the test cases are explained below.

CONDITION TESTING:-

Condition testing is a used to check the logical conditions and decision-making capabilities within the software. It ensures that all possible outcomes of conditions (e.g., if-else, switch-case) are tested to validate correct behavior under different scenarios.

This testing is crucial in verifying that the system responds correctly based on different user actions, inputs, or system states.

BOUNDARY VALUE ANALYSIS:-

Boundary value analysis is a software testing technique where test cases are designed to include values at the boundaries. This is based on the observation that many errors occur at the edges of input ranges rather than in the middle.

In the **Online Job Portal System**, BVA is applied to input fields such as age, experience, password length, and contact number to ensure that the system correctly handles boundary values.

EQUIVALENCE PARTITIONING:-

Equivalence portioning is a software testing technique that divides input data into **valid** and **invalid partitions** (classes) so that test cases can be designed to cover each partition at least once. The idea is that if one value in a class works, all other values in that class will also behave the same way.

LIMITATIONSANDFURTHER ENHANCEMENTS

LIMITATIONS:

Despite tremendous growth potentialities, the online job portal continues to face a series of challenges and contradictions in its implementation. The main shortcomings are mentioned below:

- Lack of authenticity of job postings.
- Difficulty in verifying employer and candidate profiles.
- Data privacy and security concerns.

RECOMMENDED AREAS FOR FURTHER RESEARCH

Considering the fact that has already gone ahead the following areas needs to be further probed for improving the system to meet the changing scenario.

1) There is a lot of scope for future research in areas such as intelligent resume screening, automated interview scheduling, real-time job alerts, and candidate behavioral analysis through AI-based systems.
2) Components like automated skill assessment, video resume integration, digital badges for verified skills, and AI-driven job recommendations are still in their early stages of development

SECURITY MECHANISM

Implementation of security is a key issue in any software development different application use different kind of security mechanism according to their need.In Ngo Management Systemthe implementation of security has been confined up to the user level. The "Ngo Management System" authenticates a user by its username and password to a user. A user is able to work on the software if s/he is an authorized user with user id and password.

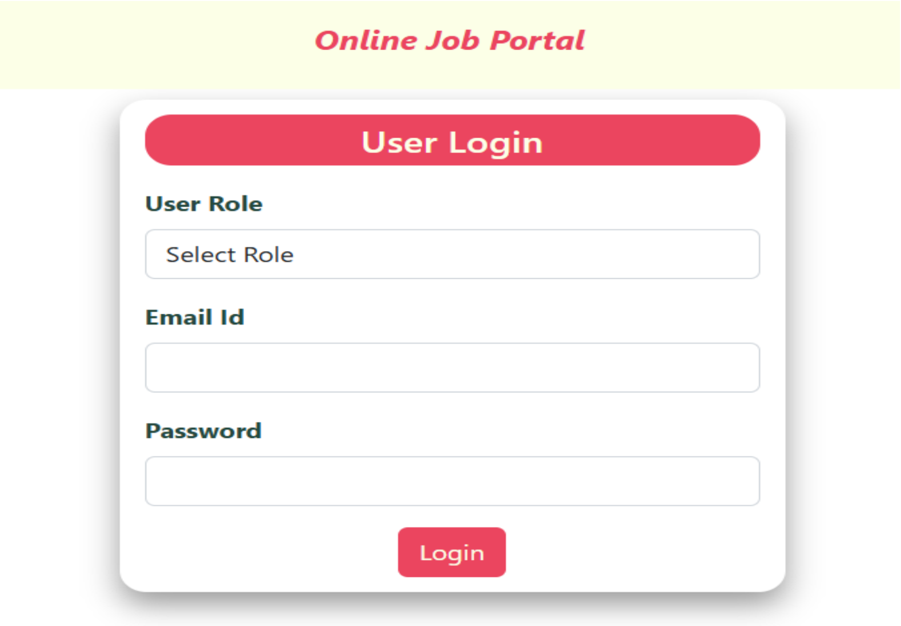
It is the system administrator who is responsible for granting permission to any user. He can grant as well as revoke permissions from any user.Another important security level is data level. In the data level security defined validation and constraints in the database. Due to this only valid data can be entered in the tables.

All information and management forms are stored within the system, and access to this sensitive information is controlled through an effective authentication system: one has to know the EMAIL ID, and PASSWORD to gain access to any of the information.

The management (administrator) can create new user, while a normal user can't do this. This system will provide the facility to change the password according to the user need. For security measure the 'Password' field will be taken as 'Varchar2' data type, so that a user can use in this field both integer and alphabets.

SNAPSHOTS

USER LOGIN FORM (login.jsx):-



code for login page :-

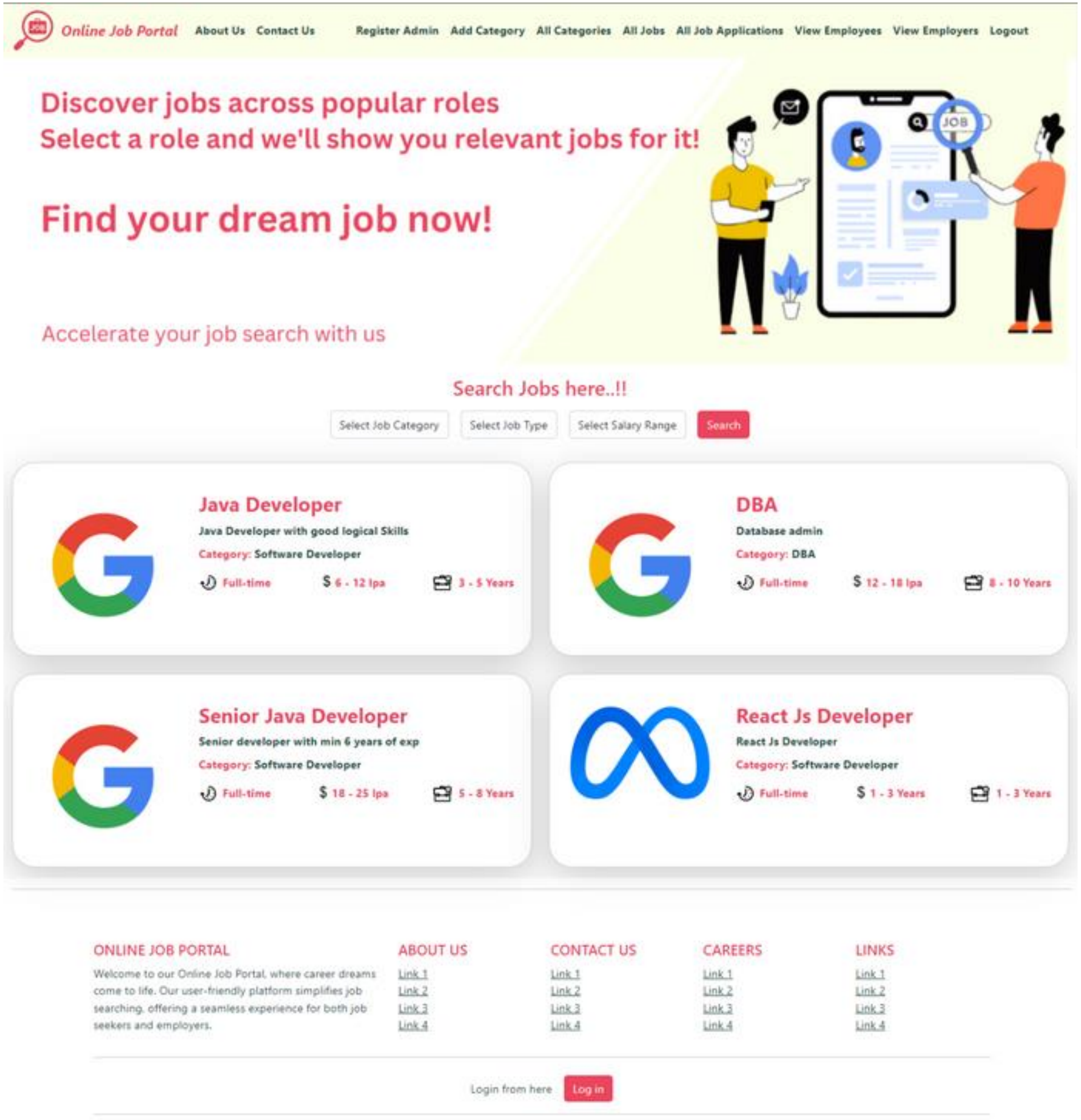
```
import { useState } from "react";
import { ToastContainer, toast } from "react-toastify";
import "react-toastify/dist/ReactToastify.css";
import { useNavigate } from "react-router-dom";

const UserLoginForm = () => {
  let navigate = useNavigate();

  const [loginRequest, setLoginRequest] = useState({
    emailId: "",
    password: "",
    role: "",
  });

  const handleUserInput = (e) => {
    setLoginRequest({ ...loginRequest, [e.target.name]: e.target.value });
  };
```

```
const loginAction = (e) => {
  fetch("http://localhost:8080/api/user/login", {
    method: "POST",
    headers: {
      Accept: "application/json",
      "Content-Type": "application/json",
    },
    body: JSON.stringify(loginRequest),
  })
    .then((result) => {
      console.log("result", result);
      result.json().then((res) => {
        if (res.success) {
          console.log("Got the success response");

          if (res.jwtToken !== null) {
            if (res.user.role === "Admin") {
              sessionStorage.setItem(
                "active-admin",
                JSON.stringify(res.user)
              );
              sessionStorage.setItem("admin-jwtToken", res.jwtToken);
            } else if (res.user.role === "Employer") {
              sessionStorage.setItem(
                "active-employer",
                JSON.stringify(res.user)
              );
              sessionStorage.setItem("employer-jwtToken", res.jwtToken);
            } else if (res.user.role === "Employee") {
              sessionStorage.setItem(
                "active-employee",
                JSON.stringify(res.user)
              );
              sessionStorage.setItem("employee-jwtToken", res.jwtToken);
            }
          }

          if (res.jwtToken !== null) {
            toast.success(res.responseMessage, {
              position: "top-center",
              autoClose: 1000,
              hideProgressBar: false,
              closeOnClick: true,
              pauseOnHover: true,
              draggable: true,
              progress: undefined,
            });
            setTimeout(() => {
              window.location.href = "/home";
            }, 1000); // Redirect after 3 seconds
          } else {
            toast.error(res.responseMessage, {
              position: "top-center",
              autoClose: 1000,
              hideProgressBar: false,
              closeOnClick: true,
              pauseOnHover: true,
              draggable: true,
              progress: undefined,
            });
          }
        } else {
          toast.error(res.responseMessage, {
            position: "top-center",
            autoClose: 1000,
            hideProgressBar: false,
            closeOnClick: true,
            pauseOnHover: true,
            draggable: true,
            progress: undefined,
          });
        }
      });
    })
    .catch((error) => {
      console.error(error);
      toast.error("It seems server is down", {
        position: "top-center",
        autoClose: 1000,
        hideProgressBar: false,
        closeOnClick: true,
        pauseOnHover: true,
        draggable: true,
        progress: undefined,
      });
    });
```

```
    e.preventDefault();
  };

  return (
    <div>
      <div className="mt-2 d-flex aligns-items-center justify-content-center">
        <div className="form-card border-color" style={{ width: "25rem" }}>
          <div className="container-fluid">
            <div
              className="card-header bg-color custom-bg-text mt-2 d-flex justify-content-center align-items-
center"
              style={{
                borderRadius: "1em",
                height: "38px",
              }}
            >
              <h4 className="card-title">User Login</h4>
            </div>
            <div className="card-body mt-3">
              <form>
                <div class="mb-3 text-color">
                  <label for="role" class="form-label">
                    <b>User Role</b>
                  </label>
                  <select
                    onChange={handleUserInput}
                    className="form-control"
                    name="role"
                  >
                    <option value="0">Select Role</option>
                    <option value="Admin"> Admin </option>
                    <option value="Employer"> Employer </option>
                    <option value="Employee"> Employee </option>
                  </select>
                </div>

                <div className="mb-3 text-color">
                  <label for="emailId" class="form-label">
                    <b>Email Id</b>
                  </label>
                  <input
                    type="email"
                    className="form-control"
                id="emailId"
                name="emailId"
                onChange={handleUserInput}
                value={loginRequest.emailId}
               />
              </div>
              <div className="mb-3 text-color">
               <label for="password" className="form-label">
                <b>Password</b>
               </label>
               <input
                type="password"
                className="form-control"
                id="password"
                name="password"
                onChange={handleUserInput}
                value={loginRequest.password}
                autoComplete="on"
               />
              </div>
              <div className="d-flex aligns-items-center justify-content-center mb-2">
               <button
                type="submit"
                className="btn bg-color custom-bg-text"
                onClick={loginAction}
               >
                Login
               </button>
               <ToastContainer />
              </div>
             </form>
            </div>
           </div>
          </div>
         </div>
        </div>
       );
      };

      export default UserLoginForm;
```

```
package com.jobportal.dto;

public class UserLoginRequest {
    private String emailId;

    private String password;

    private String role;

    public String getEmailId() {
      return emailId;
    }

    public void setEmailId(String emailId) {
      this.emailId = emailId;
    }

    public String getPassword() {
      return password;
    }

    public void setPassword(String password) {
      this.password = password;
    }

    public String getRole() {
      return role;
    }

    public void setRole(String role) {
      this.role = role;
    }

}.

package com.jobportal.dto;

import com.jobportal.entity.User;

public class UserLoginResponse extends CommonApiResponse {

    private User user;

    private String jwtToken;

    public User getUser() {
      return user;
    }

    public void setUser(User user) {
        this.user = user;
    }

    public String getJwtToken() {
        return jwtToken;
    }

    public void setJwtToken(String jwtToken) {
        this.jwtToken = jwtToken;
    }

}
```

Home Page:-



Discover jobs across popular roles
Select a role and we'll show you relevant jobs for it!

Find your dream job now!

Accelerate your job search with us

Search Jobs here..!!

Java Developer
Java Developer with good logical Skills
Category: Software Developer
Full-time $ 6 - 12 lpa 3 - 5 Years

DBA
Database admin
Category: DBA
Full-time $ 12 - 18 lpa 8 - 10 Years

Senior Java Developer
Senior developer with min 6 years of exp
Category: Software Developer
Full-time $ 18 - 25 lpa 5 - 8 Years

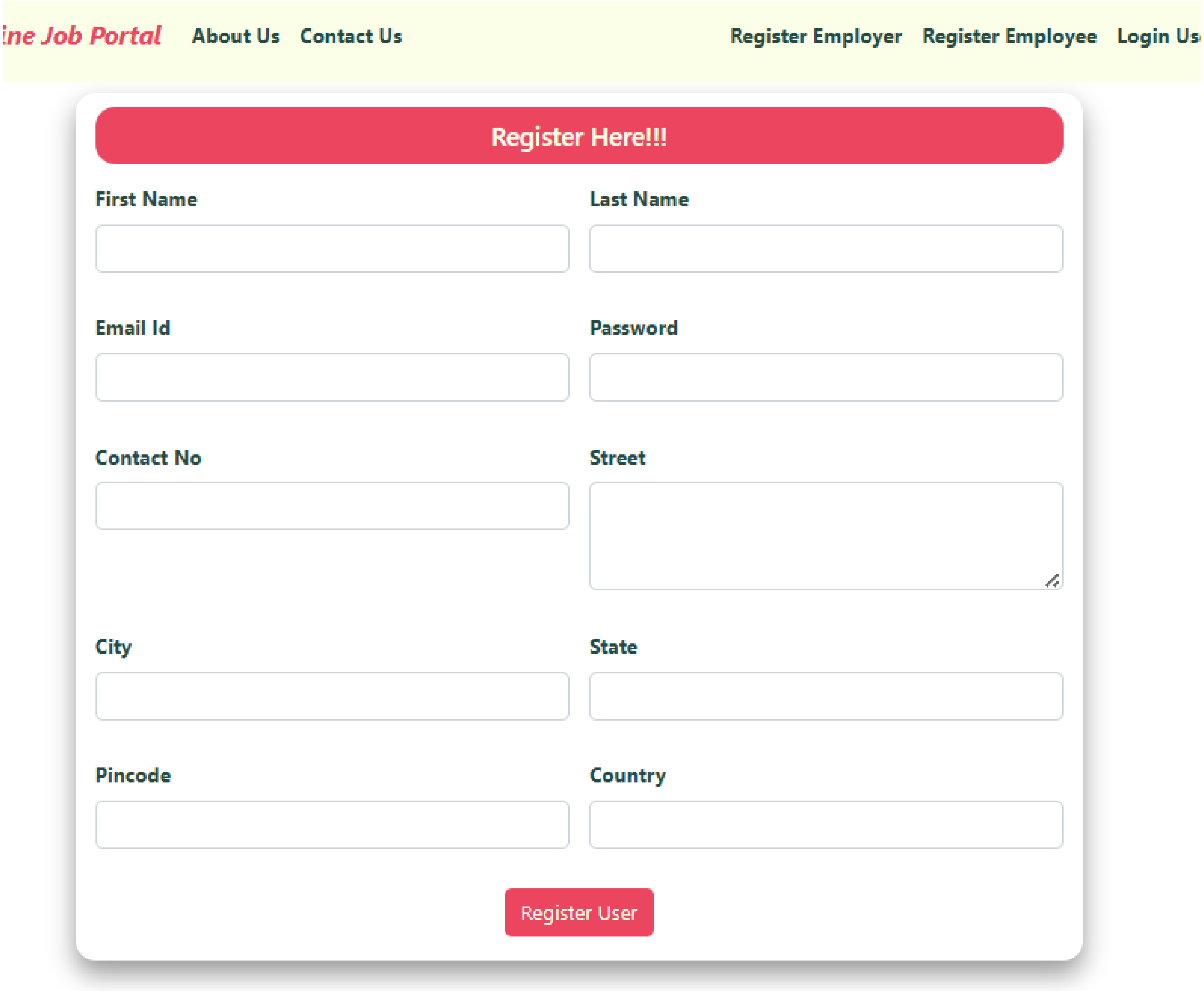
React Js Developer
React Js Developer
Category: Software Developer
Full-time $ 1 - 3 Years 1 - 3 Years

