A Project Report

On

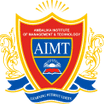
**IndiaLink: A** **Job Searching and Sharing Platform**

Submitted for partial fulfillment of award of

**Bachelor of Technology**

**In**

**Information Technology**



Submitted By:

**PRAJJVAL VERMA** (1903630130015)

**SATYAM SINGH** (1903630130022)

**ADARSH KUMAR AWASTHI** (1903630130001)

Under the guidance

Of

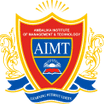
**MR. VIPIN RAWAT** (Assistant Professor)

**Ambalika Institute of Management and Technology, Lucknow**

**Affiliated To**

**Dr. A.P.J. Abdul Kalam Technical University, Lucknow, India**

**2022-2023**



Department of Information Technology

**DECLARATION**

We hereby declare that the project “Job Searching and Sharing Platform” submitted by us in the partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Information Technology of DR. **A.P.J. ABDUL KALAM TECHNICAL UNIVERSITY** **LUCKNOW,** is a record of our own work carried under the supervision and guidance of **Mr. Vipin Rawat**.

To the best of my knowledge the project has not been submitted to **Dr. A.P.J.** **ABDUL KALAM TECHNICAL UNIVERSITY, LUCKNOW** or any other University or institute for the award of the degree.

**PRAJJVAL VERMA (1903630130015)**

**SATYAM SINGH (1903630130022)**

**ADARSH KUMAR AWASTHI (1903630130001)**

### **CERTIFICATE**

Certified that **Prajjval Verma (1903630130015),** **Satyam Singh (1903630130022), Adarsh Kumar Awasthi (1903630130001)** has carried out the project work presented in this project report entitled **“India-Link (Job Searching and Sharing Platform)”** for the award of **Bachelor of Technology** (Information Technology) from **Ambalika Institute of Management & Technology, Lucknow** (Affiliated to Dr. A.P.J Abdul Kalam Technical University, Lucknow) under my guidance. The project report embodies results of original work, and studies are carried out by the student himself (print only that is applicable) and the contents of the project report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

**Mr. Vipin Rawat**  **Dr. Hemlata Pant**

(Assistant Professor) (Head Of Department)

Information Technology Information Technology

**ACKNOWLEDGEMENT**

We hereby express our sincere gratitude to the Management of **Ambalika Institute of Management and Technology, Lucknow,** for their kind encouragement bestowed upon us to complete this project. We take this opportunity to thank **Dr. Ashutosh Diwedi (Director, AIMT)** of providing us with excellent infrastructure and facilities for the development of this project.

We are greatly indebted to Professor and Head of Department **Dr. Hemlata Pant (HOD IT),** for her motivation and guidance through the course of this project work. Her advice, ideas and constant support has encouraged and helped us to get through in difficult times.

We express our deep gratitude to our Project Coordinator **Dr. Yusuf Pervez (Assistant Professor, CSE)**, for his encouragement and guidance towards the development of this project, for his support, confidence as well as awe-inspiring advice throughout the project. Again, we express our profound gratitude to our Project Guide, **Mr. Vipin Rawat (Assistant Professor, IT),** for his encouragement and guidance towards the development of this project.

We express our sincere and heartfelt gratitude to the Faculty of the Department of Information Technology, AIMT College. We would like to express our gratitude to our Parents for having supported for us in every step-in life. Lastly, we express our gratitude to the Almighty God, and all those who have extended their valuable help and suggestions, cooperation, whether directly or indirectly during the progress of the project and thereby aided the successful completion of the project.

**PREFACE**

In today's rapidly evolving job market, the traditional approach to employment has undergone significant transformations. The concept of a lifelong career within a single company is no longer the norm, as individuals seek greater flexibility, work-life balance, and the ability to explore diverse professional opportunities. In this dynamic landscape, job sharing and searching platforms have emerged as powerful tools to connect job seekers and employers, revolutionizing the way people find and engage in work.

Job sharing, once considered an unconventional arrangement, has gained traction as a flexible and innovative solution for both employees and organizations. It involves two or more individuals sharing the responsibilities and hours of a single full-time role, allowing for a more balanced workload and the opportunity to leverage complementary skills and expertise. Job sharing arrangements offer increased flexibility for employees, enabling them to pursue other interests, such as further education, family responsibilities, or entrepreneurial ventures, while still contributing to the workforce.

Meanwhile, employers’ benefit from job sharing by gaining access to a wider talent pool and the ability to retain valuable employees who might otherwise leave due to personal circumstances or the need for reduced working hours. It promotes diversity and inclusivity in the workplace, as it creates opportunities for individuals with diverse backgrounds, experiences, and perspectives to collaborate and contribute.

**ABSTRACT**

The Job Searching and Sharing Project is a comprehensive platform designed to streamline the process of job searching and foster collaboration among job seekers. In today's rapidly evolving job market, finding suitable employment opportunities can be a daunting task. This project aims to address this challenge by leveraging advanced technology and social networking principles to connect job seekers with relevant job openings and facilitate the exchange of valuable insights and resources.

The platform serves as a centralized hub where job seekers can create personalized profiles, showcasing their skills, experience, and career aspirations. Through a sophisticated algorithm, the system matches individuals with suitable job opportunities based on their profiles and preferences, saving valuable time and effort in the job search process.

One of the distinguishing features of this project is its emphasis on collaboration and knowledge sharing. Job Seekers can engage with each other in a supportive community, exchanging advice, strategies, and resources to enhance their job search effectiveness. The platform also provides tools for networking, mentorship, and professional development, enabling individuals to expand their professional connections and acquire new skills.

Employers and recruiters also benefit from this project as they gain access to a pool of talented individuals actively seeking employment. Through the platform, they can post job openings, search for potential candidates, and engage in direct communication with interested applicants, simplifying the hiring process and ensuring a better match between employers and job seekers.

Overall, the Job Searching and Sharing Project offers a comprehensive solution to the challenges faced by job seekers in the modern job market. By leveraging technology, fostering collaboration, and providing valuable insights, the platform aims to empower individuals to find meaningful employment opportunities while building a supportive community of professionals.

**Table of Contents**

[Declaration: 2](#_tyjcwt)

[Certificate:](#_tyjcwt) 3

[Acknowledgment:](#_tyjcwt) 4

[Preface: 5](#_tyjcwt)

[Abstract: 6](#_tyjcwt)

[**Chapter 1 Introduction** 10](#_Toc134708153)

1.1 [Overview of the job searching and sharing platform project: 10](#_Toc134708154)

1.2 [Purpose of the report: 10](#_Toc134708155)

1.3 [Scope of the project: 10](#_Toc134708156)

1.4 [Objectives of the project: 11](#_Toc134708157)

1.5 [What’s new: 11](#_Toc134708158)

[**Chapter 2 Background and literature Review** 12](#_Toc134708159)

2.1 [Evolution of Online Job Searching: 12](#_Toc134708160)

2.2 [Role of Technology: 12](#_Toc134708161)

2.3 [Social Networking and Professional Platforms: 13](#_Toc134708161)

2.4 [Collaborative Job Searching: 13](#_Toc134708162)

2.5 [Remote Work and Gig Economy: 13](#_Toc134708163)

2.6 [Challenges and Opportunities: 13](#_Toc134708164)

2.7 [Analysis of existing job searching and sharing platforms: 14](#_Toc134708165)

2.7.1 [User Interface and Experience: 14](#_Toc134708166)

2.7.2 [Job Matching Algorithms: 14](#_Toc134708167)

2.7.3 [Availability of Comprehensive Job Listings: 14](#_Toc134708168)

2.7.4 [Networking and Community Features: 14](#_Toc134708169)

2.7.5 [Mobile Accessibility: 14](#_Toc134708170)

2.7.6 [Data Privacy and Security: 1](#_Toc134708171)4

[**Chapter 3 Methodology** 15](#_Toc134708172)

3.1 [Description of the project methodology: 15](#_Toc134708173)

3.2 [Project Initiation: 15](#_Toc134708174)

3.3 [Market Research: 16](#_Toc134708175)

3.4 [User Experience (UX) Design: 16](#_Toc134708176)

3.5 [Platform Development: 16](#_Toc134708177)

3.6 [Quality Assurance and Testing: 16](#_Toc134708178)

3.7 [Deployment and Launch: 16](#_Toc134708179)

[**Chapter 4 System Design** 17](#_Toc134708180)

4.1 [Use Case Definition: 17](#_Toc134708181)

4.2 [System Architecture: 17](#_Toc134708182)

4.3 [User Interface Design: 17](#_Toc134708183)

4.4 [Job Listings: 18](#_Toc134708184)

4.5 [Application Tracking: 18](#_Toc134708185)

4.6 [Use Case Diagrams: 18](#_Toc134708186)

[**Chapter 5 Implementation** 23](#_Toc134708187)

5.1 [Front-end Development: 23](#_Toc134708188)

5.2 [Back-end Development: 23](#_Toc134708188)

5.3 [Conclusion: 2](#_Toc134708189)3

[**Chapter 6 Testing** 24](#_Toc134708190)

6.1 [Unit Testing: 24](#_Toc134708191)

6.2 [Integration Testing: 24](#_Toc134708192)

6.3 [User Acceptance Testing (UAT): 24](#_Toc134708193)

6.4 [Performance Testing: 25](#_Toc134708194)

6.5 [Security Testing: 25](#_Toc134708195)

6.6 [Compatibility Testing: 25](#_Toc134708196)

6.7 [Regression Testing: 25](#_Toc134708197)

[Snapshots 27](#_Toc134708198)

[Appendices: 32](#_Toc134708199)

**Chapter 7** [**Conclusion and Future Work** 123](#_Toc134708200)

7.1 [Conclusion 123](#_Toc134708201)

7.2 [Future Work: 123](#_Toc134708202)

7.2.1 [Enhanced Search Functionality: 123](#_Toc134708203)

7.2.2 Recommendation System [: 123](#_Toc134708204)

[7.2.3 Advanced User Profiles: 123](#_tyjcwt)

[Bibliography: 12](#_tyjcwt)4

**List Of Figures**

[Fig-4.1 - Component Diagram: 18](#_tyjcwt)

[Fig-4.2 - Activity Diagram: 19](#_tyjcwt)

[Fig-4.3-Sequence Diagram: 20](#_tyjcwt)

[Fig-4.4-Use Case Diagram: 21](#_tyjcwt)

[Fig-4.5-Data Flow Diagram:](#_tyjcwt) 22

# **Chapter 1 Introduction**

## 1.1 Overview of the job searching and sharing platform project:

The job searching and sharing platform project is a digital platform designed to help job seekers find job opportunities and connect with potential employers, while also helping employers to find the right candidates for their job vacancies. The platform is typically designed to have features such as job listing, job search, resume upload, and candidate application tracking. The platform may also have social networking features that allow job seekers to connect with other professionals in their industry, and for employers to connect with other businesses in their sector.

## 1.2 Purpose of the report:

The purpose of the report is to provide a comprehensive overview of the job searching and sharing platform project. It aims to provide an understanding of the project's objectives, scope, and key features. The report also highlights the methodology used in developing the platform and the tools and technologies used to implement it. The report provides insights into the potential impact of the platform on the job market and identifies areas for future development.

## 1.3 Scope of the project:

The scope of the job searching and sharing platform project involves the design and development of a web-based platform that connects job seekers with potential employers. The platform is typically designed to have features such as job listing, job search, resume upload, and candidate application tracking. The platform may also have social networking features that allow job seekers to connect with other professionals in their industry, and for employers to connect with other businesses in their sector. The project scope may also include the development of mobile applications for both Android and iOS devices.

## 1.4 Objectives of the project:

The main objectives of the job searching and sharing platform project are to:

- Create a user-friendly and intuitive platform for job seekers and employers to connect

- Provide a platform for employers to post job vacancies and receive applications from job seekers

- Provide job seekers with an easy-to-use interface to search for job opportunities and apply for them

- Provide a platform for job seekers to connect with other professionals in their industry and expand their network

- Provide employers with an effective tool to manage the recruitment process and track the progress of candidate applications

- Provide insights into the potential impact of the platform on the job market and identify areas for future development.

## 1.5 What’s new:

- The Job Searching and Sharing Platform is designed to simplify the job search process and enhance collaboration among job seekers.

- One of the standout features of the platform is its integrated quick video chat functionality, allowing candidates and employers to conduct interviews conveniently and efficiently.

- The quick video chat feature enables job seekers to showcase their communication skills, personality, and professionalism in a virtual face-to-face setting.

- Employers benefit from the feature as it helps them assess candidates' non-verbal cues and overall suitability for the role before proceeding to in-person interviews.

- The platform's quick video chat feature eliminates geographical barriers, allowing candidates and employers to connect and interview regardless of their location.

- Job seekers can schedule and conduct video interviews directly through the platform, saving time and effort associated with arranging physical interviews.

- The quick video chat feature supports real-time interaction, enabling immediate feedback and clarifications between the interviewer and the candidate.

- The platform provides a secure and reliable video chat infrastructure, ensuring privacy and confidentiality during the interview process.

- Job seekers can prepare for video interviews by accessing resources and tips available on the platform, helping them present their best selves during the virtual interview.

- Employers can efficiently manage and review recorded video interviews, enabling them to compare candidates, share interviews with colleagues, and make informed hiring decisions.

Overall, the inclusion of a quick video chat feature within the Job Searching and Sharing Platform enhances the interview process, providing a convenient and effective way for candidates and employers to connect, assess qualifications, and make informed decisions about potential job placements.

# **Chapter 2 Background and literature Review**

The job searching and sharing market is a dynamic and ever-evolving sector that plays a vital role in connecting job seekers with employment opportunities. With advancements in technology and the changing nature of work, the job market has witnessed significant shifts in recent years. This literature review aims to provide insights into the background and key trends within the job searching and sharing market.

## 2.1 Evolution of Online Job Searching:

The emergence of the internet and digital platforms has revolutionized the job searching landscape. Traditional methods, such as newspaper classifieds and physical job boards, have been largely replaced by online job portals, aggregators, and professional networking platforms (Jansen, 2011). Online job searching offers greater convenience, accessibility, and a broader range of job opportunities for both job seekers and employers.

## 2.2 Role of Technology:

Technology has been a driving force in shaping the job searching and sharing market. The use of artificial intelligence (AI), machine learning, and big data analytics has significantly impacted the efficiency and effectiveness of job matching and recommendation systems (Hutto & Gilbert, 2014). These technologies analyze job seekers' skills, preferences, and historical data to provide personalized job recommendations, enhancing the chances of finding suitable employment.

2.3 Social Networking and Professional Platforms:

Social networking platforms have become integral to the job searching and sharing market. Platforms like Linked In, with their emphasis on professional connections and personal branding, have transformed the way individuals search for and share job opportunities (Borra & Reilly, 2014).

These platforms allow job seekers to create comprehensive profiles, network with industry professionals, and showcase their skills and experience to potential employers.

## 2.4 Collaborative Job Searching:

Collaboration within the job searching market has gained prominence in recent years. Research by Kluemper and Rosen (2009) highlights the importance of social support and information sharing among job seekers. Collaborative job searching platforms enable individuals to share job leads, insights, and resources, fostering a sense of community and support. This collaboration enhances the effectiveness and outcomes of job searches.

## 2.5 Remote Work and Gig Economy:

The rise of remote work and the gig economy has also had a significant impact on the job searching and sharing market. With the increasing availability of freelance and remote work opportunities, platforms have emerged to connect job seekers with such flexible arrangements (Huws et al., 2017). These platforms provide a space for employers to post gig-based projects, and for job seekers to find short-term or project-based work.

## 2.6 Challenges and Opportunities:

While the job searching and sharing market offers numerous opportunities, it also poses challenges. The abundance of online job listings can lead to information overload and difficulty in identifying the most relevant opportunities (Duan et al., 2013). Additionally, concerns regarding data privacy, algorithmic biases, and platform monopolies need to be addressed to ensure fairness and transparency (Rosenblat & Stark, 2016).

In conclusion, the job searching and sharing market has evolved significantly with the advent of technology, social networking platforms, and changing work patterns. Collaboration, professionalization, and remote work opportunities have emerged as key trends within the market.

# 2.7 Analysis of existing job searching and sharing platforms:

### 2.7.1 User Interface and Experience:

One crucial aspect of job searching and sharing platforms is the user interface and experience. A well-designed platform with an intuitive interface can significantly enhance user satisfaction and engagement. Platforms that prioritize user experience by offering streamlined navigation, easy-to-use search filters, and clear application processes tend to attract and retain more users.

### 2.7.2 Job Matching Algorithms:

The effectiveness of job searching and sharing platforms relies on their job matching algorithms. These algorithms analyze job seekers' profiles, skills, and preferences, and match them with relevant job opportunities. The accuracy and efficiency of these algorithms greatly impact the overall success of the platform. Platforms that leverage advanced technologies, such as AI and machine learning, to improve their matching algorithms tend to provide better job recommendations and increase the likelihood of successful job placements.

### 2.7.3 Availability of Comprehensive Job Listings:

The availability of comprehensive and up-to-date job listings is a critical factor in the success of a job searching and sharing platform. Platforms that partner with a wide range of employers and continuously update their job listings tend to attract more users. It is essential for platforms to ensure the quality of job postings, including accurate job descriptions, clear requirements, and relevant contact information.