ONLINE JOB PORTAL
Welcome to our Online Job Portal, where career dreams come to life. Our user-friendly platform simplifies job searching, offering a seamless experience for both job seekers and employers.

```
              /**
               * Creates new form home
               */
import React, { useState, useEffect } from "react";
import { useParams } from "react-router-dom";
import axios from "axios";
import Carousel from "./Carousel";
import Footer from "../NavbarComponent/Footer";
import { useNavigate } from "react-router-dom";
import JobCard from "../JobComponent/JobCard";

const HomePage = () => {
  const navigate = useNavigate();

  const [allJobs, setAllJobs] = useState([
    {
      id: "",
      employer: {
        id: "",
        firstName: "",
        lastName: "",
        emailId: "",
        phoneNo: "",
        role: "",
        address: {
          id: "",
          street: "",
          city: "",
          pincode: "",
          state: "",
          country: "",
        },
        registrationDate: "",
        status: "",
      },
      title: "",
      description: "",
      category: {
        id: "",
        name: "",
        description: "",
        status: "",
      },
      companyName: "",
      companyLogo: "",
      address: {
        id: "",
        street: "",
        city: "",
        pincode: "",
        state: "",
        country: "",
      },
      jobType: "",
      salaryRange: "",
      experienceLevel: "",
      requiredSkills: "",
      status: "",
      datePosted: "",
      applicationCount: "",
    },
  ]);

  const [searchData, setSearchData] = useState({
    categoryId: "",
    jobType: "",
    salaryRange: "",
  });

  const [tempSearchData, setTempSearchData] = useState({
    categoryId: "",
    jobType: "",
    salaryRange: "",
  });

  const [categories, setCategories] = useState([]);

  const [jobTypes, setJobTypes] = useState([]);
  const [salaryRange, setSalaryRange] = useState([]);

  const handleUserInput = (e) => {
    setTempSearchData({ ...tempSearchData, [e.target.name]: e.target.value });
  };

  const retrieveAllCategories = async () => {
    const response = await axios.get(
      "http://localhost:8080/api/job/category/fetch/all"
    );
    return response.data;
  };
  const retrieveAllJobTypes = async () => {
    const response = await axios.get(
      "http://localhost:8080/api/helper/job/type/fetch/all"
    );
    return response.data;
  };
  const retrieveAllSalary = async () => {
    const response = await axios.get(
      "http://localhost:8080/api/helper/job/salary/range/fetch/all"
    );
    return response.data;
  };

  useEffect(() => {
    const getAllJobs = async () => {
      const allJobs = await retrieveAllJobs();
      if (allJobs) {
        setAllJobs(allJobs.jobs);
      }
    };

    const getSearchedJobs = async () => {
      const allJobs = await searchJobsByData();
      if (allJobs) {
        setAllJobs(allJobs.jobs);
      }
    };

    const getAllCategories = async () => {
      const resCategory = await retrieveAllCategories();
      if (resCategory) {
        setCategories(resCategory.categories);
      }
    };

    const getAllJobTypes = async () => {
      const res = await retrieveAllJobTypes();
      if (res) {
        setJobTypes(res);
      }
    };

    const getAllSalaryRange = async () => {
      const res = await retrieveAllSalary();
      if (res) {
        setSalaryRange(res);
      }
    };

    if (
      searchData.categoryId !== "" &&
      searchData.jobType !== "" &&
      searchData.salaryRange !== ""
    ) {
      getSearchedJobs();
    } else {
      getAllJobs();
    }

    getAllJobTypes();
    getAllSalaryRange();
    getAllCategories();
  }, [searchData]);

  const retrieveAllJobs = async () => {
    const response = await axios.get("http://localhost:8080/api/job/fetch/all");
    console.log(response.data);
    return response.data;
  };

  const searchJobsByData = async () => {
    const response = await axios.get(
      "http://localhost:8080/api/job/search?categoryId=" +
        searchData.categoryId +
        "&jobType=" +
        searchData.jobType +
        "&salaryRange=" +
        searchData.salaryRange
    );
    console.log(response.data);
    return response.data;
  };

  const searchJob = (e) => {
    e.preventDefault();
    setSearchData(tempSearchData);
  };

  return (
    <div className="container-fluid mb-2">
      <Carousel />
      <h3 className="text-color-second text-center mt-3">
        Search Jobs here..!!
      </h3>
      <div className="d-flex aligns-items-center justify-content-center mt-3">
        <form class="row g-3">
          <div class="col-auto">
            <select
              name="categoryId"
              onChange={handleUserInput}
              className="form-control"
              required
            >
              <option value="">Select Job Category</option>

              {categories.map((category) => {
                return <option value={category.id}> {category.name} </option>;
              })}
            </select>
          </div>

          <div class="col-auto">
            <select
              name="jobType"
              onChange={handleUserInput}
              className="form-control"
              required
            >
              <option value="">Select Job Type</option>

              {jobTypes.map((type) => {
                return <option value={type}> {type} </option>;
              })}
            </select>
          </div>

          <div class="col-auto">
            <select
              name="salaryRange"
              onChange={handleUserInput}
              className="form-control"
              required
            >
              <option value="">Select Salary Range</option>

              {salaryRange.map((range) => {
                return <option value={range}> {range} </option>;
              })}
            </select>
          </div>

          <div class="col-auto">
            <button
              type="submit"
              class="btn bg-color custom-bg-text mb-3"
              onClick={searchJob}
            >
              Search
```

```
            </button>
          </div>
        </form>
      </div>

      <div className="col-md-12 mt-3 mb-5">
        <div className="row row-cols-1 row-cols-md-2 g-4">
          {allJobs.map((job) => {
            return <JobCard item={job} key={job.id} />;
          })}
        </div>
      </div>
      <hr />
      <Footer />
    </div>
  );
};

export default HomePage;
```

Registration Form Details :



code :

```
import { useState, useEffect } from "react";
import "react-toastify/dist/ReactToastify.css";
import { ToastContainer, toast } from "react-toastify";
import { useNavigate } from "react-router-dom";
```

```
const UserRegister = () => {
  const navigate = useNavigate();

  const [user, setUser] = useState({
    firstName: "",
    lastName: "",
    emailId: "",
    password: "",
    phoneNo: "",
    street: "",
    city: "",
    pincode: "",
    role: "",
    state: "",
    country: "",
  });

  useEffect(() => {
    if (document.URL.indexOf("employee") != -1) {
      user.role = "Employee";
    } else if (document.URL.indexOf("employer") != -1) {
      user.role = "Employer";
    }
  }, []);

  const handleUserInput = (e) => {
    setUser({ ...user, [e.target.name]: e.target.value });
  };

  const saveUser = (e) => {
    e.preventDefault();

    let jwtToken;

    fetch("http://localhost:8080/api/user/register", {
      method: "POST",
      headers: {
        Accept: "application/json",
        "Content-Type": "application/json",

        //    Authorization: "Bearer " + jwtToken,
      },
      body: JSON.stringify(user),
    })
      .then((result) => {
        console.log("result", result);
        result.json().then((res) => {
          if (res.success) {
            toast.success(res.responseMessage, {
              position: "top-center",
              autoClose: 1000,
              hideProgressBar: false,
              closeOnClick: true,
              pauseOnHover: true,
              draggable: true,
              progress: undefined,
            });

            setTimeout(() => {
              navigate("/user/login");
            }, 1000);
          } else if (!res.success) {
            toast.error(res.responseMessage, {
              position: "top-center",
              autoClose: 1000,
              hideProgressBar: false,
              closeOnClick: true,
              pauseOnHover: true,
              draggable: true,
              progress: undefined,
            });

            setTimeout(() => {
              window.location.reload(true);
            }, 1000); // Redirect after 3 seconds
          } else {
            toast.error("It seems server is down", {
              position: "top-center",
              autoClose: 1000,
              hideProgressBar: false,
              closeOnClick: true,
              pauseOnHover: true,
              draggable: true,
              progress: undefined,
            });

            setTimeout(() => {
              window.location.reload(true);
            }, 1000); // Redirect after 3 seconds
          }
        });
      })
      .catch((error) => {
        console.error(error);
        toast.error("It seems server is down", {
          position: "top-center",
          autoClose: 1000,
          hideProgressBar: false,
          closeOnClick: true,
          pauseOnHover: true,
          draggable: true,
          progress: undefined,
        });
        setTimeout(() => {
          window.location.reload(true);
        }, 1000); // Redirect after 3 seconds
      });
  };

  return (
    <div>
      <div className="mt-2 d-flex aligns-items-center justify-content-center ms-2 me-2 mb-2">
        <div
          className="form-card border-color text-color"
          style={{ width: "50rem" }}
        >
          <div className="container-fluid">
            <div
              className="card-header bg-color custom-bg-text mt-2 d-flex justify-content-center align-items-center"
              style={{
                borderRadius: "1em",
                height: "45px",
              }}
            >
              <h5 className="card-title">Register Here!!!</h5>
            </div>
```

```
<div className="card-body mt-3">
  <form className="row g-3" onSubmit={saveUser}>
    <div className="col-md-6 mb-3 text-color">
      <label htmlFor="title" className="form-label">
        <b>First Name</b>
      </label>
      <input
        type="text"
        className="form-control"
        id="firstName"
        name="firstName"
        onChange={handleUserInput}
        value={user.firstName}
      />
    </div>

    <div className="col-md-6 mb-3 text-color">
      <label htmlFor="title" className="form-label">
        <b>Last Name</b>
      </label>
      <input
        type="text"
        className="form-control"
        id="lastName"
        name="lastName"
        onChange={handleUserInput}
        value={user.lastName}
      />
    </div>

    <div className="col-md-6 mb-3 text-color">
      <b>
        <label className="form-label">Email Id</label>
      </b>
      <input
        type="email"
        className="form-control"
        id="emailId"
        name="emailId"
        onChange={handleUserInput}
        value={user.emailId}
      />
```

```
</div>
<div className="col-md-6 mb-3">
  <label htmlFor="quantity" className="form-label">
    <b>Password</b>
  </label>
  <input
    type="password"
    className="form-control"
    id="password"
    name="password"
    onChange={handleUserInput}
    value={user.password}
  />
</div>

<div className="col-md-6 mb-3">
  <label htmlFor="contact" className="form-label">
    <b>Contact No</b>
  </label>
  <input
    type="number"
    className="form-control"
    id="phoneNo"
    name="phoneNo"
    onChange={handleUserInput}
    value={user.phoneNo}
  />
</div>

<div className="col-md-6 mb-3">
  <label htmlFor="description" className="form-label">
    <b> Street</b>
  </label>
  <textarea
    className="form-control"
    id="street"
    name="street"
    rows="3"
    onChange={handleUserInput}
    value={user.street}
  />
</div>
```

```
<div className="col-md-6 mb-3">
  <label htmlFor="price" className="form-label">
    <b>City</b>
  </label>
  <input
    type="text"
    className="form-control"
    id="city"
    name="city"
    onChange={handleUserInput}
    value={user.city}
  />
</div>

<div className="col-md-6 mb-3">
  <label htmlFor="price" className="form-label">
    <b>State</b>
  </label>
  <input
    type="text"
    className="form-control"
    id="state"
    name="state"
    onChange={handleUserInput}
    value={user.state}
  />
</div>

<div className="col-md-6 mb-3">
  <label htmlFor="pincode" className="form-label">
    <b>Pincode</b>
  </label>
  <input
    type="number"
    className="form-control"
    id="pincode"
    name="pincode"
    onChange={handleUserInput}
    value={user.pincode}
  />
</div>
```

```
                <div className="col-md-6 mb-3">
                  <label htmlFor="price" className="form-label">
                    <b>Country</b>
                  </label>
                  <input
                    type="text"
                    className="form-control"
                    id="country"
                    name="country"
                    onChange={handleUserInput}
                    value={user.country}
                  />
                </div>

                <div className="d-flex aligns-items-center justify-content-center">
                  <input
                    type="submit"
                    className="btn bg-color custom-bg-text"
                    value="Register User"
                  />
                </div>
                <ToastContainer />
              </form>
            </div>
          </div>
        </div>
      </div>
    </div>
  );
};

export default UserRegister;
```

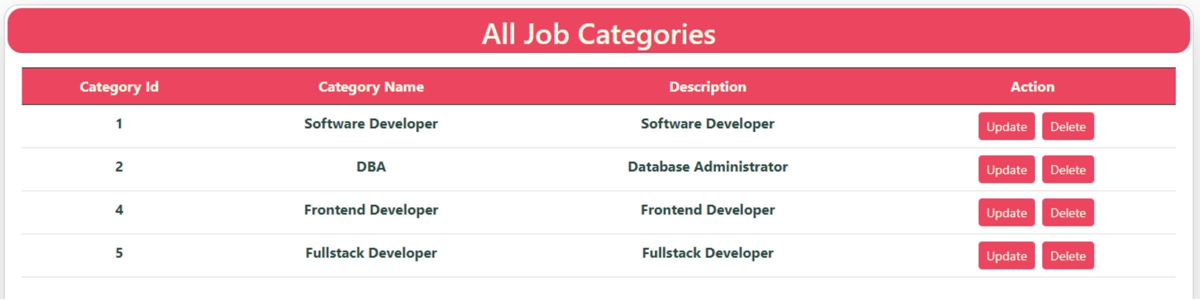
Backend code:-

```
package com.jobportal.dto;

import org.springframework.beans.BeanUtils;

import com.jobportal.entity.Address;
import com.jobportal.entity.User;
```

```
public class RegisterUserRequestDto {

    // user details
    private String firstName;

    private String lastName;

    private String emailId;

    private String password;

    private String phoneNo;

    private String role;

    // user address
    private String street;

    private String city;

    private String pincode;

    private String state;

    private String country;

    public String getFirstName() {
        return firstName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    public String getLastName() {
        return lastName;
    }

    public void setLastName(String lastName) {
        this.lastName = lastName;
    }
    public String getEmailId() {
      return emailId;
    }

    public void setEmailId(String emailId) {
      this.emailId = emailId;
    }

    public String getPassword() {
      return password;
    }

    public void setPassword(String password) {
      this.password = password;
    }

    public String getPhoneNo() {
      return phoneNo;
    }

    public void setPhoneNo(String phoneNo) {
      this.phoneNo = phoneNo;
    }

    public String getRole() {
      return role;
    }

    public void setRole(String role) {
      this.role = role;
    }

    public String getStreet() {
      return street;
    }

    public void setStreet(String street) {
      this.street = street;
    }

    public String getCity() {
      return city;
    }

    public void setCity(String city) {
        this.city = city;
    }

    public String getPincode() {
        return pincode;
    }

    public void setPincode(String pincode) {
        this.pincode = pincode;
    }

    public String getState() {
        return state;
    }

    public void setState(String state) {
        this.state = state;
    }

    public String getCountry() {
        return country;
    }

    public void setCountry(String country) {
        this.country = country;
    }

    public static User toUserEntity(RegisterUserRequestDto registerUserRequestDto) {
        User user = new User();
        BeanUtils.copyProperties(registerUserRequestDto, user);
        return user;
    }

    public static Address toAddressEntity(RegisterUserRequestDto registerUserRequestDto) {
        Address address = new Address();
        BeanUtils.copyProperties(registerUserRequestDto, address);
        return address;
    }
```

```
}


package com.jobportal.entity;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;

@Entity
public class Address {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    private String street;

    private String city;

    private String pincode;

    private String state;

    private String country;

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getStreet() {
        return street;
    }

    public void setStreet(String street) {
        this.street = street;
    }
    public String getCity() {
      return city;
    }

    public void setCity(String city) {
      this.city = city;
    }

    public String getPincode() {
      return pincode;
    }

    public void setPincode(String pincode) {
      this.pincode = pincode;
    }

    public String getState() {
      return state;
    }

    public void setState(String state) {
      this.state = state;
    }

    public String getCountry() {
      return country;
    }

    public void setCountry(String country) {
      this.country = country;
    }

}
```

Job Category Page:-



All Job Categories

Category Id	Category Name	Description	Action
1	Software Developer	Software Developer	Update Delete
2	DBA	Database Administrator	Update Delete
4	Frontend Developer	Frontend Developer	Update Delete
5	Fullstack Developer	Fullstack Developer	Update Delete

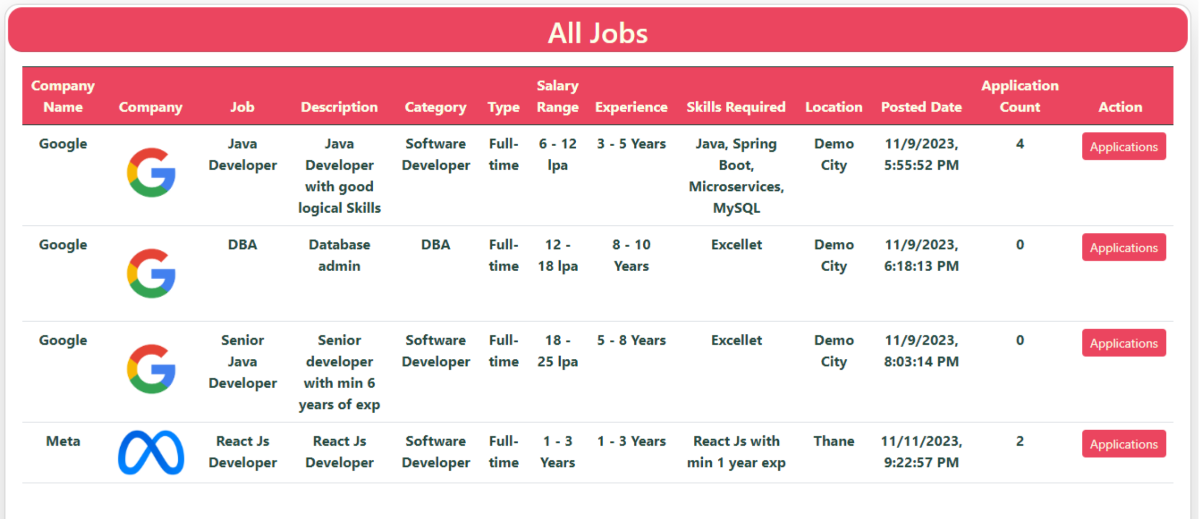
Code :

```
import { useState } from "react";
import { ToastContainer, toast } from "react-toastify";
import { useNavigate } from "react-router-dom";

const AddCategoryForm = () => {
  const [name, setName] = useState("");
  const [description, setDescription] = useState("");

  const admin_jwtToken = sessionStorage.getItem("admin-jwtToken");

  let navigate = useNavigate();

  const saveCategory = (e) => {
    let data = { name, description };

    fetch("http://localhost:8080/api/job/category/add", {
      method: "POST",
      headers: {
        Accept: "application/json",
        "Content-Type": "application/json",
        Authorization: "Bearer " + admin_jwtToken,
      },
      body: JSON.stringify(data),
    })
      .then((result) => {
        result.json().then((res) => {
          if (res.success) {
            toast.success(res.responseMessage, {
              position: "top-center",
              autoClose: 1000,
              hideProgressBar: false,
              closeOnClick: true,
              pauseOnHover: true,
              draggable: true,
              progress: undefined,
            });

            setTimeout(() => {
              navigate("/home");
            }, 2000); // Redirect after 3 seconds
          } else if (!res.success) {
            toast.error(res.responseMessage, {
              position: "top-center",
              autoClose: 1000,
              hideProgressBar: false,
              closeOnClick: true,
              pauseOnHover: true,
              draggable: true,
              progress: undefined,
            });
            setTimeout(() => {
              window.location.reload(true);
            }, 2000); // Redirect after 3 seconds
          } else {
            toast.error("It Seems Server is down!!!", {
              position: "top-center",
              autoClose: 1000,
              hideProgressBar: false,
              closeOnClick: true,
              pauseOnHover: true,
              draggable: true,
              progress: undefined,
            });
            setTimeout(() => {
              window.location.reload(true);
            }, 2000); // Redirect after 3 seconds
          }
        });
      })
      .catch((error) => {
        console.error(error);
        toast.error("It seems server is down", {
          position: "top-center",
          autoClose: 1000,
          hideProgressBar: false,
          closeOnClick: true,
          pauseOnHover: true,
          draggable: true,
          progress: undefined,
        });
        setTimeout(() => {
          window.location.reload(true);
        }, 1000); // Redirect after 3 seconds
      });
    e.preventDefault();
  };

  return (
    <div>
      <div class="mt-2 d-flex aligns-items-center justify-content-center">
        <div class="form-card border-color" style={{ width: "25rem" }}>
          <div className="container-fluid">
            <div
              className="card-header bg-color custom-bg-text mt-2 d-flex justify-content-center align-items-center"
              style={{
                borderRadius: "1em",
                height: "38px",
              }}
            >
              <h5 class="card-title">Add Job Category</h5>
            </div>
            <div class="card-body text-color mt-3">
              <form>
                <div class="mb-3">
                  <label for="title" class="form-label">
                    <b>Category Title</b>
                  </label>
                  <input
                    type="text"
                    class="form-control"
                    id="title"
                    placeholder="enter title.."
                    onChange={(e) => {
                      setName(e.target.value);
                    }}
                    value={name}
                  />
                </div>
                <div class="mb-3">
                  <label for="description" class="form-label">
                    <b>Category Description</b>
                  </label>
                  <textarea
                    class="form-control"
                    id="description"
                    rows="3"
                    placeholder="enter description.."
                    onChange={(e) => {
                      setDescription(e.target.value);
                    }}
                    value={description}
                  />
                </div>

                <div className="d-flex aligns-items-center justify-content-center mb-2">
                  <button
                    type="submit"
                    onClick={saveCategory}
                    class="btn bg-color custom-bg-text"
                  >
                    Add Job Category
                  </button>
                </div>

                <ToastContainer />
              </form>
            </div>
          </div>
        </div>
      </div>
    </div>
  );
};

export default AddCategoryForm;
```

Backend code:-

```
package com.jobportal.entity;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;

@Entity

public class JobCategory {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    private String name;

    private String description;

    private String status;
    public int getId() {
      return id;
    }

    public void setId(int id) {
      this.id = id;
    }

    public String getName() {
      return name;
    }

    public void setName(String name) {
      this.name = name;
    }

    public String getDescription() {
      return description;
    }

    public void setDescription(String description) {
      this.description = description;
    }

    public String getStatus() {
      return status;
    }

    public void setStatus(String status) {
      this.status = status;
    }

}
```

All Jobs Posted:-



All Jobs

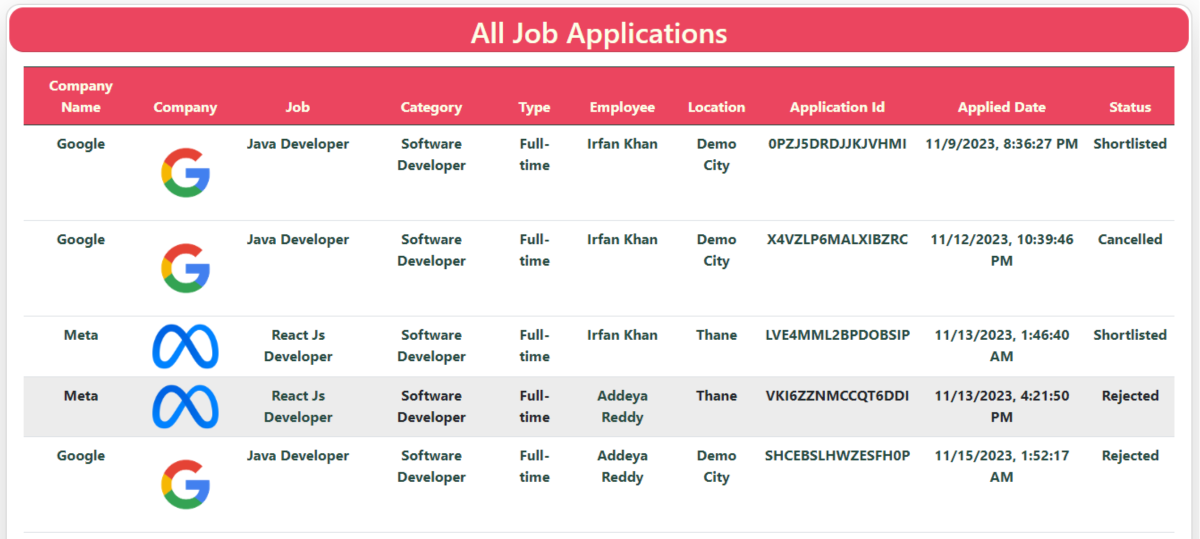
Company Name	Company	Job	Description	Category	Type	Salary Range	Experience	Skills Required	Location	Posted Date	Application Count	Action
Google		Java Developer	Java Developer with good logical Skills	Software Developer	Full-time	6 - 12 lpa	3 - 5 Years	Java, Spring Boot, Microservices, MySQL	Demo City	11/9/2023, 5:55:52 PM	4	Applications
Google		DBA	Database admin	DBA	Full-time	12 - 18 lpa	8 - 10 Years	Excellet	Demo City	11/9/2023, 6:18:13 PM	0	Applications
Google		Senior Java Developer	Senior developer with min 6 years of exp	Software Developer	Full-time	18 - 25 lpa	5 - 8 Years	Excellet	Demo City	11/9/2023, 8:03:14 PM	0	Applications
Meta		React Js Developer	React Js Developer	Software Developer	Full-time	1 - 3 Years	1 - 3 Years	React Js with min 1 year exp	Thane	11/11/2023, 9:22:57 PM	2	Applications

```
import { useState, useEffect } from "react";
import axios from "axios";
import React from "react";
import { Link, useNavigate } from "react-router-dom";
import { toast } from "react-toastify";

const ViewAllJobs = () => {
  const employer = JSON.parse(sessionStorage.getItem("active-employer"));
  const employer_jwtToken = sessionStorage.getItem("employer-jwtToken");

  const [allJobs, setAllJobs] = useState([
   {
     id: "",
     employer: {
       id: "",
       firstName: "",
       lastName: "",
       emailId: "",
       phoneNo: "",
       role: "",
       address: {
         id: "",
         street: "",
         city: "",
         pincode: "",
         state: "",
         country: "",
       },
       registrationDate: "",
       status: "",
     },
     title: "",
     description: "",
     category: {
       id: "",
       name: "",
       description: "",
       status: "",
     },
     companyName: "",
     companyLogo: "",
     address: {
       id: "",
       street: "",
       city: "",
       pincode: "",
       state: "",
       country: "",
     },
     jobType: "",
     salaryRange: "",
     experienceLevel: "",
     requiredSkills: "",
     status: "",
     datePosted: "",
     applicationCount: "",
   },
  ]);

  let navigate = useNavigate();

  useEffect(() => {
   const getAllJobs = async () => {
     const allJobs = await retrieveAllJobs();
     if (allJobs) {
       setAllJobs(allJobs.jobs);
     }
   };

   getAllJobs();
  }, []);

  const retrieveAllJobs = async () => {
   const response = await axios.get("http://localhost:8080/api/job/fetch/all");
   console.log(response.data);
   return response.data;
  };
```

```
const formatDateFromEpoch = (epochTime) => {
  const date = new Date(Number(epochTime));
  const formattedDate = date.toLocaleString(); // Adjust the format as needed

  return formattedDate;
};

const viewAppliedJobs = (jobId) => {
  navigate(`/job/${jobId}/application/all`);
};

return (
  <div className="mt-3">
    <div
      className="card form-card ms-2 me-2 mb-5 shadow-lg"
      style={{
        height: "45rem",
      }}
    >
      <div
        className="card-header custom-bg-text text-center bg-color"
        style={{
          borderRadius: "1em",
          height: "50px",
        }}
      >
        <h2>All Jobs</h2>
      </div>
      <div
        className="card-body"
        style={{
          overflowY: "auto",
        }}
      >
        <div className="table-responsive">
          <table className="table table-hover text-color text-center">
            <thead className="table-bordered border-color bg-color custom-bg-text">
              <tr>
                <th scope="col">Company Name</th>
                <th scope="col">Company</th>
                <th scope="col">Job</th>
                <th scope="col">Description</th>
                <th scope="col">Category</th>
                <th scope="col">Type</th>
                <th scope="col">Salary Range</th>
                <th scope="col">Experience</th>
                <th scope="col">Skills Required</th>
                <th scope="col">Location</th>
  <th scope="col">Posted Date</th>
  <th scope="col">Application Count</th>
  <th scope="col">Action</th>
 </tr>
</thead>
<tbody>
 {allJobs.map((job) => {
  return (
   <tr>
    <td>
     <b>{job.companyName}</b>
    </td>
    <td>
     <img
      src={
       "http://localhost:8080/api/job/" + job.companyLogo
      }
      class="img-fluid"
      alt="food_pic"
      style={{
       maxWidth: "90px",
      }}
     />
    </td>
    <td>
     <Link
      to={`/job/${job.id}/detail`}
      style={{ textDecoration: "none" }}
      className="text-color"
     >
      <b>{job.title}</b>
     </Link>
    </td>
    <td>
     <b>{job.description}</b>
    </td>
    <td>
     <b>{job.category.name}</b>
    </td>
    <td>
     <b>{job.jobType}</b>
    </td>
    <td>
     <b>{job.salaryRange}</b>
    </td>
    <td>
     <b>{job.experienceLevel}</b>
    </td>
```

```
                      <td>
                        <b>{job.requiredSkills}</b>
                      </td>
                      <td>
                        <b>{job.address.city}</b>
                      </td>
                      <td>
                        <b>{formatDateFromEpoch(job.datePosted)}</b>
                      </td>
                      <td>
                        <b>{job.applicationCount}</b>
                      </td>
                      <td>
                        <button
                          onClick={() => viewAppliedJobs(job.id)}
                          className="btn btn-sm bg-color custom-bg-text ms-2"
                        >
                          Applications
                        </button>
                      </td>
                    </tr>
                  );
                })}
              </tbody>
            </table>
          </div>
        </div>
      </div>
    </div>
  );
};

export default ViewAllJobs;
```

All Jobs Applications:-



All Job Applications

Company Name	Company	Job	Category	Type	Employee	Location	Application Id	Applied Date	Status
Google		Java Developer	Software Developer	Full-time	Irfan Khan	Demo City	0PZJ5DRDJJKJVHMI	11/9/2023, 8:36:27 PM	Shortlisted
Google		Java Developer	Software Developer	Full-time	Irfan Khan	Demo City	X4VZLP6MALXIBZRC	11/12/2023, 10:39:46 PM	Cancelled
Meta		React Js Developer	Software Developer	Full-time	Irfan Khan	Thane	LVE4MML2BPDOBSIP	11/13/2023, 1:46:40 AM	Shortlisted
Meta		React Js Developer	Software Developer	Full-time	Addeya Reddy	Thane	VKI6ZZNMCCQT6DDI	11/13/2023, 4:21:50 PM	Rejected
Google		Java Developer	Software Developer	Full-time	Addeya Reddy	Demo City	SHCEBSLHWZESFH0P	11/15/2023, 1:52:17 AM	Rejected

:

```
import { useState, useEffect } from "react";
import axios from "axios";
import React from "react";
import { Link, useNavigate } from "react-router-dom";
import { ToastContainer, toast } from "react-toastify";
import { Button, Modal } from "react-bootstrap";

const ViewAllJobApplication = () => {
  const admin = JSON.parse(sessionStorage.getItem("active-admin"));
  const admin_jwtToken = sessionStorage.getItem("admin-jwtToken");

  const [applications, setApplications] = useState([
    {
      id: 1,
      applicationId: "",
      job: {
        id: 1,
        employer: {
          id: 4,
          firstName: "",
          lastName: "",
          emailId: "",
          phoneNo: "",
        },
        title: "",
        description: "",
        category: {
          id: 1,
          name: "",
          description: "",
          status: "",
        },
        address: {
          id: "",
          street: "",
          city: "",
          pincode: "",
          state: "",
          country: "",
        },
        companyName: "",
        companyLogo: "",
        jobType: "",
        salaryRange: "",
        experienceLevel: "",
        requiredSkills: "",
        status: "",
        datePosted: "",
        applicationCount: 0,
      },
      employee: {
        userProfile: {
          id: 1,
          bio: "",
          website: "",
          resume: "",
          linkedlnProfileLink: "",
          githubProfileLink: "",
          skills: [
            {
              id: 7,
              skill: "",
              experience: 0,
              userId: 0,
            },
          ],
          educations: [
            {
              id: 5,
              degree: "",
              institution: "",
              startDate: "",
              endDate: "",
              userId: 0,
            },
          ],
          workExperiences: [
            {
              id: 2,
              experience: 0,
              jobPosition: "",
              company: "",
              startDate: "",
              endDate: "",
              userId: 0,
            },
          ],
          profilePic: "",
        },
      },
      dateTime: "",
      status: "",
      jobId: 0,
      employeeId: 0,
    },
  ]);

  let navigate = useNavigate();

  useEffect(() => {
    const getAllJobs = async () => {
      const jobApplicationResponse = await retrieveAllJobApplication();
      if (jobApplicationResponse) {
        setApplications(jobApplicationResponse.applications);
      }
    };
    getAllJobs();
  }, []);

  const retrieveAllJobApplication = async () => {
    const response = await axios.get(
      "http://localhost:8080/api/job/application/fetch/all",
      {
        headers: {
          Authorization: "Bearer " + admin_jwtToken, // Replace with your actual JWT token
        },
      }
    );
    console.log(response.data);
    return response.data;
  };

  const formatDateFromEpoch = (epochTime) => {
    const date = new Date(Number(epochTime));
    const formattedDate = date.toLocaleString(); // Adjust the format as needed

    return formattedDate;
  };

  const viewEmployeeProfile = (employee) => {
    navigate("/employee/profile/detail", { state: employee });
  };

  return (
    <div className="mt-3">
      <div
        className="card form-card ms-2 me-2 mb-5 shadow-lg"
        style={{
          height: "45rem",
        }}
      >
        <div
          className="card-header custom-bg-text text-center bg-color"
          style={{
            borderRadius: "1em",
            height: "50px",
          }}
        >
          <h2>All Job Applications</h2>
        </div>
        <div
          className="card-body"
          style={{
            overflowY: "auto",
          }}
        >
          <div className="table-responsive">
            <table className="table table-hover text-color text-center">
              <thead className="table-bordered border-color bg-color custom-bg-text">
                <tr>
                  <th scope="col">Company Name</th>
                  <th scope="col">Company</th>
                  <th scope="col">Job</th>
                  <th scope="col">Category</th>
                  <th scope="col">Type</th>
                  <th scope="col">Employee</th>
                  <th scope="col">Location</th>
                  <th scope="col">Application Id</th>
                  <th scope="col">Applied Date</th>
                  <th scope="col">Status</th>
                </tr>
              </thead>
              <tbody>
                {applications.map((application) => {
                  return (
                    <tr>
                      <td>
                        <b>{application.job.companyName}</b>
                      </td>
                      <td>
                        <img
                          src={
                            "http://localhost:8080/api/job/" +
                            application.job.companyLogo
                          }
                          class="img-fluid"
                          alt="company_pic"
                          style={{
                            maxWidth: "90px",
                          }}
                        />
                      </td>
                      <td>
                        <Link
                          to={`/job/${application.job.id}/detail`}
                          style={{ textDecoration: "none" }}
                          className="text-color"
                        >
                          <b>{application.job.title}</b>
                        </Link>
                      </td>

                      <td>
                        <b>{application.job.category.name}</b>
                      </td>
                      <td>
                        <b>{application.job.jobType}</b>
                      </td>

                      <td>
                        <b
                          className="text-color"
                          onClick={() =>
                            viewEmployeeProfile(application.employee)
                          }
                        >
                          {application.employee.firstName +
                            " " +
                            application.employee.lastName}
                        </b>
                      </td>

                      <td>
                        <b>{application.job.address.city}</b>
                      </td>
                      <td>
                        <b>{application.applicationId}</b>
                      </td>
                      <td>
                        <b>{formatDateFromEpoch(application.dateTime)}</b>
                      </td>
                      <td>
```

```
                        <b>{application.status}</b>
                      </td>
                    </tr>
                  );
                })}
              </tbody>
            </table>
          </div>
        </div>
      </div>
    </div>
  );
};

export default ViewAllJobApplication;
```