### 2.7.4 Networking and Community Features:

Job searching and sharing platforms that offer networking and community features provide added value to users. Features like professional profiles, the ability to connect with industry professionals, and join relevant groups or forums create opportunities for job seekers to expand their network, gain insights, and receive support. Platforms that foster a sense of community tend to foster engagement and increase user satisfaction.

### 2.7.5 Mobile Accessibility:

The growing use of mobile devices highlights the importance of mobile accessibility for job searching and sharing platforms. Platforms that offer mobile applications or responsive mobile-friendly websites enable users to access job listings, apply for positions, and engage with the platform conveniently on their smartphones or tablets. Mobile accessibility enhances the platform's reach and ensures users can access job opportunities on the go.

### 2.7.6 Data Privacy and Security:

Job searching and sharing platforms handle sensitive personal information. It is crucial for these platforms to prioritize data privacy and security. Platforms that implement robust security measures, provide transparent privacy policies, and adhere to data protection regulations gain user trust and confidence.

# **Chapter 3 Methodology**

## 3.1 Description of the project methodology:

The Job Searching and Sharing Project follows a systematic methodology to ensure the effective development and implementation of the platform. The methodology comprises several key stages that guide the project from initiation to completion. Here is a description of the methodology:

## 3.2 Project Initiation:

The project begins with a thorough analysis of the requirements and objectives of the job searching and sharing platform. This includes identifying target users, understanding their needs, and defining the desired features and functionalities of the platform. Stakeholder engagement and consultation are crucial during this stage to gather inputs and align expectations.

## 

## 3.3 Market Research:

A comprehensive market research phase is conducted to gain insights into the existing job searching and sharing landscape. This involves studying competitors' platforms, analyzing market trends, and identifying gaps and opportunities. The research findings inform the development of a unique value proposition and help shape the platform's positioning and differentiation strategies.

## 3.4 User Experience (UX) Design:

The UX design phase focuses on creating an intuitive and user-friendly interface for the platform. User personas and user journey mapping techniques are employed to understand user behavior and design optimal workflows. Wire frames and prototypes are developed, allowing stakeholders to visualize the platform's structure and provide feedback before proceeding to the development phase.

### 3.5 Platform Development:

The development stage involves translating the UX design into a functional job searching and sharing platform. Agile development methodologies are often employed to ensure iterative development, allowing for flexibility and adaptation throughout the process. Development teams collaborate to implement the core features of the platform, including job listing management, user profiles, search algorithms, application tracking, and communication channels.

### 3.6 Quality Assurance and Testing:

The quality assurance and testing phase ensures the platform's reliability, performance, and security. Rigorous testing procedures, including functional testing, usability testing, compatibility testing, and security testing, are conducted to identify and resolve any bugs or issues. Feedback from testing is used to make necessary refinements and optimizations to enhance the platform's stability and user experience.

### 

### 3.7 Deployment and Launch:

Once development and testing are completed, the platform is deployed to a production environment. This involves setting up hosting infrastructure, configuring databases, and ensuring system scalability and security. User data migration and integration with external systems, such as email notification services or payment gateways, are also performed. A comprehensive launch plan is developed to introduce the platform to the target audience, leveraging marketing and communication strategies to drive user adoption.

**Chapter 4 System Design**

4.1 RequirementsGathering:

- Identified key features such as user registration, job posting, job search, resume creation and submission, application tracking, messaging, and notifications.

- Explored additional functionalities like a recommendation engine, analytics, and integration with third-party services.

## 4.2 Use Case Definition:

- Defined various use cases, including job searching, job posting, application tracking.

- Created detailed flowcharts and user stories to map out the steps involved in each use case.

- Identified primary actors (job seekers, employers) and their interactions with the platform.

## 4.3 System Architecture:

- Developed a high-level system architecture that consists of front-end interfaces, back-end systems, and third-parties integration.

- Utilized a web-based architecture to ensure platform accessibility across different devices.

## 4.4 User Interface Design:

- Developed an intuitive and user-friendly interface for both job seekers and employers.

- Implemented responsive design to support various screen sizes and devices.

- Incorporated features such as registration and login, profile creation, job search filters, job posting forms, resume submission forms, messaging interface, and notification alerts.

## 4.5 Job Listings:

- Created a module for employers to post job openings with details like job title, description, requirements, location, and application deadline.

- Implemented search functionality for job seekers based on filters like keywords, location, industry, and job type.

- Provided detailed job descriptions, including company information and application instructions.

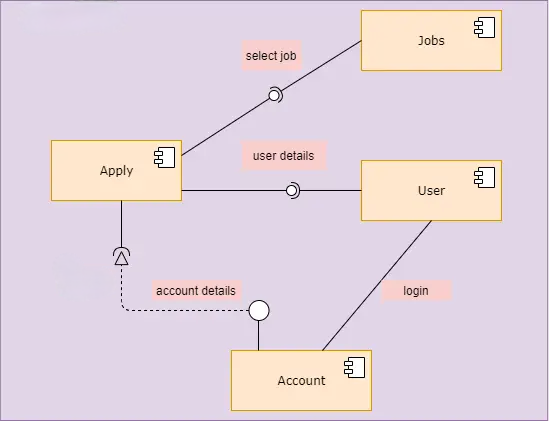
## 4.6 Application Tracking:

- Provided job seekers with an overview of their applications, including status (submitted, in review, rejected, selected), application history, and relevant employer feedback.

- Implemented features for tracking application progress and managing follow-up actions.