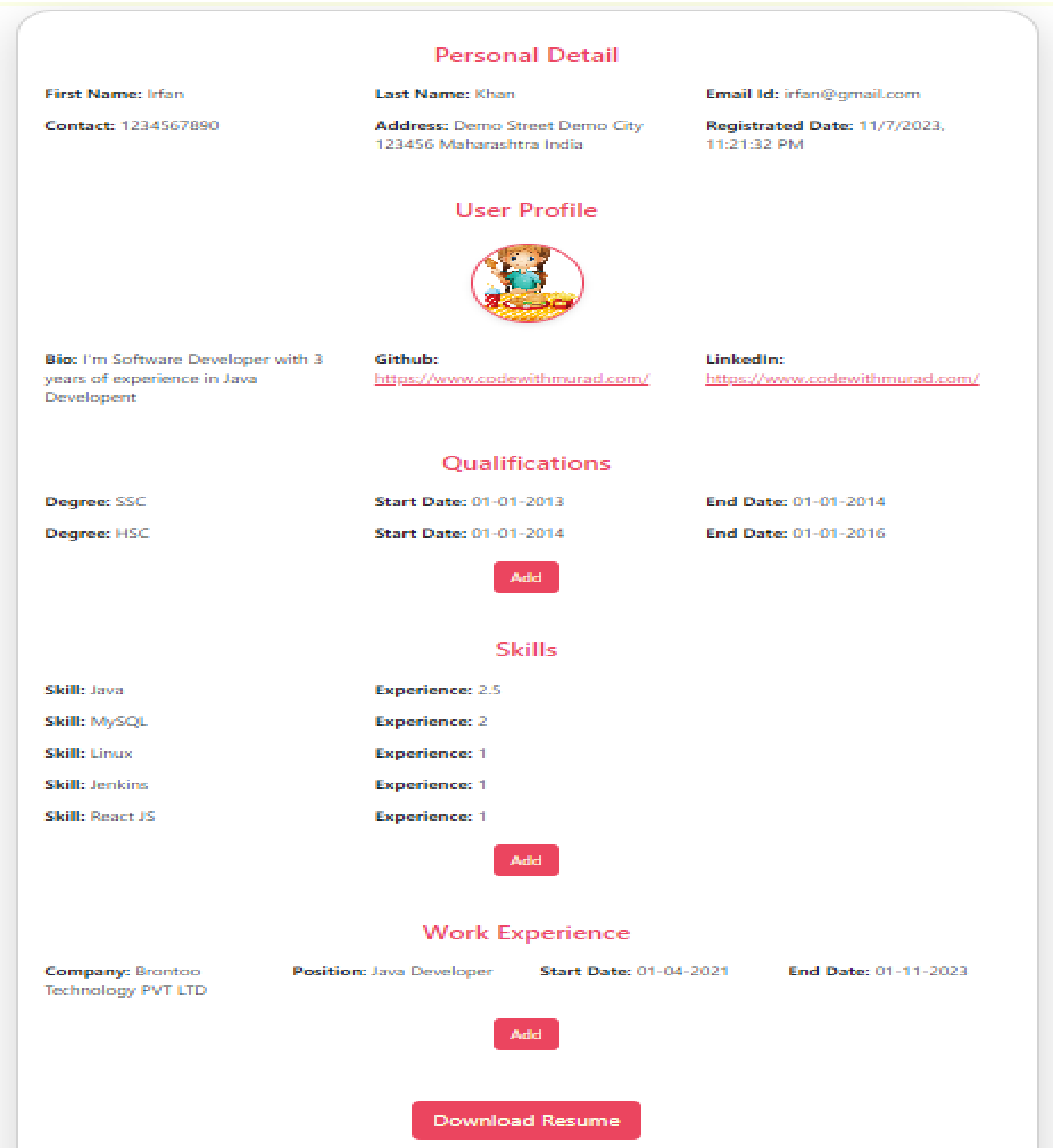
Backend code:-

```
package com.jobportal.entity;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.JoinColumn;
import jakarta.persistence.ManyToOne;
import jakarta.persistence.Transient;

@Entity
public class JobApplication {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    private String applicationId;

    @ManyToOne
    @JoinColumn(name = "job_id")
    private Job job;

    @ManyToOne
    @JoinColumn(name = "employee_id")
    private User employee;
    private String dateTime;

    private String status; // Applied, Shortlisted, Rejected

    @Transient
    private int jobId;

    @Transient
    private int employeeId;

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getApplicationId() {
        return applicationId;
    }

    public void setApplicationId(String applicationId) {
        this.applicationId = applicationId;
    }

    public Job getJob() {
        return job;
    }

    public void setJob(Job job) {
        this.job = job;
    }

    public User getEmployee() {
        return employee;
    }

    public void setEmployee(User employee) {
        this.employee = employee;
    }
```

```
   public String getDateTime() {
     return dateTime;
   }

   public void setDateTime(String dateTime) {
     this.dateTime = dateTime;
   }

   public String getStatus() {
     return status;
   }

   public void setStatus(String status) {
     this.status = status;
   }

   public int getJobId() {
     return jobId;
   }

   public void setJobId(int jobId) {
     this.jobId = jobId;
   }

   public int getEmployeeId() {
     return employeeId;
   }

   public void setEmployeeId(int employeeId) {
     this.employeeId = employeeId;
   }

}
```

Employee Profile

Update Employee Details:-



Personal Detail

First Name: Irfan

Last Name: Khan

Email Id: irfan@gmail.com

Contact: 1234567890

Address: Demo Street Demo City 123456 Maharashtra India

Registrated Date: 11/7/2023, 11:21:32 PM

User Profile

Bio: I'm Software Developer with 3 years of experience in Java Developent

Github: https://www.codewithmurad.com/

LinkedIn: https://www.codewithmurad.com/

Qualifications

Degree: SSC	**Start Date:** 01-01-2013	**End Date:** 01-01-2014
Degree: HSC	**Start Date:** 01-01-2014	**End Date:** 01-01-2016

Skills

Skill: Java	**Experience:** 2.5
Skill: MySQL	**Experience:** 2
Skill: Linux	**Experience:** 1
Skill: Jenkins	**Experience:** 1
Skill: React JS	**Experience:** 1

Work Experience

Company: Brontoo Technology PVT LTD	**Position:** Java Developer	**Start Date:** 01-04-2021	**End Date:** 01-11-2023

Code:

```
import { useState, useEffect } from "react";
import axios from "axios";
import { toast } from "react-toastify";
import { useNavigate } from "react-router-dom";

const UpdateUserProfileForm = () => {
  const employee = JSON.parse(sessionStorage.getItem("active-employee"));
  const employee_jwtToken = sessionStorage.getItem("employee-jwtToken");

  let navigate = useNavigate();

  const [selectedImage1, setSelectImage1] = useState(null);
  const [selectedResume, setSelectResume] = useState(null);

  const [profile, setProfile] = useState({
    userId: employee.id,
    bio: "",
    website: "",
    linkedlnProfileLink: "",
    githubProfileLink: "",
    profilePic: "",
    resume: "",
  });

  const handleInput = (e) => {
    setProfile({ ...profile, [e.target.name]: e.target.value });
  };

  const saveUserProfile = (e) => {
    e.preventDefault();
    if (profile === null) {
      toast.error("invalid input!!!", {
        position: "top-center",
        autoClose: 3000,
        hideProgressBar: false,
        closeOnClick: true,
        pauseOnHover: true,
        draggable: true,
        progress: undefined,
      });

      return;
    }
```

```
const formData = new FormData();
formData.append("userId", profile.userId);
formData.append("bio", profile.bio);
formData.append("website", profile.website);
formData.append("githubProfileLink", profile.githubProfileLink);
formData.append("linkedlnProfileLink", profile.linkedlnProfileLink);
formData.append("resume", selectedResume);
formData.append("profilePic", selectedImage1);

axios
  .put("http://localhost:8080/api/user/profile/add", formData, {
    headers: {
      Authorization: "Bearer " + employee_jwtToken, // Replace with your actual JWT token
    },
  })
  .then((resp) => {
    let response = resp.data;

    if (response.success) {
      toast.success(response.responseMessage, {
        position: "top-center",
        autoClose: 1000,
        hideProgressBar: false,
        closeOnClick: true,
        pauseOnHover: true,
        draggable: true,
        progress: undefined,
      });

      setTimeout(() => {
        navigate("/employee/profile/detail");
      }, 2000); // Redirect after 3 seconds
    } else if (!response.success) {
      toast.error(response.responseMessage, {
        position: "top-center",
        autoClose: 1000,
        hideProgressBar: false,
        closeOnClick: true,
        pauseOnHover: true,
        draggable: true,
        progress: undefined,
      });
      setTimeout(() => {
        window.location.reload(true);
      }, 2000); // Redirect after 3 seconds
    } else {
      toast.error("It Seems Server is down!!!", {
        position: "top-center",
            autoClose: 1000,
            hideProgressBar: false,
            closeOnClick: true,
            pauseOnHover: true,
            draggable: true,
            progress: undefined,
          });
          setTimeout(() => {
            window.location.reload(true);
          }, 2000); // Redirect after 3 seconds
        }
      })
      .catch((error) => {
        console.error(error);
        toast.error("It seems server is down", {
          position: "top-center",
          autoClose: 1000,
          hideProgressBar: false,
          closeOnClick: true,
          pauseOnHover: true,
          draggable: true,
          progress: undefined,
        });
        setTimeout(() => {
          window.location.reload(true);
        }, 2000); // Redirect after 3 seconds
      });
  };

  return (
    <div>
      <div class="mt-2 d-flex aligns-items-center justify-content-center mb-4">
        <div class="card form-card shadow-lg" style={{ width: "60rem" }}>
          <div className="container-fluid">
            <div
              className="card-header bg-color custom-bg-text mt-2 text-center"
              style={{
                borderRadius: "1em",
                height: "45px",
              }}
            >
              <h5 class="card-title">Update Profile</h5>
            </div>
            <div class="card-body text-color">
              <form className="row g-3">
                <div className="col-md-6 mb-3">
                  <label htmlFor="title" className="form-label">
                    <b>Bio</b>
                  </label>
                  <input
                    type="text"
                    className="form-control"
                    id="bio"
                    name="bio"
                    onChange={handleInput}
                    value={profile.bio}
                  />
                </div>

                <div className="col-md-6 mb-3">
                  <label className="form-label">
                    <b>Website</b>
                  </label>
                  <input
                    type="text"
                    className="form-control"
                    name="website"
                    onChange={handleInput}
                    value={profile.website}
                  />
                </div>

                <div className="col-md-6 mb-3">
                  <label className="form-label">
                    <b>Github Link</b>
                  </label>
                  <input
                    type="text"
                    className="form-control"
                    name="githubProfileLink"
                    onChange={handleInput}
                    value={profile.githubProfileLink}
                  />
                </div>

                <div className="col-md-6 mb-3">
                  <label className="form-label">
                    <b>LinkedIn Profile Link</b>
                  </label>
                  <input
                    type="text"
                    className="form-control"
                    name="linkedlnProfileLink"
                    onChange={handleInput}
                    value={profile.linkedlnProfileLink}
                  />
```

```
                </div>

                <div className="col-md-6 mb-3">
                  <label for="formFile" class="form-label">
                    <b> Select Profile Pic</b>
                  </label>
                  <input
                    class="form-control"
                    type="file"
                    id="formFile"
                    name="profilePic"
                    onChange={(e) => setSelectImage1(e.target.files[0])}
                    required
                  />
                </div>

                <div className="col-md-6 mb-3">
                  <label for="formFile" class="form-label">
                    <b> Select Resume</b>
                  </label>
                  <input
                    class="form-control"
                    type="file"
                    id="formFile"
                    name="resume"
                    onChange={(e) => setSelectResume(e.target.files[0])}
                    required
                  />
                </div>

                <div className="d-flex aligns-items-center justify-content-center mb-2">
                  <button
                    type="submit"
                    class="btn bg-color custom-bg-text"
                    onClick={saveUserProfile}
                  >
                    Update Profile
                  </button>
                </div>
              </form>
            </div>
          </div>
        </div>
      </div>
    </div>
  );
};
```

```
package com.jobportal.entity;

import com.fasterxml.jackson.annotation.JsonIgnore;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.JoinColumn;
import jakarta.persistence.ManyToOne;
import jakarta.persistence.Transient;

@Entity
public class UserEducation {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    private String degree;

    private String institution;

    private String startDate;

    private String endDate;

    @Transient
    private int userId;

    @JsonIgnore
    @ManyToOne
    @JoinColumn(name = "userProfile_id")
    private UserProfile userProfile;

    public int getId() {
      return id;
    }

    public void setId(int id) {
      this.id = id;
    }

    public String getDegree() {
      return degree;
    }
    public void setDegree(String degree) {
        this.degree = degree;
    }

    public String getInstitution() {
        return institution;
    }

    public void setInstitution(String institution) {
        this.institution = institution;
    }

    public String getStartDate() {
        return startDate;
    }

    public void setStartDate(String startDate) {
        this.startDate = startDate;
    }

    public String getEndDate() {
        return endDate;
    }

    public void setEndDate(String endDate) {
        this.endDate = endDate;
    }

    public int getUserId() {
        return userId;
    }

    public void setUserId(int userId) {
        this.userId = userId;
    }

    public UserProfile getUserProfile() {
        return userProfile;
    }

    public void setUserProfile(UserProfile userProfile) {
        this.userProfile = userProfile;
    }

}
```

```
package com.jobportal.entity;

import java.util.List;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.OneToMany;

@Entity
public class UserProfile {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    private String bio;

    private String website; // optional

    private String resume; // resume path

    private String linkedlnProfileLink;

    private String githubProfileLink; // optional

    // Establish a one-to-many relationship between UserProfile and UserSkills
    @OneToMany(mappedBy = "userProfile")
    private List<UserSkill> skills;

    @OneToMany(mappedBy = "userProfile")
    private List<UserEducation> educations;

    @OneToMany(mappedBy = "userProfile")
    private List<UserWorkExperience> workExperiences;

    private String profilePic;

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getBio() {
        return bio;
    }

    public void setBio(String bio) {
        this.bio = bio;
    }

    public String getWebsite() {
        return website;
    }

    public void setWebsite(String website) {
        this.website = website;
    }

    public String getResume() {
        return resume;
    }

    public void setResume(String resume) {
        this.resume = resume;
    }

    public String getLinkedlnProfileLink() {
        return linkedlnProfileLink;
    }

    public void setLinkedlnProfileLink(String linkedlnProfileLink) {
        this.linkedlnProfileLink = linkedlnProfileLink;
    }

    public String getGithubProfileLink() {
        return githubProfileLink;
    }

    public void setGithubProfileLink(String githubProfileLink) {
        this.githubProfileLink = githubProfileLink;
    }

    public List<UserSkill> getSkills() {
        return skills;
    }

    public void setSkills(List<UserSkill> skills) {
        this.skills = skills;
    }

    public List<UserEducation> getEducations() {
```

```
    return educations;
  }

  public void setEducations(List<UserEducation> educations) {
    this.educations = educations;
  }

  public List<UserWorkExperience> getWorkExperiences() {
    return workExperiences;
  }

  public void setWorkExperiences(List<UserWorkExperience> workExperiences) {
    this.workExperiences = workExperiences;
  }

  public String getProfilePic() {
    return profilePic;
  }

  public void setProfilePic(String profilePic) {
    this.profilePic = profilePic;
  }

}

package com.jobportal.entity;

import com.fasterxml.jackson.annotation.JsonIgnore;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.JoinColumn;
import jakarta.persistence.ManyToOne;
import jakarta.persistence.Transient;

@Entity
public class UserSkill {

  @Id
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  private int id;

  private String skill;

  private double experience;
```

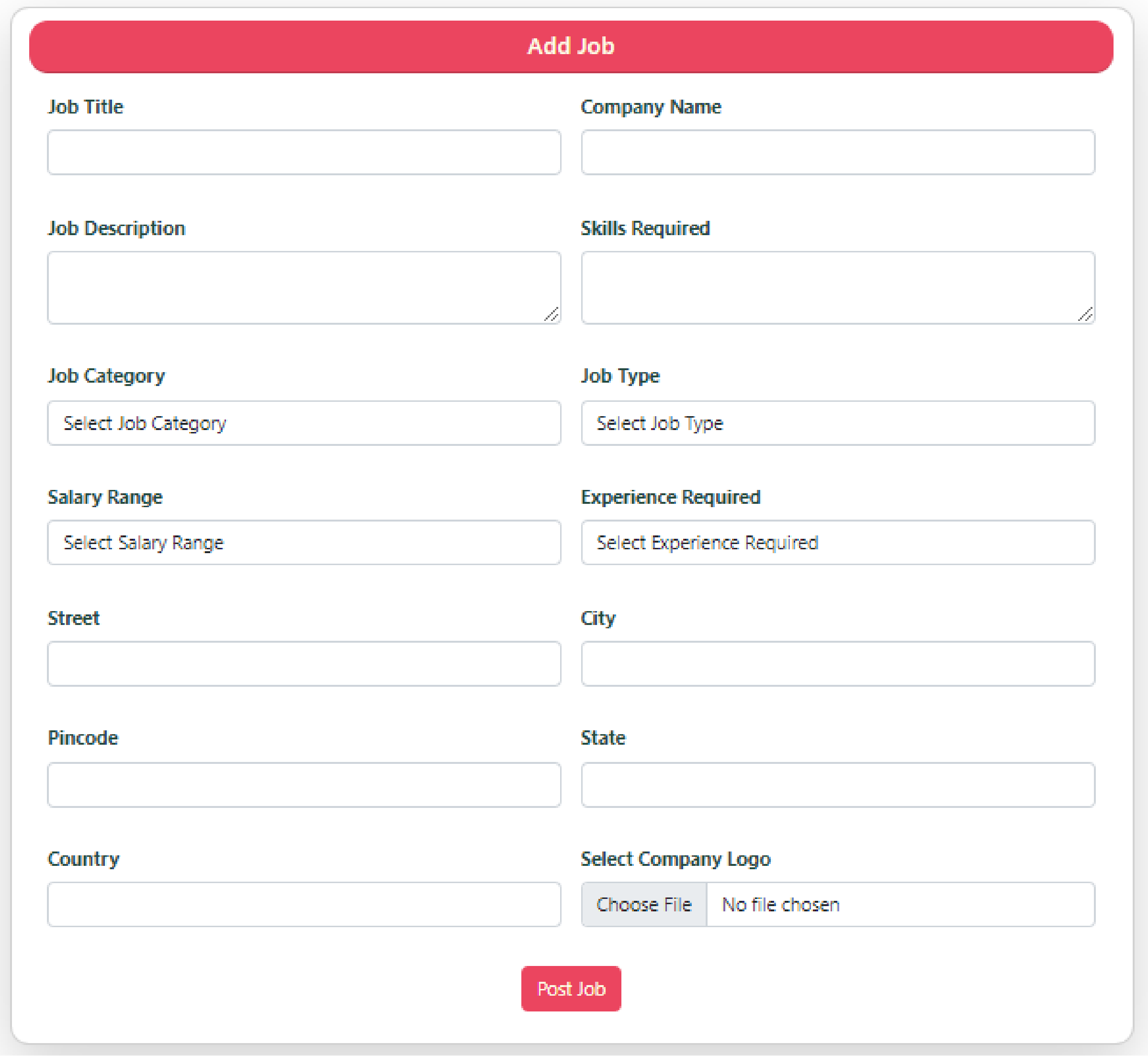
```
    @Transient
    private int userId;

    @JsonIgnore
    @ManyToOne
    @JoinColumn(name = "userProfile_id")
    private UserProfile userProfile;

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getSkill() {
        return skill;
    }

    public void setSkill(String skill) {
        this.skill = skill;
    }

    public double getExperience() {
        return experience;
    }

    public void setExperience(double experience) {
        this.experience = experience;
    }

    public int getUserId() {
        return userId;
    }

    public void setUserId(int userId) {
        this.userId = userId;
    }

    public UserProfile getUserProfile() {
        return userProfile;
    }

    public void setUserProfile(UserProfile userProfile) {
        this.userProfile = userProfile;
    }
```

```
}

package com.jobportal.entity;

import com.fasterxml.jackson.annotation.JsonIgnore;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.JoinColumn;
import jakarta.persistence.ManyToOne;
import jakarta.persistence.Transient;

@Entity
public class UserWorkExperience {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    private double experience;

    private String jobPosition;

    private String company;

    private String startDate;

    private String endDate;

    @Transient
    private int userId;

    @JsonIgnore
    @ManyToOne
    @JoinColumn(name = "userProfile_id")
    private UserProfile userProfile;

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public double getExperience() {
        return experience;
    }

    public void setExperience(double experience) {
        this.experience = experience;
    }

    public String getJobPosition() {
        return jobPosition;
    }

    public void setJobPosition(String jobPosition) {
        this.jobPosition = jobPosition;
    }

    public String getCompany() {
        return company;
    }

    public void setCompany(String company) {
        this.company = company;
    }

    public String getStartDate() {
        return startDate;
    }

    public void setStartDate(String startDate) {
        this.startDate = startDate;
    }

    public String getEndDate() {
        return endDate;
    }

    public void setEndDate(String endDate) {
        this.endDate = endDate;
    }

    public int getUserId() {
        return userId;
    }

    public void setUserId(int userId) {
        this.userId = userId;
    }

    public UserProfile getUserProfile() {
        return userProfile;
    }

    public void setUserProfile(UserProfile userProfile) {
        this.userProfile = userProfile;
    }

}



export default UpdateUserProfileForm;


package com.jobportal.dto;

public class UpdateProfileDetailRequest {

    private int userId;

    private String bio;

    private String website;

    private String linkedlnProfileLink;

    private String githubProfileLink;

    public int getUserId() {
        return userId;
    }

    public void setUserId(int userId) {
        this.userId = userId;
    }

    public String getBio() {
        return bio;
    }

    public void setBio(String bio) {
        this.bio = bio;
    }

    public String getWebsite() {
        return website;
    }
    public void setWebsite(String website) {
        this.website = website;
    }

    public String getLinkedlnProfileLink() {
        return linkedlnProfileLink;
    }

    public void setLinkedlnProfileLink(String linkedlnProfileLink) {
        this.linkedlnProfileLink = linkedlnProfileLink;
    }

    public String getGithubProfileLink() {
        return githubProfileLink;
    }

    public void setGithubProfileLink(String githubProfileLink) {
        this.githubProfileLink = githubProfileLink;
    }

}
```

```
package com.jobportal.dto;

import org.springframework.web.multipart.MultipartFile;

public class UpdateUserProfilePicRequest {

    private int userId;

    private MultipartFile profilePic;

    public int getUserId() {
        return userId;
    }

    public void setUserId(int userId) {
        this.userId = userId;
    }

    public MultipartFile getProfilePic() {
        return profilePic;
    }

    public void setProfilePic(MultipartFile profilePic) {
        this.profilePic = profilePic;
    }
```

```
}

package com.jobportal.dto;

import org.springframework.web.multipart.MultipartFile;

public class UpdateUserProfileRequest {

    private Integer userId;

    private String bio;

    private String website;

    private MultipartFile resume;

    private String linkedlnProfileLink;

    private String githubProfileLink;

    private MultipartFile profilePic;

    public Integer getUserId() {
      return userId;
    }

    public void setUserId(Integer userId) {
      this.userId = userId;
    }

    public String getBio() {
      return bio;
    }

    public void setBio(String bio) {
      this.bio = bio;
    }

    public String getWebsite() {
      return website;
    }

    public void setWebsite(String website) {
      this.website = website;
    }

    public MultipartFile getResume() {
        return resume;
    }

    public void setResume(MultipartFile resume) {
        this.resume = resume;
    }

    public String getLinkedlnProfileLink() {
        return linkedlnProfileLink;
    }

    public void setLinkedlnProfileLink(String linkedlnProfileLink) {
        this.linkedlnProfileLink = linkedlnProfileLink;
    }

    public String getGithubProfileLink() {
        return githubProfileLink;
    }

    public void setGithubProfileLink(String githubProfileLink) {
        this.githubProfileLink = githubProfileLink;
    }

    public MultipartFile getProfilePic() {
        return profilePic;
    }

    public void setProfilePic(MultipartFile profilePic) {
        this.profilePic = profilePic;
    }

}


package com.jobportal.dto;

import org.springframework.web.multipart.MultipartFile;

public class UpdateUserResumeRequest {

    private int userId;

    private MultipartFile resume;

    public int getUserId() {
        return userId;
    }
    public void setUserId(int userId) {
        this.userId = userId;
    }

    public MultipartFile getResume() {
        return resume;
    }

    public void setResume(MultipartFile resume) {
        this.resume = resume;
    }

}
```

Add Job form:-



Add Job

Job Title

Company Name

Job Description

Skills Required

Job Category

Select Job Category

Job Type

Select Job Type

Salary Range

Select Salary Range

Experience Required

Select Experience Required

Street

City

Pincode

State

Country

Select Company Logo

Choose File No file chosen

Post Job

Code :

```
import { useState, useEffect } from "react";
import axios from "axios";
import { toast } from "react-toastify";
import { useNavigate } from "react-router-dom";
```

```
const AddJobForm = () => {
  const [categories, setCategories] = useState([]);

  const [jobTypes, setJobTypes] = useState([]);
  const [salaryRange, setSalaryRange] = useState([]);
  const [experience, setExperience] = useState([]);

  const employer = JSON.parse(sessionStorage.getItem("active-employer"));
  const employer_jwtToken = sessionStorage.getItem("employer-jwtToken");

  let navigate = useNavigate();

  const retrieveAllCategories = async () => {
    const response = await axios.get(
      "http://localhost:8080/api/job/category/fetch/all"
    );
    return response.data;
  };
  const retrieveAllJobTypes = async () => {
    const response = await axios.get(
      "http://localhost:8080/api/helper/job/type/fetch/all"
    );
    return response.data;
  };
  const retrieveAllSalary = async () => {
    const response = await axios.get(
      "http://localhost:8080/api/helper/job/salary/range/fetch/all"
    );
    return response.data;
  };

  const retrieveAllExperience = async () => {
    const response = await axios.get(
      "http://localhost:8080/api/helper/job/expereince/fetch/all"
    );
    return response.data;
  };

  useEffect(() => {
    const getAllCategories = async () => {
      const resCategory = await retrieveAllCategories();
      if (resCategory) {
        setCategories(resCategory.categories);
      }
    };

    const getAllJobTypes = async () => {
      const res = await retrieveAllJobTypes();
      if (res) {
        setJobTypes(res);
      }
    };

    const getAllExperience = async () => {
      const res = await retrieveAllExperience();
      if (res) {
        setExperience(res);
      }
    };

    const getAllSalaryRange = async () => {
      const res = await retrieveAllSalary();
      if (res) {
        setSalaryRange(res);
      }
    };

    getAllExperience();
    getAllJobTypes();
    getAllSalaryRange();
    getAllCategories();
  }, []);

  const [selectedImage1, setSelectImage1] = useState(null);

  const [job, setJob] = useState({
    employerId: employer.id,
    jobCategoryId: "",
    title: "",
    description: "",
    companyName: "",
    companyLogo: "",
    jobType: "",
    salaryRange: "",
    experienceLevel: "",
    requiredSkills: "",
    street: "",
    city: "",
    pincode: "",
    state: "",
    country: "",
  });

  const handleInput = (e) => {
    setJob({ ...job, [e.target.name]: e.target.value });
  };

  const saveJob = (e) => {
    e.preventDefault();
    if (job === null) {
      toast.error("invalid input!!!", {
        position: "top-center",
        autoClose: 3000,
        hideProgressBar: false,
        closeOnClick: true,
        pauseOnHover: true,
        draggable: true,
        progress: undefined,
      });

      return;
    }

    const formData = new FormData();
    formData.append("employerId", job.employerId);
    formData.append("jobCategoryId", job.jobCategoryId);
    formData.append("title", job.title);
    formData.append("description", job.description);
    formData.append("companyName", job.companyName);
    formData.append("companyLogo", selectedImage1);
    formData.append("jobType", job.jobType);
    formData.append("salaryRange", job.salaryRange);
    formData.append("experienceLevel", job.experienceLevel);
    formData.append("requiredSkills", job.requiredSkills);
    formData.append("street", job.street);
```

```
formData.append("city", job.city);
formData.append("pincode", job.pincode);
formData.append("state", job.state);
formData.append("country", job.country);
axios
  .post("http://localhost:8080/api/job/add", formData, {
    headers: {
      Authorization: "Bearer " + employer_jwtToken, // Replace with your actual JWT token
    },
  })
  .then((resp) => {
    let response = resp.data;

    if (response.success) {
      toast.success(response.responseMessage, {
        position: "top-center",
        autoClose: 1000,
        hideProgressBar: false,
        closeOnClick: true,
        pauseOnHover: true,
        draggable: true,
        progress: undefined,
      });

      setTimeout(() => {
        navigate("/home");
      }, 2000); // Redirect after 3 seconds
    } else if (!response.success) {
      toast.error(response.responseMessage, {
        position: "top-center",
        autoClose: 1000,
        hideProgressBar: false,
        closeOnClick: true,
        pauseOnHover: true,
        draggable: true,
        progress: undefined,
      });
      setTimeout(() => {
        window.location.reload(true);
      }, 2000); // Redirect after 3 seconds
    } else {
      toast.error("It Seems Server is down!!!", {
            position: "top-center",
            autoClose: 1000,
            hideProgressBar: false,
            closeOnClick: true,
            pauseOnHover: true,
            draggable: true,
            progress: undefined,
          });
          setTimeout(() => {
            window.location.reload(true);
          }, 2000); // Redirect after 3 seconds
        }
      })
      .catch((error) => {
        console.error(error);
        toast.error("It seems server is down", {
          position: "top-center",
          autoClose: 1000,
          hideProgressBar: false,
          closeOnClick: true,
          pauseOnHover: true,
          draggable: true,
          progress: undefined,
        });
        setTimeout(() => {
          window.location.reload(true);
        }, 2000); // Redirect after 3 seconds
      });
  };

  return (
    <div>
      <div class="mt-2 d-flex aligns-items-center justify-content-center mb-4">
        <div class="card form-card shadow-lg" style={{ width: "60rem" }}>
          <div className="container-fluid">
            <div
              className="card-header bg-color custom-bg-text mt-2 text-center"
              style={{
                borderRadius: "1em",
                height: "45px",
              }}
            >
              <h5 class="card-title">Add Job</h5>
            </div>
            <div class="card-body text-color">
              <form className="row g-3">
                <div className="col-md-6 mb-3">
                  <label htmlFor="title" className="form-label">
                    <b>Job Title</b>
                  </label>
                  <input
                    type="text"
                    className="form-control"
                    id="title"
                    name="title"
                    onChange={handleInput}
                    value={job.title}
                  />
                </div>

                <div className="col-md-6 mb-3">
                  <label className="form-label">
                    <b>Company Name</b>
                  </label>
                  <input
                    type="text"
                    className="form-control"
                    name="companyName"
                    onChange={handleInput}
                    value={job.companyName}
                  />
                </div>

                <div className="col-md-6 mb-3">
                  <label htmlFor="description" className="form-label">
                    <b>Job Description</b>
                  </label>
                  <textarea
                    className="form-control"
                    id="description"
                    name="description"
                    rows="2"
                    onChange={handleInput}
                    value={job.description}
                  />
                </div>

                <div className="col-md-6 mb-3">
                  <label htmlFor="description" className="form-label">
                    <b>Skills Required</b>
                  </label>
                  <textarea
                    className="form-control"
                    id="description"
                    name="requiredSkills"
                    rows="2"
                    onChange={handleInput}
                    value={job.requiredSkills}
                  />
                </div>

                <div className="col-md-6 mb-3">
                  <label className="form-label">
                    <b>Job Category</b>
                  </label>

                  <select
                    name="jobCategoryId"
                    onChange={handleInput}
                    className="form-control"
                  >
                    <option value="">Select Job Category</option>

                    {categories.map((category) => {
                      return (
                        <option value={category.id}> {category.name} </option>
                      );
                    })}
                  </select>
                </div>

                <div className="col-md-6 mb-3">
                  <label className="form-label">
                    <b>Job Type</b>
                  </label>
```

```
                  <select
                    name="jobType"
                    onChange={handleInput}
                    className="form-control"
                  >
                    <option value="">Select Job Type</option>

                    {jobTypes.map((type) => {
                      return <option value={type}> {type} </option>;
                    })}
                  </select>
                </div>

                <div className="col-md-6 mb-3">
                  <label className="form-label">
                    <b>Salary Range</b>
                  </label>

                  <select
                    name="salaryRange"
                    onChange={handleInput}
                    className="form-control"
                  >
                    <option value="">Select Salary Range</option>

                    {salaryRange.map((range) => {
                      return <option value={range}> {range} </option>;
                    })}
                  </select>
                </div>

                <div className="col-md-6 mb-3">
                  <label className="form-label">
                    <b>Experience Required</b>
                  </label>

                  <select
                    name="experienceLevel"
                    onChange={handleInput}
                    className="form-control"
                  >
                    <option value="">Select Experience Required</option>
```

```
    {experience.map((exp) => {
      return <option value={exp}> {exp} </option>;
    })}
  </select>
</div>

{/* Address Fields */}
<div className="col-md-6 mb-3">
  <label htmlFor="street" className="form-label">
    <b>Street</b>
  </label>
  <input
    type="text"
    className="form-control"
    id="street"
    name="street"
    onChange={handleInput}
    value={job.street}
  />
</div>

<div className="col-md-6 mb-3">
  <label htmlFor="city" className="form-label">
    <b>City</b>
  </label>
  <input
    type="text"
    className="form-control"
    id="city"
    name="city"
    onChange={handleInput}
    value={job.city}
  />
</div>

<div className="col-md-6 mb-3">
  <label htmlFor="pincode" className="form-label">
    <b>Pincode</b>
  </label>
  <input
    type="text"
    className="form-control"
    id="pincode"
    name="pincode"
    onChange={handleInput}
    value={job.pincode}
  />
</div>

<div className="col-md-6 mb-3">
  <label htmlFor="state" className="form-label">
    <b>State</b>
  </label>
  <input
    type="text"
    className="form-control"
    id="state"
    name="state"
    onChange={handleInput}
    value={job.state}
  />
</div>

<div className="col-md-6 mb-3">
  <label htmlFor="country" className="form-label">
    <b>Country</b>
  </label>
  <input
    type="text"
    className="form-control"
    id="country"
    name="country"
    onChange={handleInput}
    value={job.country}
  />
</div>

<div className="col-md-6 mb-3">
  <label for="formFile" class="form-label">
    <b> Select Company Logo</b>
  </label>
  <input
    class="form-control"
```

```
                    type="file"
                    id="formFile"
                    name="companyLogo"
                    onChange={(e) => setSelectImage1(e.target.files[0])}
                    required
                  />
                </div>

                <div className="d-flex aligns-items-center justify-content-center mb-2">
                  <button
                    type="submit"
                    class="btn bg-color custom-bg-text"
                    onClick={saveJob}
                  >
                    Post Job
                  </button>
                </div>
              </form>
            </div>
          </div>
        </div>
      </div>
    </div>
  );
};

export default AddJobForm;
```

```
package com.jobportal.entity;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.JoinColumn;
import jakarta.persistence.ManyToOne;

@Entity
public class Job {
```

```
@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
private int id;

@ManyToOne
@JoinColumn(name = "employer_id")
private User employer;

private String title;

private String description;

@ManyToOne
@JoinColumn(name = "category_id")
private JobCategory category;

private String companyName;

private String companyLogo;

@ManyToOne
@JoinColumn(name = "address_id")
private Address address;

private String jobType; // full time, part time, contract

private String salaryRange;

private String experienceLevel;

private String requiredSkills;

private String status;

private String datePosted;

private int applicationCount;

public int getId() {
  return id;
}
```

```
    public void setId(int id) {
        this.id = id;
    }

    public User getEmployer() {
        return employer;
    }

    public void setEmployer(User employer) {
        this.employer = employer;
    }

    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }

    public JobCategory getCategory() {
        return category;
    }

    public void setCategory(JobCategory category) {
        this.category = category;
    }

    public String getCompanyName() {
        return companyName;
    }

    public void setCompanyName(String companyName) {
        this.companyName = companyName;
    }

    public String getCompanyLogo() {
      return companyLogo;
    }

    public void setCompanyLogo(String companyLogo) {
      this.companyLogo = companyLogo;
    }

    public Address getAddress() {
      return address;
    }

    public void setAddress(Address address) {
      this.address = address;
    }

    public String getJobType() {
      return jobType;
    }

    public void setJobType(String jobType) {
      this.jobType = jobType;
    }

    public String getSalaryRange() {
      return salaryRange;
    }

    public void setSalaryRange(String salaryRange) {
      this.salaryRange = salaryRange;
    }

    public String getExperienceLevel() {
      return experienceLevel;
    }

    public void setExperienceLevel(String experienceLevel) {
      this.experienceLevel = experienceLevel;
    }
    public String getRequiredSkills() {
      return requiredSkills;
    }

    public void setRequiredSkills(String requiredSkills) {
      this.requiredSkills = requiredSkills;
    }

    public String getStatus() {
      return status;
    }

    public void setStatus(String status) {
      this.status = status;
    }

    public String getDatePosted() {
      return datePosted;
    }

    public void setDatePosted(String datePosted) {
      this.datePosted = datePosted;
    }

    public int getApplicationCount() {
      return applicationCount;
    }

    public void setApplicationCount(int applicationCount) {
      this.applicationCount = applicationCount;
    }

}


package com.jobportal.entity;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.JoinColumn;
import jakarta.persistence.ManyToOne;
import jakarta.persistence.Transient;

@Entity
public class JobApplication {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    private String applicationId;

    @ManyToOne
    @JoinColumn(name = "job_id")
    private Job job;

    @ManyToOne
    @JoinColumn(name = "employee_id")
    private User employee;

    private String dateTime;

    private String status; // Applied, Shortlisted, Rejected

    @Transient
    private int jobId;

    @Transient
    private int employeeId;

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getApplicationId() {
        return applicationId;
    }
    public void setApplicationId(String applicationId) {
        this.applicationId = applicationId;
    }

    public Job getJob() {
        return job;
    }

    public void setJob(Job job) {
        this.job = job;
    }

    public User getEmployee() {
        return employee;
    }

    public void setEmployee(User employee) {
        this.employee = employee;
    }

    public String getDateTime() {
        return dateTime;
    }

    public void setDateTime(String dateTime) {
        this.dateTime = dateTime;
    }

    public String getStatus() {
        return status;
    }

    public void setStatus(String status) {
        this.status = status;
    }

    public int getJobId() {
        return jobId;
    }

    public void setJobId(int jobId) {
        this.jobId = jobId;
    }

    public int getEmployeeId() {
        return employeeId;
    }

    public void setEmployeeId(int employeeId) {
        this.employeeId = employeeId;
    }

}


package com.jobportal.entity;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;

@Entity

public class JobCategory {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    private String name;

    private String description;

    private String status;

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }
```