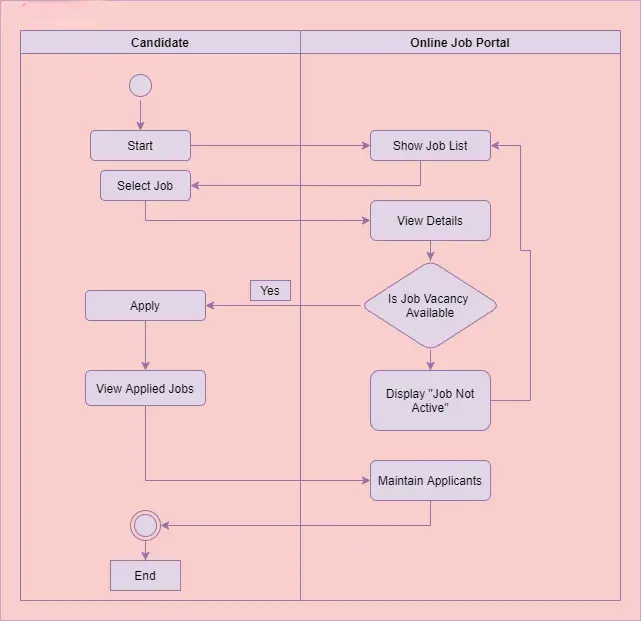
## 

4.7 Use Case Diagrams:

**4.1.**

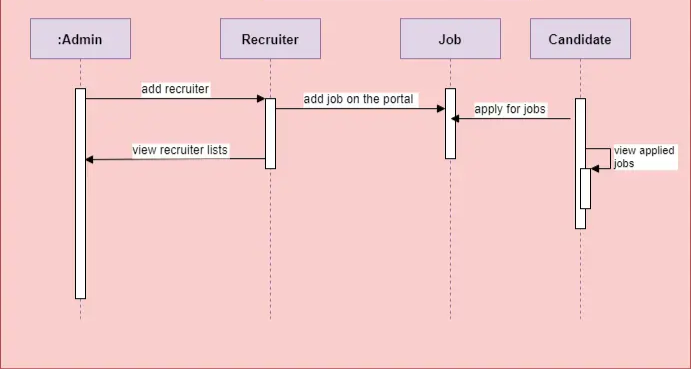
**Figure 4.1 Component Diagram**

**4.2.**

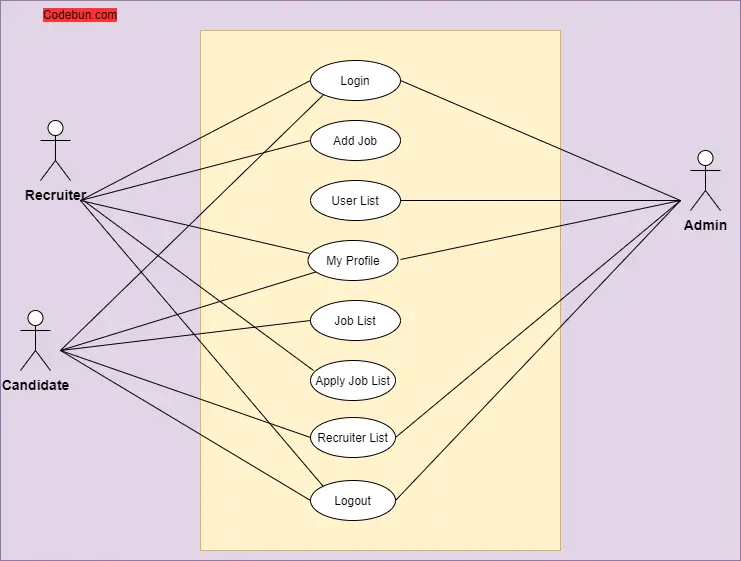


**Figure 4.2 – Activity Diagram**

**4.3.**

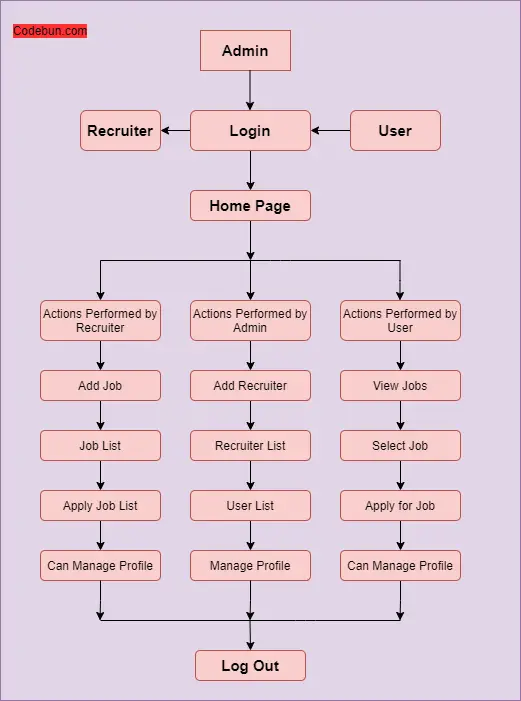


**Figure 4.3 Sequence Diagram**

**4.4.**

**Figure 4.4 Use Case Diagram.**

**4.5**

****

**Figure 4.5-Data Flow Diagram**

# **Chapter 5 Implementation**

5.1 Front-end Development**:**

The front-end development is carried out using React, a popular JavaScript library for building user interfaces. React components are developed to create the different views, including the job search interface, job listing creation form, and user authentication pages. Redux is integrated to manage the application state and facilitate data flow between components.

## 5.2 Back-end Development:

The back-end development primarily involves utilizing Firebase services for user authentication, database management, and real-time updates. Firebase Authentication is implemented to enable user registration, login, and password management. The Firebase Fire-store database is used to store job listings and user profiles. Cloud Functions and Firebase Cloud Messaging are utilized to provide real-time updates and notifications to users.

## 5.3 Conclusion:

The job searching and sharing platform was successfully designed and developed using React, Redux, and Firebase. The platform is user-friendly, fast, efficient, and secure. The platform includes several key features, including job search, job listings, user authentication, user profile, and notifications. The platform is highly scale-able and can be easily expanded to meet the needs of any organization. The

use of modern technologies and best practices ensures that the platform is robust and reliable.

# **Chapter 6 Testing**

Testing for the job searching and sharing platform is crucial to ensure that the platform functions correctly, meets the specified requirements, and provides a smooth user experience. The testing process includes various types of testing, such as unit testing, integration testing, and user acceptance testing. Here's an overview of the testing approach for the job searching and sharing platform:

## 6.1 Unit Testing:

Unit testing involves testing individual components or functions in isolation to verify their behavior. In the context of the platform, unit testing can include testing Redux actions, reducers, and utility functions. Mocking frameworks like Jest can be used to create mock data and simulate different scenarios. The aim is to ensure that each component functions as intended and handles different scenarios correctly.

## 6.2 Integration Testing:

Integration testing focuses on testing the interaction between different components or systems. For the job searching and sharing platform, integration testing can involve testing the integration between React components, Redux store, and Firebase services. This ensures that the components work together seamlessly and the data flow is consistent.

## 6.3 User Acceptance Testing (UAT):

UAT involves testing the platform from the end-user's perspective. It ensures that the platform meets the user's expectations, is user-friendly, and provides the desired functionalities. UAT can be performed by creating test scenarios and involving real users or stakeholders to test the platform. The feedback received during UAT helps identify any usability issues, bugs, or improvements required.

## 6.4 Performance Testing:

Performance testing is important to ensure that the platform can handle a significant load and performs well under different conditions. This includes testing the platform's response time, scalability, and resource usage. Tools like JMeter can be used to simulate high traffic and measure the platform's performance metrics.

## 6.5 Security Testing:

Security testing is crucial to identify and mitigate any vulnerabilities in the platform. It involves testing the platform for potential security risks, such as unauthorized access, data breaches, or injection attacks. Security testing should cover aspects like authentication mechanisms, data encryption, and proper access controls.

## 6.7 Compatibility Testing:

Compatibility testing ensures that the platform works correctly across different devices, browsers, and operating systems. It involves testing the platform on various combinations of devices, browsers, and screen sizes to identify any compatibility issues. Responsive design principles and cross-browser testing tools can be used to ensure a consistent experience.

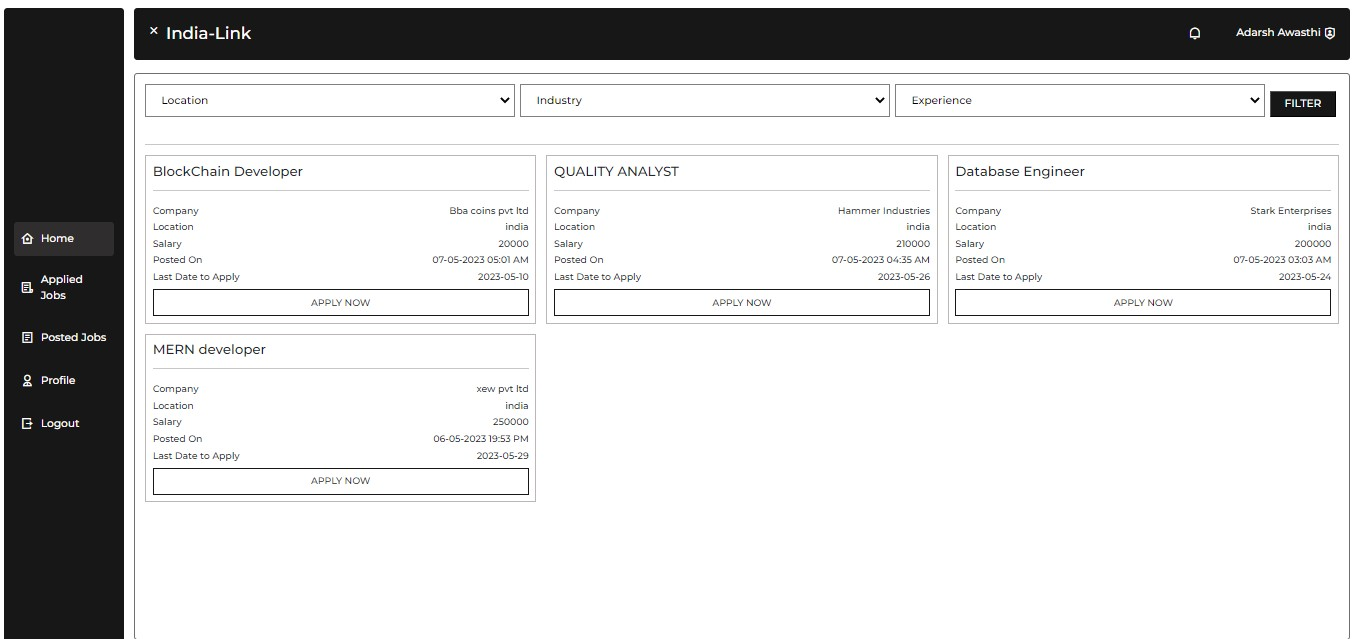
## 6.8 Regression Testing:

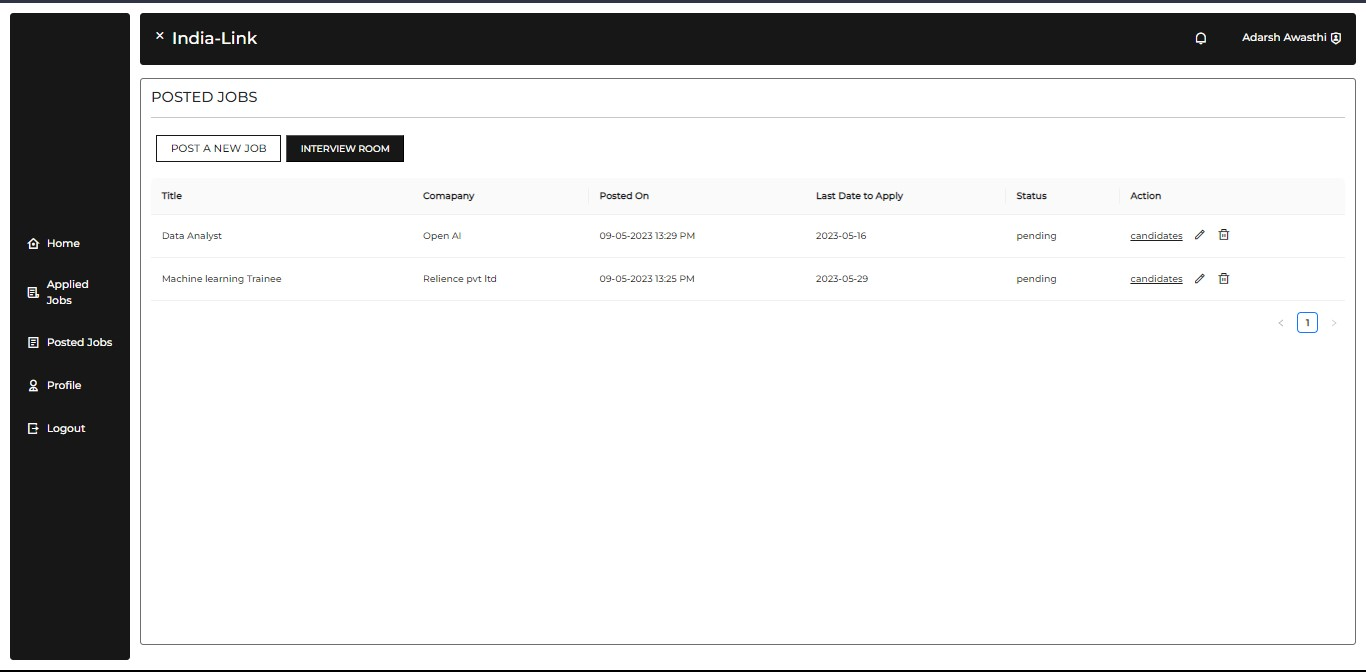
Regression testing is performed after making changes or adding new features to the platform. It ensures that the existing functionalities are not affected by the recent changes. Regression tests cover critical workflows and functionalities to verify that everything continues to work as expected.