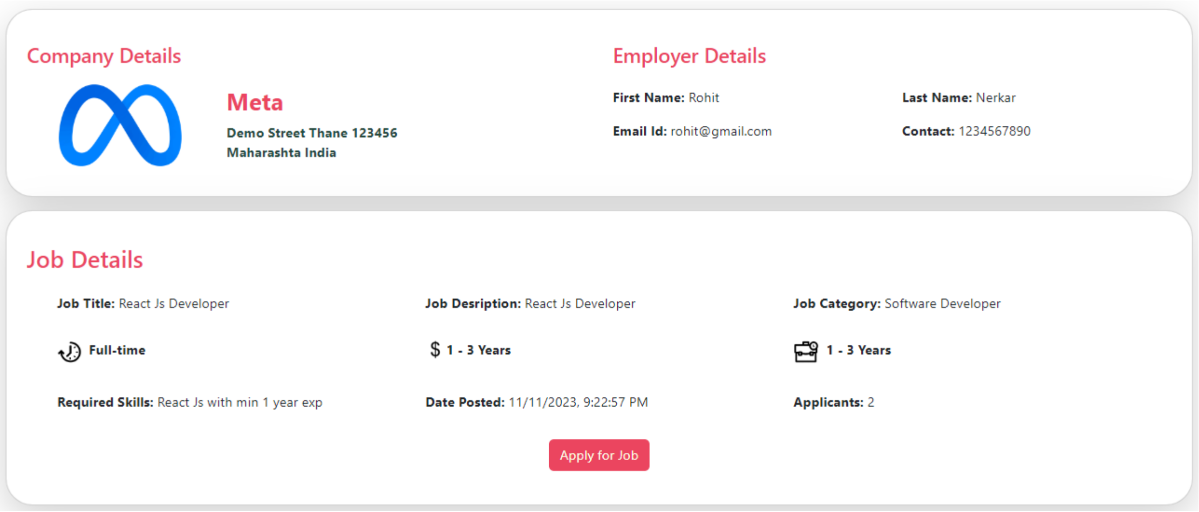
```java
    public String getName() {
      return name;
    }

    public void setName(String name) {
      this.name = name;
    }

    public String getDescription() {
      return description;
    }

    public void setDescription(String description) {
      this.description = description;
    }

    public String getStatus() {
      return status;
    }

    public void setStatus(String status) {
      this.status = status;
    }

}


package com.jobportal.dto;

import org.springframework.beans.BeanUtils;
import org.springframework.web.multipart.MultipartFile;

import com.jobportal.entity.Address;
import com.jobportal.entity.Job;

import jakarta.validation.constraints.NotBlank;
import jakarta.validation.constraints.NotNull;

public class JobAddRequest {

    @NotNull
    private Integer employerId; // user id with role employer
```

```
@NotNull
private Integer jobCategoryId;

@NotBlank
private String title;

@NotBlank
private String description;

@NotBlank
private String companyName;

@NotBlank
private MultipartFile companyLogo;

@NotBlanka
private String jobType; // full time, part time, contract

@NotBlank
private String salaryRange;

@NotBlank
private String experienceLevel;

@NotBlank
private String requiredSkills;

// address
@NotBlank
private String street;

@NotBlank
private String city;

@NotBlank
private String pincode;

@NotBlank
private String state;

@NotBlank
```

```
  private String country;

  public Integer getEmployerId() {
    return employerId;
  }

  public void setEmployerId(Integer employerId) {
    this.employerId = employerId;
  }

  public Integer getJobCategoryId() {
    return jobCategoryId;
  }

  public void setJobCategoryId(Integer jobCategoryId) {
    this.jobCategoryId = jobCategoryId;
  }

  public String getTitle() {
    return title;
  }

  public void setTitle(String title) {
    this.title = title;
  }

  public String getDescription() {
    return description;
  }

  public void setDescription(String description) {
    this.description = description;
  }

  public String getCompanyName() {
    return companyName;
  }

  public void setCompanyName(String companyName) {
    this.companyName = companyName;
  }
```

```
    public MultipartFile getCompanyLogo() {
        return companyLogo;
    }

    public void setCompanyLogo(MultipartFile companyLogo) {
        this.companyLogo = companyLogo;
    }

    public String getJobType() {
        return jobType;
    }

    public void setJobType(String jobType) {
        this.jobType = jobType;
    }

    public String getSalaryRange() {
        return salaryRange;
    }

    public void setSalaryRange(String salaryRange) {
        this.salaryRange = salaryRange;
    }

    public String getExperienceLevel() {
        return experienceLevel;
    }

    public void setExperienceLevel(String experienceLevel) {
        this.experienceLevel = experienceLevel;
    }

    public String getRequiredSkills() {
        return requiredSkills;
    }

    public void setRequiredSkills(String requiredSkills) {
        this.requiredSkills = requiredSkills;
    }

    public String getStreet() {
        return street;
    }

    public void setStreet(String street) {
        this.street = street;
    }

    public String getCity() {
        return city;
    }

    public void setCity(String city) {
        this.city = city;
    }

    public String getPincode() {
        return pincode;
    }

    public void setPincode(String pincode) {
        this.pincode = pincode;
    }

    public String getState() {
        return state;
    }

    public void setState(String state) {
        this.state = state;
    }

    public String getCountry() {
        return country;
    }

    public void setCountry(String country) {
        this.country = country;
    }

    public static Job toJobEntity(JobAddRequest jobAddRequest) {
        Job job = new Job();
        BeanUtils.copyProperties(jobAddRequest, job, "companyLogo", "employerId", "jobCategoryId");
        return job;
    }

    public static Address toAddressEntity(JobAddRequest jobAddRequest) {
        Address address = new Address();
        BeanUtils.copyProperties(jobAddRequest, address);
        return address;
    }

}
```

Posted Job Details:-



Company Details

Meta

Demo Street Thane 123456
Maharashta India

Employer Details

First Name: Rohit

Last Name: Nerkar

Email Id: rohit@gmail.com

Contact: 1234567890

Job Details

Job Title: React Js Developer

Job Desription: React Js Developer

Job Category: Software Developer

Full-time

$ **1 - 3 Years**

1 - 3 Years

Required Skills: React Js with min 1 year exp

Date Posted: 11/11/2023, 9:22:57 PM

Applicants: 2

Code:-

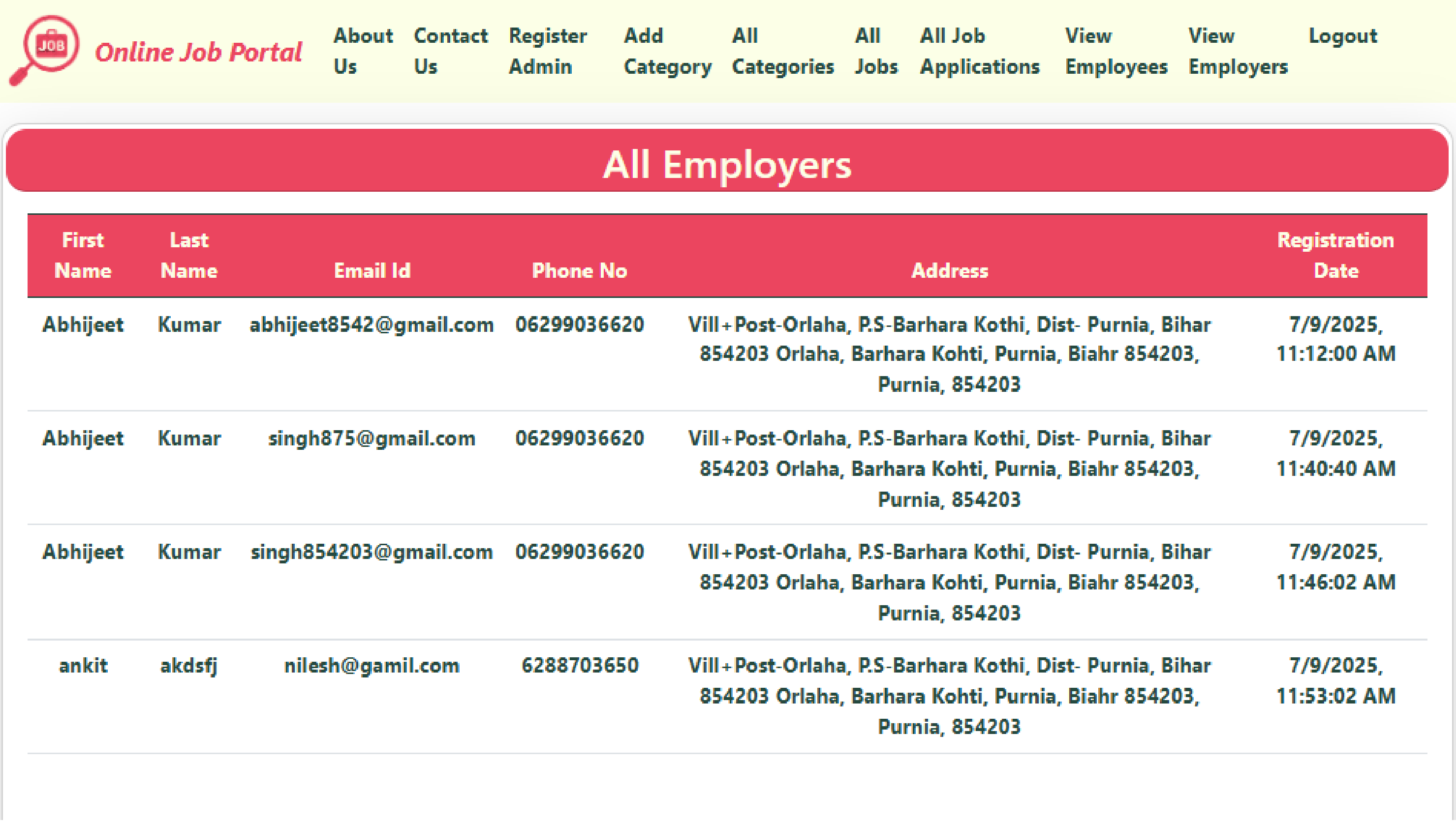
```
import { useParams, useNavigate } from "react-router-dom";
import { useState, useEffect } from "react";
import axios from "axios";
import dollor from "../images/dollor_logo.png";
import timing from "../images/timing_logo.png";
import experience from "../images/experience_logo.png";
import { ToastContainer, toast } from "react-toastify";

const JobDetailPage = () => {
  const { jobId } = useParams();

  const employee = JSON.parse(sessionStorage.getItem("active-employee"));
  const employee_jwtToken = sessionStorage.getItem("employee-jwtToken");

  const navigate = useNavigate();

  const [jobApplyRequest, setJobApplyRequest] = useState({
    jobId: "",
    employeeId: "",
  });

  const [job, setJob] = useState({
    id: "",
    employer: {
      id: "",
      firstName: "",
      lastName: "",
      emailId: "",
      phoneNo: "",
      role: "",
      address: {
        id: "",
        street: "",
        city: "",
        pincode: "",
        state: "",
        country: "",
      },
      registrationDate: "",
      status: "",
    },
    title: "",
    description: "",
    category: {
      id: "",
      name: "",
      description: "",
      status: "",
    },
    companyName: "",
    companyLogo: "",
    address: {
      id: "",
      street: "",
      city: "",
      pincode: "",
      state: "",
      country: "",
    },
    jobType: "",
    salaryRange: "",
    experienceLevel: "",
    requiredSkills: "",
    status: "",
    datePosted: "",
    applicationCount: "",
  });

  useEffect(() => {
    const getJob = async () => {
      const fetchJobResponse = await retrieveJob();
      if (fetchJobResponse) {
        setJob(fetchJobResponse.jobs[0]);
      }
    };
    getJob();
  }, []);

  const retrieveJob = async () => {
    const response = await axios.get(
      "http://localhost:8080/api/job/fetch?jobId=" + jobId
    );
    console.log(response.data);
    return response.data;
  };

  const formatDateFromEpoch = (epochTime) => {
    const date = new Date(Number(epochTime));
    const formattedDate = date.toLocaleString(); // Adjust the format as needed

    return formattedDate;
  };

  const applyForJob = (jobId, e) => {
    e.preventDefault();
    if (employee_jwtToken === null || employee_jwtToken === "") {
      toast.error("Please login as employee, for applying any Job", {
        position: "top-center",
        autoClose: 1000,
        hideProgressBar: false,
        closeOnClick: true,
        pauseOnHover: true,
        draggable: true,
        progress: undefined,
      });
    } else {
      jobApplyRequest.employeeId = employee.id;
      jobApplyRequest.jobId = jobId;

      fetch("http://localhost:8080/api/job/application/add", {
        method: "POST",
        headers: {
          Accept: "application/json",
          "Content-Type": "application/json",
          //   Authorization: "Bearer " + admin_jwtToken,
        },
        body: JSON.stringify(jobApplyRequest),
      })
        .then((result) => {
          console.log("result", result);
          result.json().then((res) => {
            if (res.success) {
              toast.success(res.responseMessage, {
                position: "top-center",
                autoClose: 1000,
                hideProgressBar: false,
                closeOnClick: true,
                pauseOnHover: true,
                draggable: true,
                progress: undefined,
              });

              setTimeout(() => {
                navigate("/home");
              }, 1000);
            } else if (!res.success) {
              toast.error(res.responseMessage, {
                position: "top-center",
                autoClose: 1000,
                hideProgressBar: false,
                closeOnClick: true,
                pauseOnHover: true,
                draggable: true,
                progress: undefined,
              });

              setTimeout(() => {
                window.location.reload(true);
              }, 1000); // Redirect after 3 seconds
            } else {
              toast.error("It seems server is down", {
                position: "top-center",
                autoClose: 1000,
                hideProgressBar: false,
                closeOnClick: true,
                pauseOnHover: true,
                draggable: true,
                progress: undefined,
              });

              setTimeout(() => {
                window.location.reload(true);
              }, 1000); // Redirect after 3 seconds
            }
          });
        })
        .catch((error) => {
          console.error(error);
```

```
      toast.error("It seems server is down", {
       position: "top-center",
       autoClose: 1000,
       hideProgressBar: false,
       closeOnClick: true,
       pauseOnHover: true,
       draggable: true,
       progress: undefined,
      });
     });
   }
  };

  return (
   <div className="mb-3">
    <div className="col ml-5 mt-3 ms-5 me-5">
     {/* Company and Employer Details Card */}
     <div className="card rounded-card h-100 shadow-lg ">
      <div className="row g-0">
       {/* Left side - Company Details Card */}
       <div className="col-md-6">
        <div className="card-body">
         <h4 className="card-title text-color-second">
          Company Details
         </h4>
         <div className="row g-0">
          {/* Left side - Company Logo */}
          <div className="col-md-4 d-flex align-items-center justify-content-center">
           <img
            src={"http://localhost:8080/api/job/" + job.companyLogo}
            className="card-img-top rounded img-fluid"
            alt="Company Logo"
            style={{
             maxHeight: "100px",
             width: "auto",
            }}
           />
          </div>
          {/* Right side - Job Details */}
          <div className="col-md-8">
           <div className="card-body text-color">
            <h3 className="card-title d-flex justify-content-between text-color-second">
             <div>
              <b>{job.companyName}</b>
             </div>
            </h3>
            <b className="card-text">
             {job.address.street +
              " " +
              job.address.city +
              " " +
              job.address.pincode}
            </b>
            <br />
            <b className="card-text">
             {job.address.state + " " + job.address.country}
            </b>
           </div>
          </div>
         </div>
        </div>
       </div>

       {/* Right side - Employer Details Card */}
       <div className="col-md-6">
        <div className="card-body">
         <h4 className="card-title text-color-second">
          Employer Details
         </h4>
         {/* Include the necessary details for the employer */}
         {/* Display First Name and Last Name in a row */}
         <div className="row mt-4">
          <div className="col-md-6">
           <p className="mb-2">
            <b>First Name:</b> {job.employer.firstName}
           </p>
          </div>
          <div className="col-md-6">
           <p className="mb-2">
            <b>Last Name:</b> {job.employer.lastName}
           </p>
          </div>
         </div>
         <div className="row mt-2">
          <div className="col-md-6">
           <p className="mb-2">
            <b>Email Id:</b> {job.employer.emailId}
           </p>
          </div>
          <div className="col-md-6">
           <p className="mb-2">
            <b>Contact:</b> {job.employer.phoneNo}
           </p>
          </div>
         </div>
        </div>
```

```
      </div>
    </div>
  </div>

  {/* New Row - Job Details Card */}
  <div className="row mt-3">
    <div className="col">
      <div className="card rounded-card h-100 shadow-lg">
        <div className="card-body">
          <h3 className="card-title text-color-second">Job Details</h3>
          {/* Row 1 - Job Title */}
          <div className="row mt-4 ms-4 me-4">
            <div className="col-md-4">
              <p className="mb-2">
                <b className="mb-2">Job Title:</b> {job.title}
              </p>
            </div>
            <div className="col-md-4">
              <p className="mb-2">
                <b>Job Desription:</b> {job.description}
              </p>
            </div>
            <div className="col-md-4">
              <p className="mb-2">
                <b>Job Category:</b> {job.category.name}
              </p>
            </div>
          </div>

          {/* Row 2 - Job Title */}
          <div className="row mt-4 ms-4 me-4">
            <div className="col-md-4">
              <p className="mb-2">
                <b>
                  <img
                    src={timing}
                    height="30"
                    width="auto"
                    class="d-inline-block align-top me-2"
                    alt=""
                  />
                  {job.jobType}
                </b>
              </p>
            </div>
            <div className="col-md-4">
              <p className="mb-2">
                <b>
                  <img
                    src={dollor}
                    height="25"
                    width="auto"
                    class="d-inline-block align-top"
                    alt=""
                  />
                  {job.salaryRange}
                </b>
              </p>
            </div>
            <div className="col-md-4">
              <p className="mb-2">
                <b>
                  <img
                    src={experience}
                    height="28"
                    width="auto"
                    class="d-inline-block align-top me-2"
                    alt=""
                  />
                  {job.experienceLevel}
                </b>
              </p>
            </div>
          </div>

          {/* Row 3 - Job Title */}
          <div className="row mt-4 ms-4 me-4">
            <div className="col-md-4">
              <p className="mb-2">
                <b>Required Skills:</b> {job.requiredSkills}
              </p>
            </div>
            <div className="col-md-4">
              <p className="mb-2">
                <b>Date Posted:</b> {formatDateFromEpoch(job.datePosted)}
              </p>
            </div>
            <div className="col-md-4">
              <p className="mb-2">
                <b>Applicants:</b> {job.applicationCount}
              </p>
            </div>
          </div>

          <div className="d-flex justify-content-center mt-4">
            <button
```

```
                    type="button"
                    className="btn bg-color custom-bg-text mb-3"
                    onClick={(e) => applyForJob(job.id, e)}
                  >
                    Apply for Job
                  </button>
                  <ToastContainer />
                </div>
              </div>
            </div>
          </div>
        </div>
      </div>
    </div>
  );
};

export default JobDetailPage;
```

All Employers Details:-

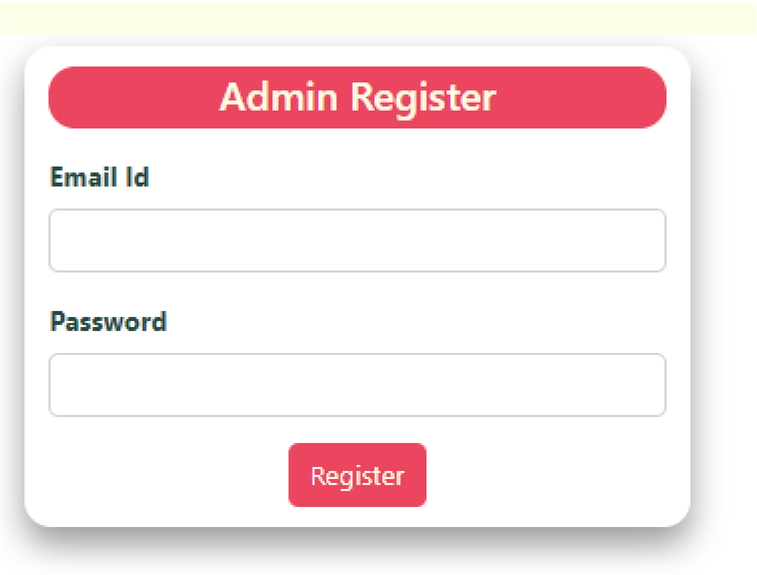


All Employers

First Name	Last Name	Email Id	Phone No	Address	Registration Date
Abhijeet	Kumar	abhijeet8542@gmail.com	06299036620	Vill+Post-Orlaha, P.S-Barhara Kothi, Dist- Purnia, Bihar 854203 Orlaha, Barhara Kohti, Purnia, Biahr 854203, Purnia, 854203	7/9/2025, 11:12:00 AM
Abhijeet	Kumar	singh875@gmail.com	06299036620	Vill+Post-Orlaha, P.S-Barhara Kothi, Dist- Purnia, Bihar 854203 Orlaha, Barhara Kohti, Purnia, Biahr 854203, Purnia, 854203	7/9/2025, 11:40:40 AM
Abhijeet	Kumar	singh854203@gmail.com	06299036620	Vill+Post-Orlaha, P.S-Barhara Kothi, Dist- Purnia, Bihar 854203 Orlaha, Barhara Kohti, Purnia, Biahr 854203, Purnia, 854203	7/9/2025, 11:46:02 AM
ankit	akdsfj	nilesh@gamil.com	6288703650	Vill+Post-Orlaha, P.S-Barhara Kothi, Dist- Purnia, Bihar 854203 Orlaha, Barhara Kohti, Purnia, Biahr 854203, Purnia, 854203	7/9/2025, 11:53:02 AM

```
import { useState, useEffect } from "react";
import axios from "axios";
import React from "react";

const ViewAllEmployers = () => {
  const [allEmployer, setAllEmployer] = useState([]);
  const admin_jwtToken = sessionStorage.getItem("admin-jwtToken");

  useEffect(() => {
    const getAllUsers = async () => {
      const allUsers = await retrieveAllUser();
      if (allUsers) {
        setAllEmployer(allUsers.users);
      }
    };

    getAllUsers();
  }, []);

  const retrieveAllUser = async () => {
    const response = await axios.get(
      "http://localhost:8080/api/user/fetch/role-wise?role=Employer",
      {
        headers: {
          Authorization: "Bearer " + admin_jwtToken, // Replace with your actual JWT token
        },
      }
    );
    console.log(response.data);
    return response.data;
  };

  const formatDateFromEpoch = (epochTime) => {
    const date = new Date(Number(epochTime));
    const formattedDate = date.toLocaleString(); // Adjust the format as needed

    return formattedDate;
  };

  return (
    <div className="mt-3">
      <div
        className="card form-card ms-2 me-2 mb-5 shadow-lg"
        style={{
          height: "45rem",
        }}
      >
        <div
          className="card-header custom-bg-text text-center bg-color"
          style={{
            borderRadius: "1em",
            height: "50px",
          }}
        >
          <h2>All Employers</h2>
        </div>
        <div
          className="card-body"
          style={{
            overflowY: "auto",
          }}
        >
          <div className="table-responsive">
            <table className="table table-hover text-color text-center">
              <thead className="table-bordered border-color bg-color custom-bg-text">
                <tr>
                  <th scope="col">First Name</th>
                  <th scope="col">Last Name</th>
                  <th scope="col">Email Id</th>
                  <th scope="col">Phone No</th>
                  <th scope="col">Address</th>
                  <th scope="col">Registration Date</th>
                </tr>
              </thead>
              <tbody>
                {allEmployer.map((employer) => {
                  return (
                    <tr>
                      <td>
                        <b>{employer.firstName}</b>
                      </td>
                      <td>
                        <b>{employer.lastName}</b>
                      </td>
                      <td>
                        <b>{employer.emailId}</b>
                      </td>
                      <td>
                        <b>{employer.phoneNo}</b>
                      </td>
                      <td>
                        <b>
                          {employer.address.street +
                            ", " +
                            employer.address.city +
                            ", " +
                            employer.address.pincode}
                        </b>
                      </td>
                      <td>
                        <b>{formatDateFromEpoch(employer.registrationDate)}</b>
                      </td>
                    </tr>
                  );
                })}
              </tbody>
            </table>
          </div>
        </div>
      </div>
    </div>
  );
};

export default ViewAllEmployers;
```

Admin Register:-

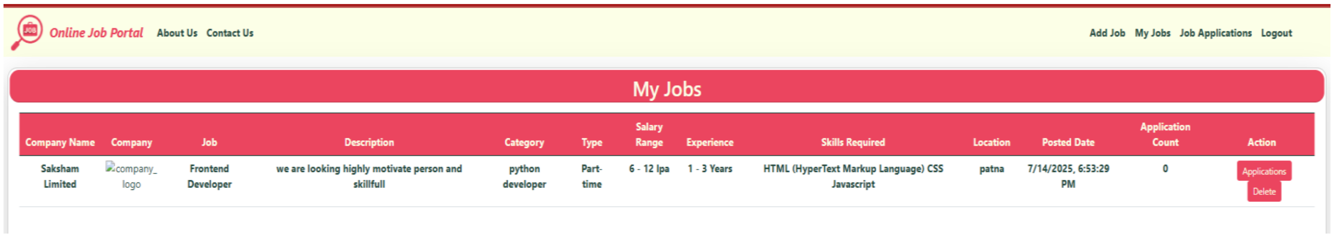


```
import { useState } from "react";
import { ToastContainer, toast } from
"react-toastify";
import "react-
toastify/dist/ReactToastify.css";
import { useNavigate } from "react-router-
dom";

const AdminRegisterForm = () => {
  let navigate = useNavigate();
  const admin_jwtToken =
sessionStorage.getItem("admin-
jwtToken");

  const [registerRequest,
setRegisterRequest] = useState({});

  const handleUserInput = (e) => {
    setRegisterRequest({ ...registerRequest,
[e.target.name]: e.target.value });
  };

  const registerAdmin = (e) => {
    fetch("http://localhost:8080/api/user/a
dmin/register", {
      method: "POST",
      headers: {
        Accept: "application/json",
        "Content-Type": "application/json",
        Authorization: "Bearer " +
admin_jwtToken,
      },
      body: JSON.stringify(registerRequest),
    })
      .then((result) => {
        console.log("result", result);
        result.json().then((res) => {
          if (res.success) {
            toast.success(res.responseMessage,
{
              position: "top-center",
              autoClose: 1000,
              hideProgressBar: false,
              closeOnClick: true,
              pauseOnHover: true,
              draggable: true,
              progress: undefined,
            });

            setTimeout(() => {
              navigate("/home");
            }, 1000);
          } else if (!res.success) {
            toast.error(res.responseMessage, {
              position: "top-center",
              autoClose: 1000,
              hideProgressBar: false,
              closeOnClick: true,
              pauseOnHover: true,
              draggable: true,
              progress: undefined,
      });

      setTimeout(() => {
       window.location.reload(true);
      }, 1000); // Redirect after 3 seconds
     } else {
      toast.error("It seems server is
down", {
       position: "top-center",
       autoClose: 1000,
       hideProgressBar: false,
       closeOnClick: true,
       pauseOnHover: true,
       draggable: true,
       progress: undefined,
      });

      setTimeout(() => {
       window.location.reload(true);
      }, 1000); // Redirect after 3 seconds
     }
    });
   })
   .catch((error) => {
    console.error(error);
    toast.error("It seems server is down",
{
     position: "top-center",
     autoClose: 1000,
     hideProgressBar: false,
     closeOnClick: true,
     pauseOnHover: true,
     draggable: true,
     progress: undefined,
    });
   });
  e.preventDefault();
 };

 return (
  <div>
   <div className="mt-2 d-flex aligns-
```

```
items-center justify-content-center">
    <div className="form-card border-
color mb-2" style={{ width: "25rem" }}>
     <div className="container-fluid">
      <div
       className="card-header bg-color
custom-bg-text mt-2 d-flex justify-
content-center align-items-center"
       style={{
        borderRadius: "1em",
        height: "38px",
       }}
      >
       <h4 className="card-
title">Admin Register</h4>
      </div>
      <div className="card-body mt-3">
       <form>
        <div className="mb-3 text-
color">
         <label for="emailId"
className="form-label">
          <b>Email Id</b>
         </label>
         <input
          type="email"
          className="form-control"
          id="email"
          name="emailId"
          onChange={handleUserInput}
          value={registerRequest.emailId
}
         />
        </div>
        <div className="mb-3 text-
color">
         <label for="password"
className="form-label">
          <b>Password</b>
         </label>
         <input
          type="password"
```

```
                    className="form-control"
                    id="password"
                    name="password"
                    onChange={handleUserInput}
                    value={registerRequest.password}
                    autoComplete="on"
                  />
                </div>
                <div className="d-flex aligns-items-center justify-content-center">
                  <button
                    type="submit"
                    className="btn bg-color custom-bg-text mb-2"
                    onClick={registerAdmin}
                  >
                    Register
                  </button>
                </div>

                <ToastContainer />
              </form>
            </div>
          </div>
        </div>
      </div>
    </div>
  );
};

export default AdminRegisterForm;
```

Posted Jobs:



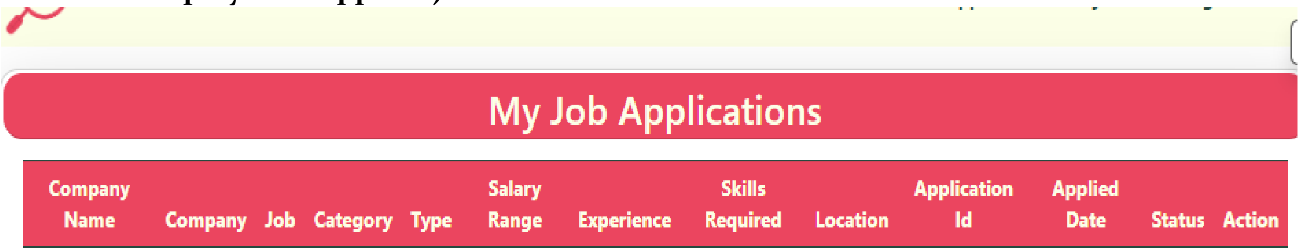
My Jobs

Company Name	Company	Job	Description	Category	Type	Salary Range	Experience	Skills Required	Location	Posted Date	Application Count	Action
Saksham Limited	company_ logo	Frontend Developer	we are looking highly motivate person and skillfull	python developer	Part-time	6 - 12 lpa	1 - 3 Years	HTML (HyperText Markup Language) CSS Javascript	patna	7/14/2025, 6:53:29 PM	0	Applications Delete

```
import { useState, useEffect } from "react";
import axios from "axios";
import React from "react";
import { Link, useNavigate } from "react-router-dom";
import { toast } from "react-toastify";

const ViewEmployerJobs = () => {
  const employer = JSON.parse(sessionStorage.getItem("active-employer"));
  const employer_jwtToken = sessionStorage.getItem("employer-jwtToken");

  const [allJobs, setAllJobs] = useState([
    {
      id: "",
      employer: {
        id: "",
        firstName: "",
        lastName: "",
        emailId: "",
        phoneNo: "",
        role: "",
        address: {
          id: "",
          street: "",
          city: "",
          pincode: "",
          state: "",
          country: "",
        },
        registrationDate: "",
        status: "",
      },
      title: "",
      description: "",
      category: {
        id: "",
        name: "",
        description: "",
        status: "",
      },
      companyName: "",
      companyLogo: "",
      address: {
        id: "",
        street: "",
        city: "",
        pincode: "",
        state: "",
        country: "",
      },
      jobType: "",
      salaryRange: "",
      experienceLevel: "",
      requiredSkills: "",
      status: "",
      datePosted: "",
      applicationCount: "",
    },
  ]);

  let navigate = useNavigate();

  useEffect(() => {
    const getAllJobs = async () => {
      const allJobs = await retrieveAllJobs();
      if (allJobs) {
        setAllJobs(allJobs.jobs);
      }
    };

    getAllJobs();
  }, []);

  const retrieveAllJobs = async () => {
    const response = await axios.get(
      "http://localhost:8080/api/job/fetch/employer-wise?employerId=" +
        employer.id,
      {
        headers: {
          Authorization: "Bearer " + employer_jwtToken, // Replace with your actual JWT token
        },
      }
    );
    console.log(response.data);
    return response.data;
  };

  const deleteJob = (jobId, e) => {
    fetch("http://localhost:8080/api/job/delete?jobId=" + jobId, {
      method: "DELETE",
      headers: {
        Accept: "application/json",
        "Content-Type": "application/json",
        Authorization: "Bearer " + employer_jwtToken,
      },
  })
    .then((result) => {
      result.json().then((res) => {
        if (res.success) {
          toast.success(res.responseMessage, {
            position: "top-center",
            autoClose: 1000,
            hideProgressBar: false,
            closeOnClick: true,
            pauseOnHover: true,
            draggable: true,
            progress: undefined,
          });

          setTimeout(() => {
            window.location.reload(true);
          }, 1000); // Redirect after 3 seconds
        } else if (!res.success) {
          toast.error(res.responseMessage, {
            position: "top-center",
            autoClose: 1000,
            hideProgressBar: false,
            closeOnClick: true,
            pauseOnHover: true,
            draggable: true,
            progress: undefined,
          });
          setTimeout(() => {
            window.location.reload(true);
          }, 1000); // Redirect after 3 seconds
        }
      });
    })
    .catch((error) => {
      console.error(error);
      toast.error("It seems server is down", {
        position: "top-center",
        autoClose: 1000,
        hideProgressBar: false,
        closeOnClick: true,
        pauseOnHover: true,
        draggable: true,
        progress: undefined,
      });
      setTimeout(() => {
        window.location.reload(true);
      }, 1000); // Redirect after 3 seconds
    });
  };

  const formatDateFromEpoch = (epochTime) => {
    const date = new Date(Number(epochTime));
    const formattedDate = date.toLocaleString(); // Adjust the format as needed

    return formattedDate;
  };

  const viewAppliedJobs = (jobId) => {
    navigate(`/job/${jobId}/application/all`);
  };

  return (
    <div className="mt-3">
      <div
        className="card form-card ms-2 me-2 mb-5 shadow-lg"
        style={{
          height: "45rem",
        }}
      >
        <div
          className="card-header custom-bg-text text-center bg-color"
          style={{
            borderRadius: "1em",
            height: "50px",
          }}
        >
          <h2>My Jobs</h2>
        </div>
        <div
          className="card-body"
          style={{
            overflowY: "auto",
          }}
        >
          <div className="table-responsive">
            <table className="table table-hover text-color text-center">
              <thead className="table-bordered border-color bg-color custom-bg-text">
                <tr>
                  <th scope="col">Company Name</th>
                  <th scope="col">Company</th>
                  <th scope="col">Job</th>
                  <th scope="col">Description</th>
                  <th scope="col">Category</th>
                  <th scope="col">Type</th>
                  <th scope="col">Salary Range</th>
                  <th scope="col">Experience</th>
                  <th scope="col">Skills Required</th>
                  <th scope="col">Location</th>
                  <th scope="col">Posted Date</th>
                  <th scope="col">Application Count</th>
                  <th scope="col">Action</th>
                </tr>
              </thead>
              <tbody>
                {allJobs.map((job) => {
                  return (
                    <tr>
                      <td>
                        <b>{job.companyName}</b>
                      </td>
                      <td>
                        <img
                          src={
                            "http://localhost:8080/api/job/" + job.companyLogo
                          }
                          class="img-fluid"
                          alt="company_logo"
                          style={{
                            maxWidth: "90px",
                          }}
                        />
                      </td>
                      <td>
                        <Link
                          to={`/job/${job.id}/detail`}
                          style={{ textDecoration: "none" }}
                          className="text-color"
                        >
                          <b>{job.title}</b>
                        </Link>
                      </td>

                      <td>
                        <b>{job.description}</b>
                      </td>
                      <td>
                        <b>{job.category.name}</b>
                      </td>
                      <td>
                        <b>{job.jobType}</b>
                      </td>
                      <td>
                        <b>{job.salaryRange}</b>
                      </td>
```

```
                      <td>
                        <b>{job.experienceLevel}</b>
                      </td>
                      <td>
                        <b>{job.requiredSkills}</b>
                      </td>
                      <td>
                        <b>{job.address.city}</b>
                      </td>
                      <td>
                        <b>{formatDateFromEpoch(job.datePosted)}</b>
                      </td>
                      <td>
                        <b>{job.applicationCount}</b>
                      </td>
                      <td>
                        <button
                          onClick={() => viewAppliedJobs(job.id)}
                          className="btn btn-sm bg-color custom-bg-text ms-2"
                        >
                          Applications
                        </button>
                        <button
                          onClick={() => deleteJob(job.id)}
                          className="btn btn-sm bg-color custom-bg-text ms-2"
                        >
                          Delete
                        </button>
                      </td>
                    </tr>
                  );
                })}
              </tbody>
            </table>
          </div>
        </div>
      </div>
    </div>
  );
};

export default ViewEmployerJobs;
```