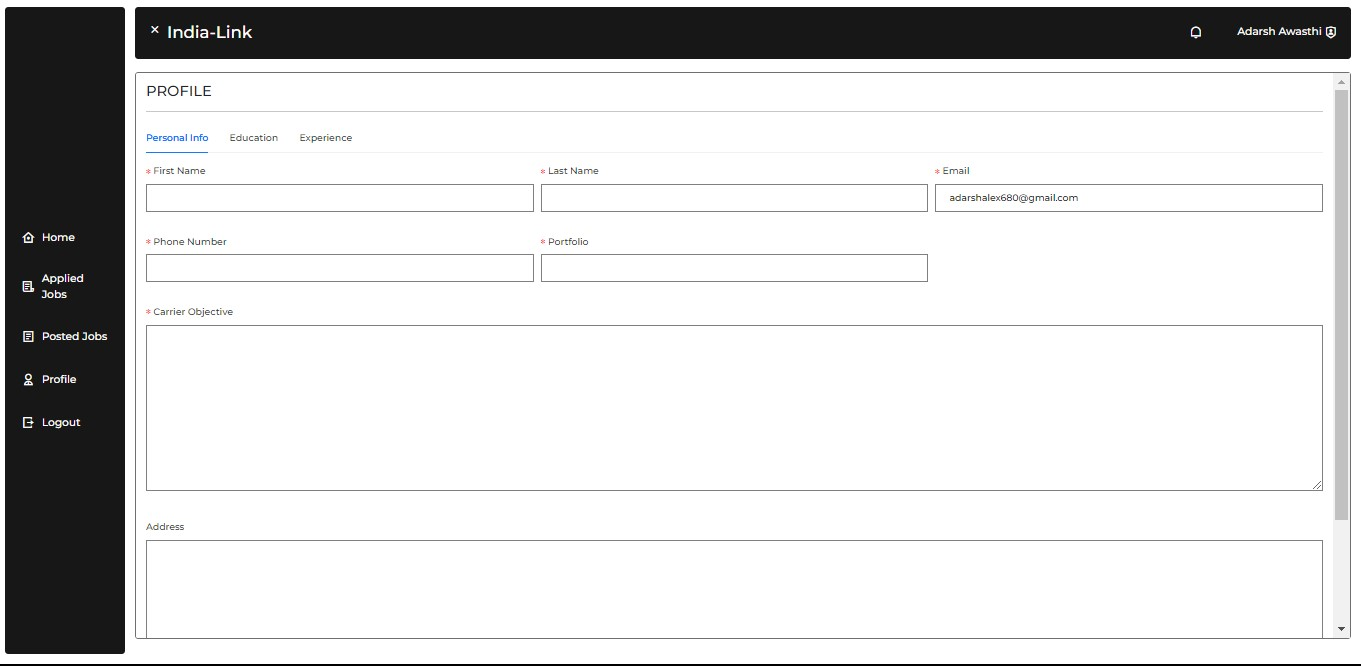
# **Snapshots**

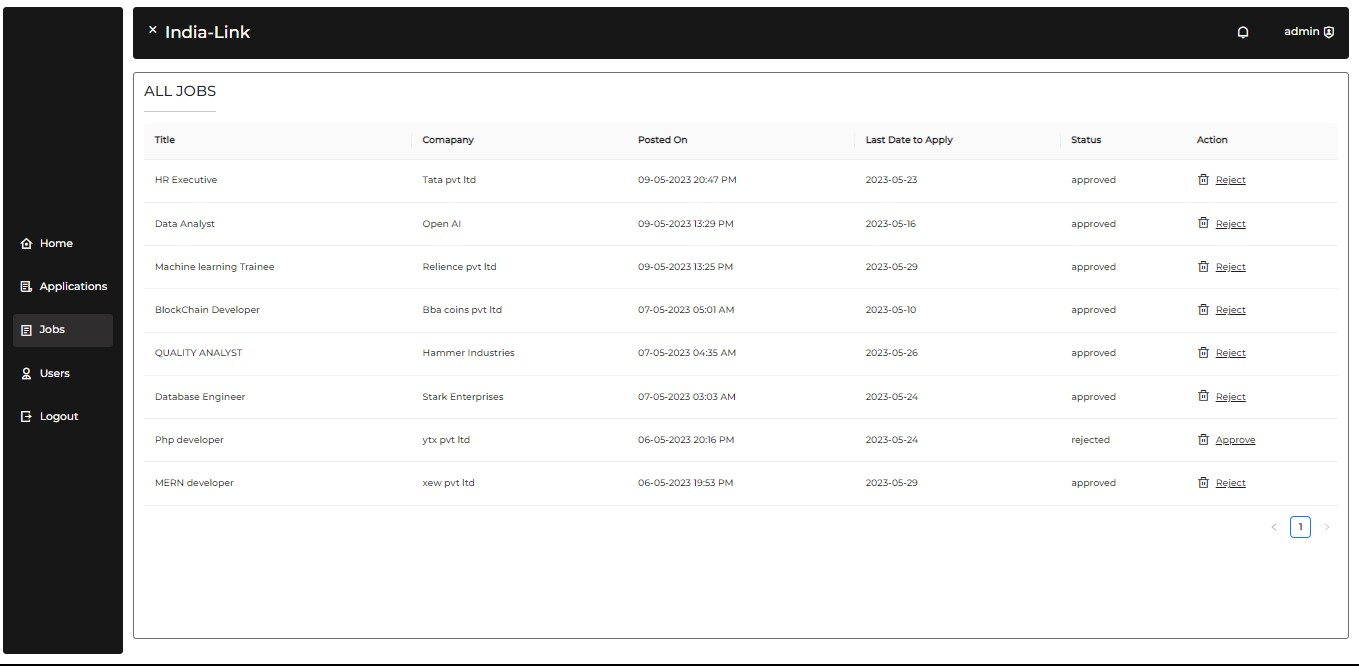
Login Page

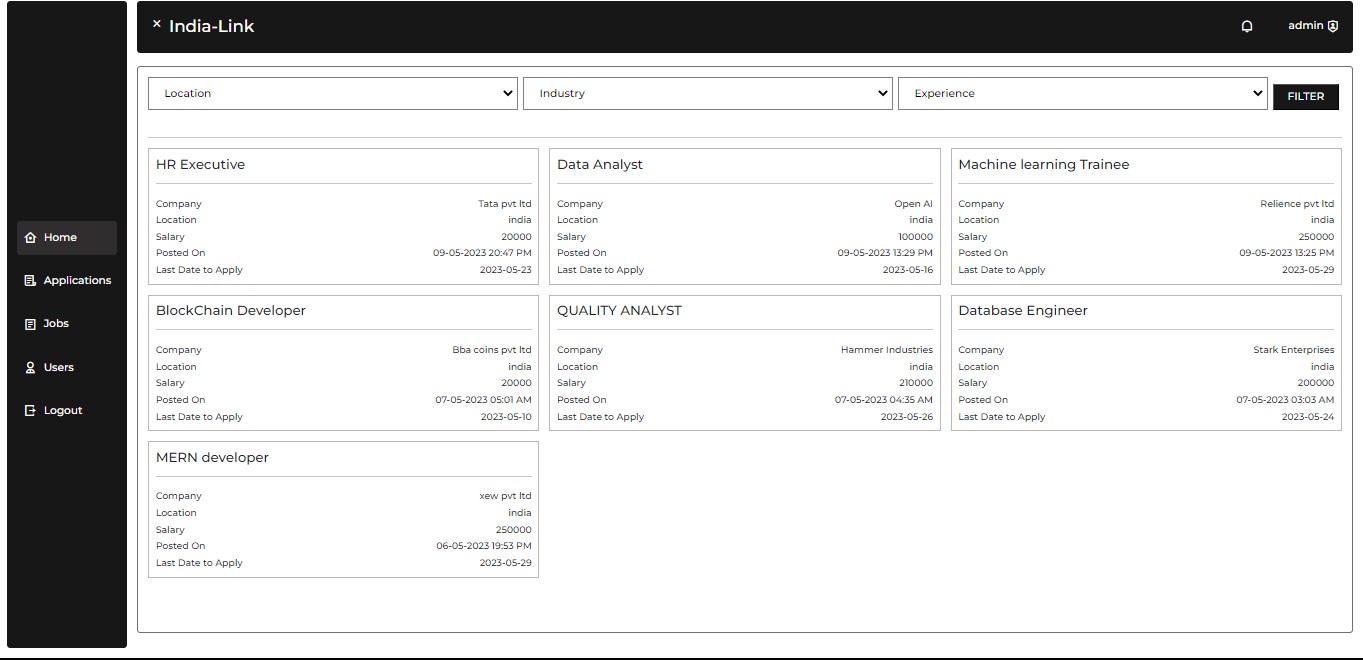
****

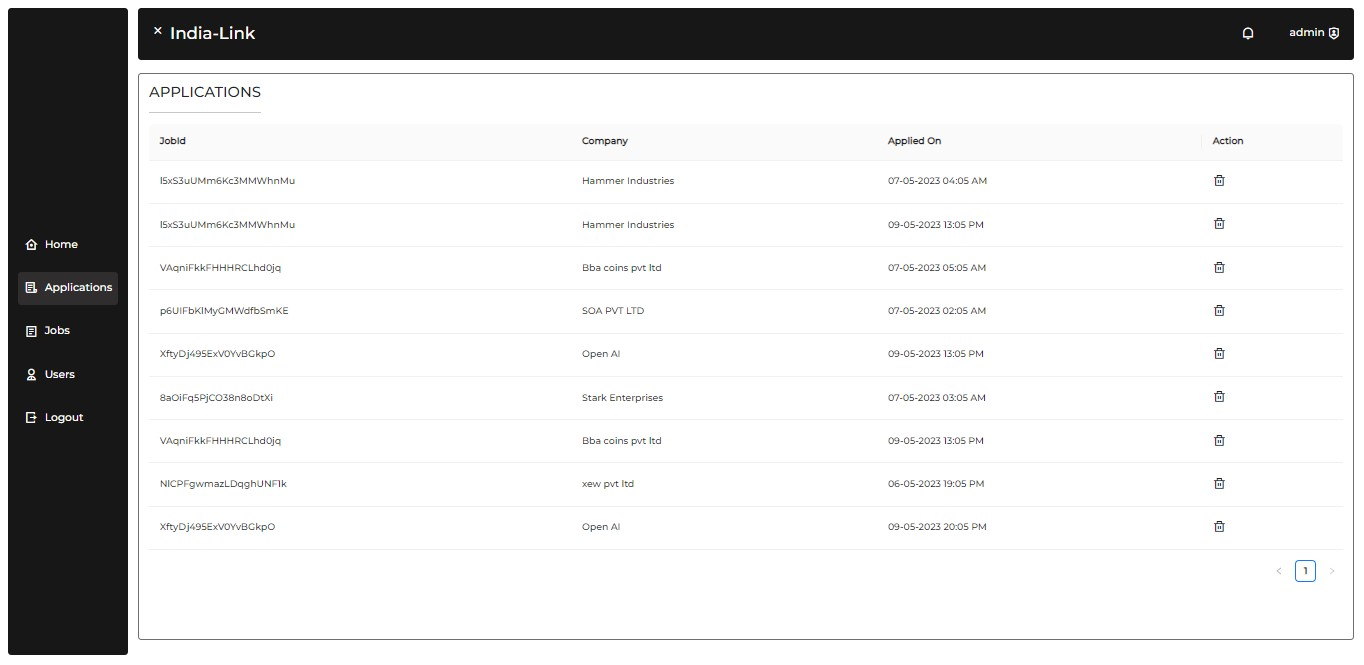
Home-Page

Posted Jobs

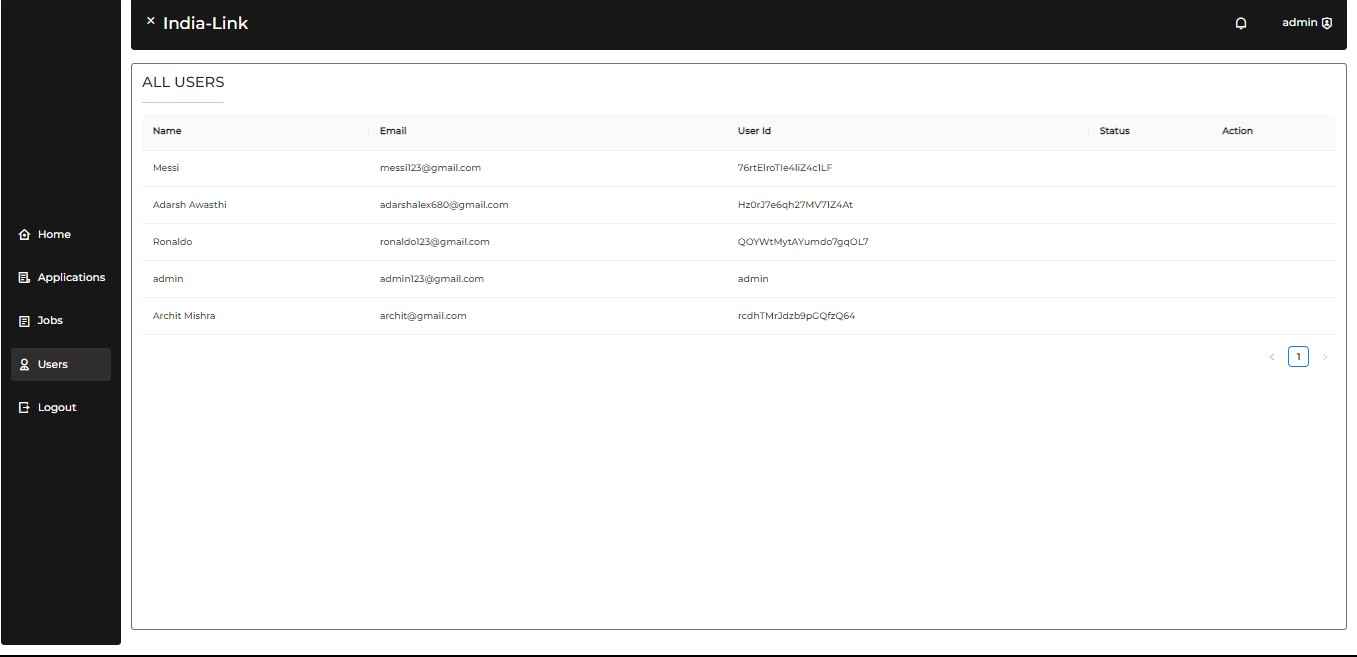
Profile Edit

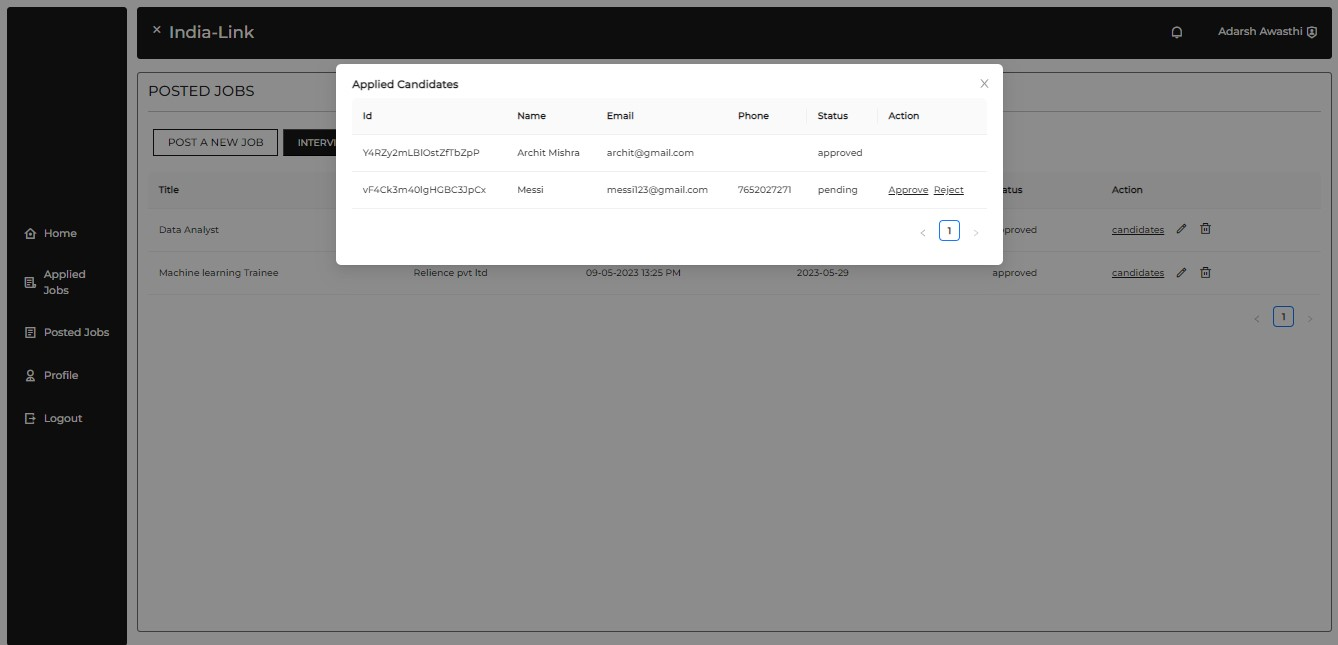
Admin-Job-View

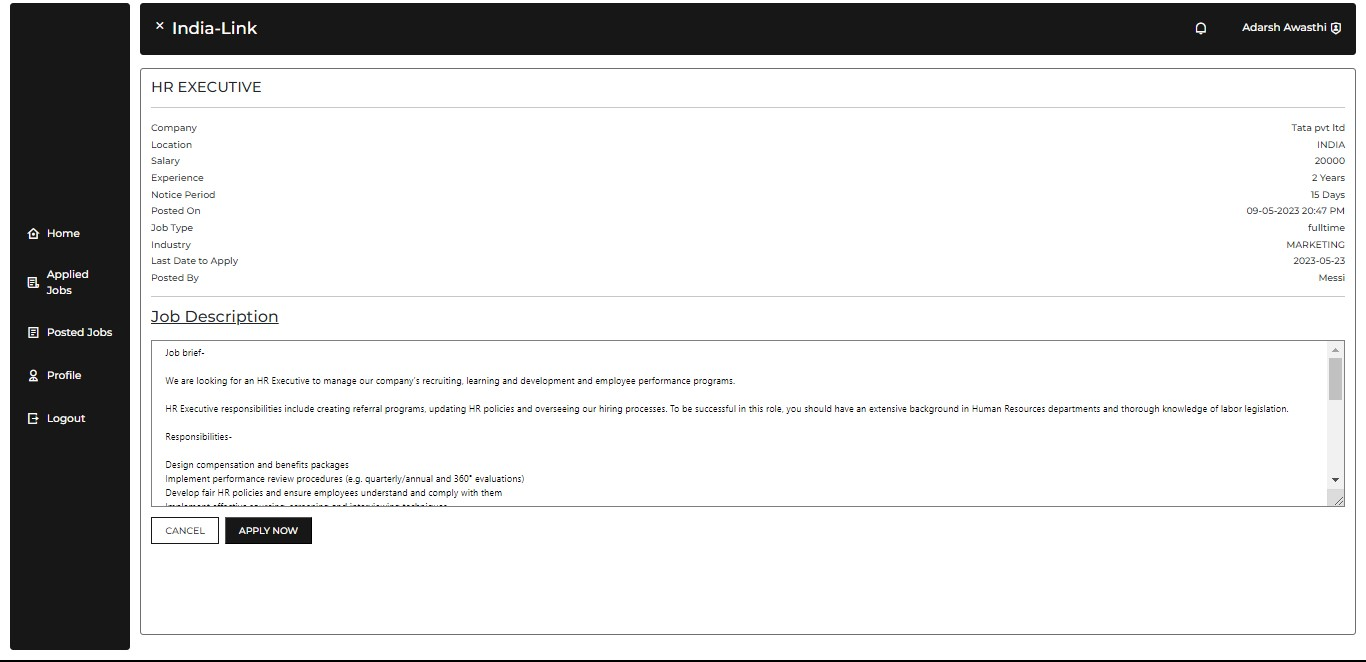
Admin-job-Board

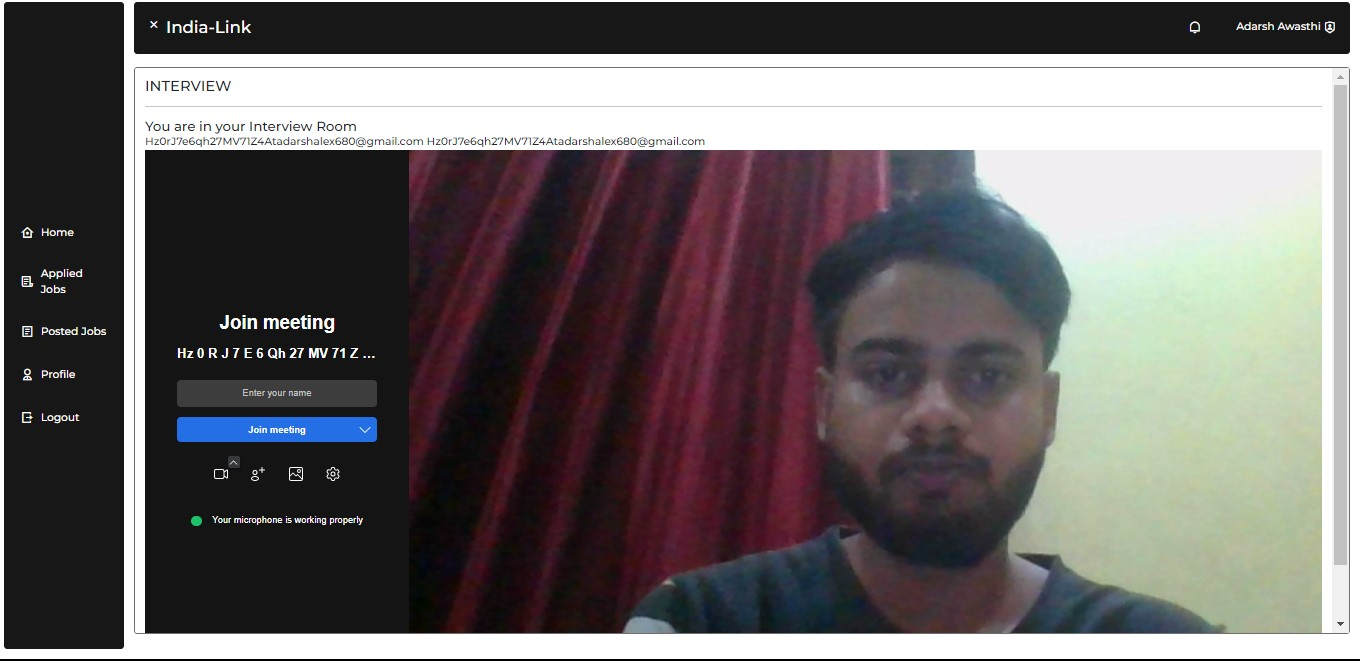
Admin-Applications-View

Admin-Total-User-View



View Candidates

View and Apply Jobs

****

Interview-Dashboard

# **Appendices**:

**index.js**

import React from 'react';

import ReactDOM from 'react-dom/client';

import './index.css';

import App from './App';

import reportWebVitals from './reportWebVitals';

// import 'antd/dist/antd.min.css';

import store from './redux/store.js';

import {Provider} from 'react-redux'

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(

<Provider store = {store} >

<App />

</Provider>

);

// If you want to start measuring performance in your app, pass a function

// to log results (for example: reportWebVitals(console.log))

// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals

reportWebVitals();