Employer all applied job



My Job Applications

Company Name	Company	Job	Category	Type	Salary Range	Experience	Skills Required	Location	Application Id	Applied Date	Status	Action

```
import { useState, useEffect } from "react";
import axios from "axios";
import React from "react";
import { Link, useNavigate } from "react-router-dom";
import { toast } from "react-toastify";

const ViewEmployeeJobApplication = () => {
  const employee = JSON.parse(sessionStorage.getItem("active-employee"));
  const employee_jwtToken = sessionStorage.getItem("employee-jwtToken");

  const [applications, setApplications] = useState([
    {
      id: 1,
      applicationId: "",
      job: {
        id: 1,
        employer: {
          id: 4,
          firstName: "",
          lastName: "",
          emailId: "",
          phoneNo: "",
        },
        title: "",
        description: "",
        category: {
          id: 1,
          name: "",
          description: "",
          status: "",
        },
        address: {
          id: "",
          street: "",
          city: "",
          pincode: "",
          state: "",
          country: "",
        },
        companyName: "",
        companyLogo: "",
        jobType: "",
        salaryRange: "",
        experienceLevel: "",
        requiredSkills: "",
        status: "",
        datePosted: "",
        applicationCount: 0,
      },
      employee: {
        userProfile: {
          id: 1,
          bio: "",
          website: "",
          resume: "",
          linkedInProfileLink: "",
          githubProfileLink: "",
          skills: [
            {
              id: 7,
              skill: "",
              experience: 0,
              userId: 0,
            },
          ],
          educations: [
            {
              id: 5,
              degree: "",
              institution: "",
              startDate: "",
              endDate: "",
              userId: 0,
            },
          ],
          workExperiences: [
            {
              id: 2,
              experience: 0,
              jobPosition: "",
              company: "",
              startDate: "",
              endDate: "",
              userId: 0,
            },
          ],
          profilePic: "",
        },
      },
      dateTime: "",
      status: "",
      jobId: 0,
      employeeId: 0,
    },
  ]);

  let navigate = useNavigate();

  useEffect(() => {
    const getAllJobs = async () => {
      const jobApplicationResponse = await retrieveAllJobApplication();
      if (jobApplicationResponse) {
        setApplications(jobApplicationResponse.applications);
      }
    };

    getAllJobs();
  }, []);

  const retrieveAllJobApplication = async () => {
    const response = await axios.get(
      "http://localhost:8080/api/job/application/fetch/all/employee?employeeId=" +
        employee.id,
      {
        headers: {
          Authorization: "Bearer " + employee_jwtToken, // Replace with your actual JWT token
        },
      }
    );
    console.log(response.data);
    return response.data;
  };

  const cancelApplication = (applicationId, e) => {
    fetch("http://localhost:8080/api/job/application/update/status", {
      method: "PUT",
      headers: {
        Accept: "application/json",
        "Content-Type": "application/json",
        Authorization: "Bearer " + employee_jwtToken,
      },
      body: JSON.stringify({
        id: applicationId,
        status: "Cancelled",
      }),
    })
      .then((result) => {
        result.json().then((res) => {
          if (res.success) {
            toast.success(res.responseMessage, {
              position: "top-center",
              autoClose: 1000,
              hideProgressBar: false,
              closeOnClick: true,
              pauseOnHover: true,
              draggable: true,
              progress: undefined,
            });

            setTimeout(() => {
              window.location.reload(true);
            }, 1000); // Redirect after 3 seconds
          } else if (!res.success) {
            toast.error(res.responseMessage, {
              position: "top-center",
              autoClose: 1000,
              hideProgressBar: false,
              closeOnClick: true,
              pauseOnHover: true,
              draggable: true,
              progress: undefined,
            });
            setTimeout(() => {
              window.location.reload(true);
            }, 1000); // Redirect after 3 seconds
          }
        });
      })
      .catch((error) => {
        console.error(error);
        toast.error("It seems server is down", {
          position: "top-center",
          autoClose: 1000,
          hideProgressBar: false,
          closeOnClick: true,
          pauseOnHover: true,
          draggable: true,
          progress: undefined,
        });
        setTimeout(() => {
          window.location.reload(true);
        }, 1000); // Redirect after 3 seconds
      });
  };

  const formatDateFromEpoch = (epochTime) => {
    const date = new Date(Number(epochTime));
    const formattedDate = date.toLocaleString(); // Adjust the format as needed

    return formattedDate;
  };

  return (
    <div className="mt-3">
      <div
        className="card form-card ms-2 me-2 mb-5 shadow-lg"
        style={{
          height: "45rem",
        }}
      >
        <div
          className="card-header custom-bg-text text-center bg-color"
          style={{
            borderRadius: "1em",
            height: "50px",
          }}
        >
          <h2>My Job Applications</h2>
        </div>
        <div
          className="card-body"
          style={{
            overflowY: "auto",
          }}
        >
          <div className="table-responsive">
            <table className="table table-hover text-color text-center">
              <thead className="table-bordered border-color bg-color custom-bg-text">
                <tr>
                  <th scope="col">Company Name</th>
                  <th scope="col">Company</th>
                  <th scope="col">Job</th>
                  <th scope="col">Category</th>
                  <th scope="col">Type</th>
                  <th scope="col">Salary Range</th>
                  <th scope="col">Experience</th>
                  <th scope="col">Skills Required</th>
                  <th scope="col">Location</th>
                  <th scope="col">Application Id</th>
                  <th scope="col">Applied Date</th>
                  <th scope="col">Status</th>
                  <th scope="col">Action</th>
                </tr>
              </thead>
              <tbody>
                {applications.map((application) => {
                  return (
                   <tr>
                    <td>
                     <b>{application.job.companyName}</b>
                    </td>
                    <td>
                     <img
                      src={
                       "http://localhost:8080/api/job/" +
                       application.job.companyLogo
                      }
                      class="img-fluid"
                      alt="food_pic"
                      style={{
                       maxWidth: "90px",
                      }}
                     />
                    </td>
                    <td>
                     <Link
                      to={`/job/${application.job.id}/detail`}
                      style={{ textDecoration: "none" }}
                      className="text-color"
                     >
                      <b>{application.job.title}</b>
                     </Link>
                    </td>

                    <td>
                     <b>{application.job.category.name}</b>
                    </td>
                    <td>
                     <b>{application.job.jobType}</b>
                    </td>
                    <td>
                     <b>{application.job.salaryRange}</b>
                    </td>
                    <td>
                     <b>{application.job.experienceLevel}</b>
```

```
                      </td>
                      <td>
                        <b>{application.job.requiredSkills}</b>
                      </td>
                      <td>
                        <b>{application.job.address.city}</b>
                      </td>
                      <td>
                        <b>{application.applicationId}</b>
                      </td>
                      <td>
                        <b>{formatDateFromEpoch(application.dateTime)}</b>
                      </td>
                      <td>
                        <b>{application.status}</b>
                      </td>
                      <td>
                        {(() => {
                          if (application.status === "Applied") {
                            return (
                              <div>
                                <input
                                  type="submit"
                                  className="btn bg-color custom-bg-text mb-3"
                                  value="Cancel"
                                  onClick={() =>
                                    cancelApplication(application.id)
                                  }
                                />
                              </div>
                            );
                          }
                        })()}
                      </td>
                    </tr>
                  );
                })}
              </tbody>
            </table>
          </div>
        </div>
      </div>
    </div>
  );
};

export default ViewEmployeeJobApplication;
```

```
package com.jobportal.resource;

import java.time.LocalDateTime;
import java.time.ZoneId;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Component;
import org.springframework.util.CollectionUtils;

import com.jobportal.dto.CommonApiResponse;
import com.jobportal.dto.JobApplicationResponse;
import com.jobportal.entity.Job;
import com.jobportal.entity.JobApplication;
import com.jobportal.entity.User;
import com.jobportal.exception.JobApplicationSaveException;
import com.jobportal.service.JobApplicationService;
import com.jobportal.service.JobService;
import com.jobportal.service.UserService;
import com.jobportal.utility.Constants.JobApplicationStatus;
import com.jobportal.utility.Helper;

@Component
public class JobApplicationResource {

  private final Logger LOG = LoggerFactory.getLogger(UserResource.class);

  @Autowired
  private JobService jobService;

  @Autowired
  private JobApplicationService jobApplicationService;

  @Autowired
  private UserService userService;

  public ResponseEntity<CommonApiResponse> addJobApplication(JobApplication request) {

    LOG.info("Request received for addding job application");
```

```
CommonApiResponse response = new CommonApiResponse();

if (request == null || request.getJobId() == 0 || request.getEmployeeId() == 0) {
  response.setResponseMessage("missing input");
  response.setSuccess(false);

  return new ResponseEntity<CommonApiResponse>(response, HttpStatus.BAD_REQUEST);
}

Job job = this.jobService.getById(request.getJobId());

if (job == null) {
  response.setResponseMessage("job not found");
  response.setSuccess(false);

  return new ResponseEntity<CommonApiResponse>(response, HttpStatus.BAD_REQUEST);
}

User employee = this.userService.getUserById(request.getEmployeeId());

if (employee == null) {
  response.setResponseMessage("employee not found");
  response.setSuccess(false);

  return new ResponseEntity<CommonApiResponse>(response, HttpStatus.BAD_REQUEST);
}

request.setEmployee(employee);
request.setJob(job);
request.setStatus(JobApplicationStatus.APPLIED.value());
request.setApplicationId(Helper.generateApplicationId());
request.setDateTime(
    String.valueOf(LocalDateTime.now().atZone(ZoneId.systemDefault()).toInstant().toEpochMilli()));

JobApplication savedApplication = this.jobApplicationService.add(request);

if (savedApplication == null) {
  throw new JobApplicationSaveException("Failed to add job application");
}

int totalJobApplicants = job.getApplicationCount();

job.setApplicationCount(totalJobApplicants + 1);

Job updatedJob = this.jobService.update(job);

if (updatedJob == null) {
  throw new JobApplicationSaveException("Failed to add job application");
```

```
        }

        response.setResponseMessage("Job Applied Successful!!!");
        response.setSuccess(true);

        return new ResponseEntity<CommonApiResponse>(response, HttpStatus.OK);

    }

    public ResponseEntity<JobApplicationResponse> fetchAllJobApplications() {

        LOG.info("Request received for fetching all job applications");

        JobApplicationResponse response = new JobApplicationResponse();

        List<JobApplication> applications = new ArrayList<>();

        applications = this.jobApplicationService.getAll();

        if (CollectionUtils.isEmpty(applications)) {
            response.setResponseMessage("No Job Applications not found!!");
            response.setSuccess(false);

            return new ResponseEntity<JobApplicationResponse>(response, HttpStatus.OK);
        }

        response.setApplications(applications);
        response.setResponseMessage("Job Application Fetched Successful");
        response.setSuccess(true);

        return new ResponseEntity<JobApplicationResponse>(response, HttpStatus.OK);
    }

    public ResponseEntity<JobApplicationResponse> fetchAllJobApplicationsByEmployee(int employeeId) {

        LOG.info("Request received for fetching all job applications by employee id");

        JobApplicationResponse response = new JobApplicationResponse();

        if (employeeId == 0) {
            response.setResponseMessage("employee not found");
            response.setSuccess(false);

            return new ResponseEntity<JobApplicationResponse>(response, HttpStatus.BAD_REQUEST);
        }

        User employee = this.userService.getUserById(employeeId);
    if (employee == null) {
      response.setResponseMessage("employee not found");
      response.setSuccess(false);

      return new ResponseEntity<JobApplicationResponse>(response, HttpStatus.BAD_REQUEST);
    }

    List<JobApplication> applications = new ArrayList<>();

    applications = this.jobApplicationService.getByEmployee(employee);

    if (CollectionUtils.isEmpty(applications)) {
      response.setResponseMessage("No Job Applications not found!!");
      response.setSuccess(false);

      return new ResponseEntity<JobApplicationResponse>(response, HttpStatus.OK);
    }

    response.setApplications(applications);
    response.setResponseMessage("Job Application Fetched Successful");
    response.setSuccess(true);

    return new ResponseEntity<JobApplicationResponse>(response, HttpStatus.OK);
  }

  public ResponseEntity<JobApplicationResponse> fetchAllJobApplicationsByEmployer(int employerId) {

    LOG.info("Request received for fetching all job applications by employer id");

    JobApplicationResponse response = new JobApplicationResponse();

    if (employerId == 0) {
      response.setResponseMessage("employer not found");
      response.setSuccess(false);

      return new ResponseEntity<JobApplicationResponse>(response, HttpStatus.BAD_REQUEST);
    }

    User employer = this.userService.getUserById(employerId);

    if (employer == null) {
      response.setResponseMessage("employer not found");
      response.setSuccess(false);

      return new ResponseEntity<JobApplicationResponse>(response, HttpStatus.BAD_REQUEST);
    }

    List<JobApplication> applications = new ArrayList<>();
    applications = this.jobApplicationService.getByEmployeeAndStatus(employer,
        Arrays.asList(JobApplicationStatus.APPLIED.value(), JobApplicationStatus.REJECTED.value(),
            JobApplicationStatus.SHORTLISTED.value()));

    if (CollectionUtils.isEmpty(applications)) {
      response.setResponseMessage("No Job Applications not found!!");
      response.setSuccess(false);

      return new ResponseEntity<JobApplicationResponse>(response, HttpStatus.OK);
    }

    response.setApplications(applications);
    response.setResponseMessage("Job Application Fetched Successful");
    response.setSuccess(true);

    return new ResponseEntity<JobApplicationResponse>(response, HttpStatus.OK);
  }

  public ResponseEntity<JobApplicationResponse> fetchAllJobApplicationsByEmployerAndStatus(int employerId,
      String status) {

    LOG.info("Request received for fetching all job applications by employer id and status");

    JobApplicationResponse response = new JobApplicationResponse();

    if (employerId == 0 || status == null) {
      response.setResponseMessage("employer not found");
      response.setSuccess(false);

      return new ResponseEntity<JobApplicationResponse>(response, HttpStatus.BAD_REQUEST);
    }

    User employer = this.userService.getUserById(employerId);

    if (employer == null) {
      response.setResponseMessage("employer not found");
      response.setSuccess(false);

      return new ResponseEntity<JobApplicationResponse>(response, HttpStatus.BAD_REQUEST);
    }

    List<JobApplication> applications = new ArrayList<>();

    applications = this.jobApplicationService.getByEmployeeAndStatus(employer, Arrays.asList(status));

    if (CollectionUtils.isEmpty(applications)) {
      response.setResponseMessage("No Job Applications not found!!");
      response.setSuccess(false);

      return new ResponseEntity<JobApplicationResponse>(response, HttpStatus.OK);
    }

    response.setApplications(applications);
    response.setResponseMessage("Job Application Fetched Successful");
    response.setSuccess(true);

    return new ResponseEntity<JobApplicationResponse>(response, HttpStatus.OK);
  }

  public ResponseEntity<CommonApiResponse> updateJobApplicationStatus(JobApplication request) {

    LOG.info("Request received for updating the job application");

    CommonApiResponse response = new CommonApiResponse();

    if (request == null || request.getId() == 0 || request.getStatus() == null) {
      response.setResponseMessage("missing input");
      response.setSuccess(false);

      return new ResponseEntity<CommonApiResponse>(response, HttpStatus.BAD_REQUEST);
    }

    JobApplication jobApplication = this.jobApplicationService.getById(request.getId());

    if (jobApplication == null) {
      response.setResponseMessage("job application not found");
      response.setSuccess(false);

      return new ResponseEntity<CommonApiResponse>(response, HttpStatus.BAD_REQUEST);
    }

    jobApplication.setStatus(request.getStatus());

    JobApplication savedApplication = this.jobApplicationService.update(jobApplication);

    if (savedApplication == null) {
      throw new JobApplicationSaveException("Failed to update the job application status");
    }

    response.setResponseMessage("Job Application Status Updated Successful!!!");
    response.setSuccess(true);

    return new ResponseEntity<CommonApiResponse>(response, HttpStatus.OK);

  }
```

```
public ResponseEntity<JobApplicationResponse> fetchAllJobApplicationsByJob(int jobId) {

  LOG.info("Request received for fetching all job applications by job id");

  JobApplicationResponse response = new JobApplicationResponse();

  if (jobId == 0) {
    response.setResponseMessage("job not found");
    response.setSuccess(false);

    return new ResponseEntity<JobApplicationResponse>(response, HttpStatus.BAD_REQUEST);
  }

  Job job = this.jobService.getById(jobId);

  if (job == null) {
    response.setResponseMessage("job not found");
    response.setSuccess(false);

    return new ResponseEntity<JobApplicationResponse>(response, HttpStatus.BAD_REQUEST);
  }

  List<JobApplication> applications = new ArrayList<>();

  applications = this.jobApplicationService.getByJob(job);

  if (CollectionUtils.isEmpty(applications)) {
    response.setResponseMessage("No Job Applications not found!!");
    response.setSuccess(false);

    return new ResponseEntity<JobApplicationResponse>(response, HttpStatus.OK);
  }
```

BIBLIOGRAPHY

- **Lim, Greg. *Beginning React* – A beginner-friendly guide to mastering React fundamentals.**
  (ISBN: 978-1793843812)
- **Walls, Craig. *Spring in Action* – A comprehensive book for building enterprise applications using Spring Boot.**
  (Manning Publications)
- **Rajput, Dinesh. *Mastering Spring Boot 3.0* – Covers REST API, Spring Security, and real-world implementations.**
  (BPB Publications)
- **Robson, Elisabeth, and Freeman, Eric. *Head First HTML and CSS* – For understanding frontend basics.**
  (O'Reilly Media)