**APP.js**

import { BrowserRouter, Routes, Route } from "react-router-dom";

import Login from "./pages/Login.js";

import Register from "./pages/Register.js";

import Home from "./pages/Home.js";

import ProtectedRoute from "./components/ProtectedRoute.js";

import PublicRoute from "./components/PublicRoute.js";

import { useSelector } from "react-redux";

import Loader from "./components/Loader.js";

import "./stylesheets/custom-components.css";

import AppliedJobs from "./pages/user/AppliedJobs.js";

import Profile from "./pages/user/profile/index.js";

import PostedJobs from "./pages/user/postedjobs/index.js";

import NewEditJob from "./pages/user/postedjobs/NewEditJob.js";

import Alljobs from "./pages/admin/Alljobs.js";

import AllUsers from "./pages/admin/AllUsers.js";

import JobDesciption from "./pages/JobDesciption.js";

import Notifications from "./pages/Notifications.js";

import AllApplications from "./pages/admin/AllApplications.js";

import Interview from "./components/Interview.js";

function App() {

const { loading } = useSelector((state) => state.alert);

return (

<div className="">

{loading && <Loader />}

<BrowserRouter>

<Routes>

<Route

path="/"

element={

<ProtectedRoute>

{" "}

<Home />{" "}

</ProtectedRoute>

}

/>

<Route

path="/login"

element={

<PublicRoute>

{" "}

<Login />{" "}

</PublicRoute>

}

/>

<Route

path="/register"

element={

<PublicRoute>

{" "}

<Register />{" "}

</PublicRoute>

}

/>

<Route

path="/applied-jobs"

element={

<ProtectedRoute>

{" "}

<AppliedJobs />{" "}

</ProtectedRoute>

}

/>

<Route

path="/profile/:id"

element={

<ProtectedRoute>

{" "}

<Profile />{" "}

</ProtectedRoute>

}

/>

<Route

path="/notifications"

element={

<ProtectedRoute>

{" "}

<Notifications />{" "}

</ProtectedRoute>

}

/>

<Route

path='/posted-jobs/edit/:id'

element={

<ProtectedRoute>

<NewEditJob/>

</ProtectedRoute>

}

/>

<Route

path='/posted-jobs/new'

element={

<ProtectedRoute>

<NewEditJob />

</ProtectedRoute>

}

/>

<Route

path='/posted-jobs'

element={

<ProtectedRoute>

<PostedJobs />

</ProtectedRoute>

}

/>

<Route

path="/admin/jobs"

element={

<ProtectedRoute>

<Alljobs />

</ProtectedRoute>

}

/>

<Route

path="/admin/users"

element={

<ProtectedRoute>

<AllUsers />

</ProtectedRoute>

}

/>

<Route

path="/admin/applications"

element={

<ProtectedRoute>

<AllApplications />

</ProtectedRoute>

}

/>

<Route

path="/job-description/:id"

element={

<ProtectedRoute>

{" "}

<JobDesciption />{" "}

</ProtectedRoute>

}

/>

<Route

path="/interview/:roomid"

element={

<ProtectedRoute>

{" "}

<Interview />

</ProtectedRoute>

}

/>

</Routes>

</BrowserRouter>

</div>

);

}

export default Filtars;

**Interview.js**

import { JitsiMeeting } from "@jitsi/react-sdk";

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

import { useParams } from "react-router-dom";

import PageTitle from "./PageTitle";

function Interview() {

const id = useParams();

const [isPermissionGranted, setIsPermissionGranted] = useState(false);

const user = JSON.parse(localStorage.getItem("user"));

useEffect(() => {

async function getMedia() {

try {

const mediaStream = await navigator.mediaDevices.getUserMedia({

video: true,

audio: true,

});

setIsPermissionGranted(true);

} catch (error) {

console.error("Error getting media: ", error);

}

}

getMedia();

}, [isPermissionGranted]);

if (!isPermissionGranted) {

return <div>Waiting for permission to access camera and audio...</div>;

}

return (

<>

{id ? (

<div>

<PageTitle title={"Interview"} />

{id.roomid !== `${user.id+''+user.email}` ? <h5>You are in someone's else Interview <small> Having an interview? Good luck</small></h5> : <h5>You are in your Interview Room</h5>}

<div>

{`${user.id+''+user.email} ${id.roomid}`}

<JitsiMeeting

configOverwrite={{

startWithAudioMuted: true,

hiddenPremeetingButtons: ["microphone"],

}}

roomName={`${id.roomid}`}

getIFrameRef={(node) => (node.style.height = "800px")}

/>

</div>

</div>

) : (

<div>No id parameter found</div>

)}

</>

);

}

export default Interview;

**Loader.js**

import React from "react";

function Loader() {

return (

<div className="loader-parent">

<div

className="spinner-grow"

style={{width: '3rem', height: '3rem' , BackgroundColor: 'white'}}

role="status"

>

<span className="visually-hidden">Loading...</span>

</div>

</div>

);

}

export default Loader;

**PageTitle.js**

Oimport React from 'react'

function PageTitle({title}) {

return (

<div className = "page-title">

<h1>{title}</h1>

<hr></hr>

</div>

)

}

export default PageTitle

**ProtectedRoute.js**

import React from 'react'

import DefaultLayout from './DefaultLayout';

function ProtectedRoute({children}) {

const user = JSON.parse(localStorage.getItem('user'))

if(user){

return <DefaultLayout> {children} </DefaultLayout>;

}else{

window.location.href = "/login"

}

}

export default ProtectedRoute

**PublicRoute.js**

import React from 'react'

import {Route, Redirect} from 'react-router-dom'

function PublicRoute({children}) {

const user = JSON.parse(localStorage.getItem('user'))

if(user){

window.location.href = '/'

}else{

return children

}

}

export default PublicRoute

**AllApplication.js**

import React from "react";

import { useDispatch } from "react-redux";

import { Link, useNavigate } from "react-router-dom";

import PageTitle from "../../components/PageTitle";

import { HideLoading, ShowLoading } from "../../redux/alertSlice";

import { getPostedJobsByUserId ,deleteJob, deleteJobById, getApplicationsByUserId, getAllApplications, deleteApplication } from "../../apis/jobs";

import { Table, message } from "antd";

function AllApplications() {

const navigate = useNavigate();

const dispatch = useDispatch();

const [data, setData] = React.useState([]);

const getData = async () => {

try {

dispatch(ShowLoading());

const response = await getAllApplications();

if (response.success) {

setData(response.data);

}

dispatch(HideLoading());

} catch (error) {

dispatch(HideLoading());

message.error(error.message);

}

};

const columns = [

{

title:'JobId',

dataIndex:'jobId',

},

{

title: "Company",

dataIndex: "company",

},

{

title: "Applied On",

dataIndex: "appliedOn",

},

{

title:"Action",

dataIndex:'action',

render:(text,record) =>(

<div className='d-flex gap-3 align-items-center' >

<i class = "ri-delete-bin-line" onClick={async() =>{

try {

dispatch(ShowLoading());

const response = await deleteApplication(record.id);

if (response.success) {

message.success(response.message + `You may wanna reload the page!`);

}

navigate('/admin/applications');

dispatch(HideLoading());

} catch (error) {

dispatch(HideLoading());

message.error(error.message);

}

}

} ></i>

</div>

)

}

];

React.useEffect(() => {

getData();

}, []);

return (

<div>

<div className="d-flex justify-content-between">

<PageTitle title="Applications" />

</div>

<Table columns={columns} dataSource={data} />

</div>

);

}

export default AllApplications;

**Alljobs.js**

import React from "react";

import { useDispatch } from "react-redux";

import { Link, useNavigate } from "react-router-dom";

import PageTitle from "../../components/PageTitle";

import { HideLoading, ShowLoading } from "../../redux/alertSlice";

import {

getPostedJobsByUserId,

deleteJob,

deleteJobById,

editJobDetails,

changeJobStatusFromAdmin,

} from "../../apis/jobs";

import { Table, message } from "antd";

import { getAllJobs } from "../../apis/jobs";

function Alljobs() {

const navigate = useNavigate();

const dispatch = useDispatch();

const [data, setData] = React.useState([]);

const getData = async () => {

try {

dispatch(ShowLoading());

const response = await getAllJobs();

if (response.success) {

setData(response.data);

}

dispatch(HideLoading());

} catch (error) {

dispatch(HideLoading());

message.error(error.message);

}

};

const deleteJob = async (id) => {

try {

dispatch(ShowLoading());

const response = await deleteJobById(id);

if (response.success) {

setData(response.data);

getData();

}

dispatch(HideLoading());

} catch (error) {

dispatch(HideLoading());

message.error(error.message);

}

};

const changeStatus = async (jobData, status) => {

try {

dispatch(ShowLoading());

const response = await changeJobStatusFromAdmin({ ...jobData, status });

if (response.success) {

setData(response.data);

getData();

}

dispatch(HideLoading());

} catch (error) {

dispatch(HideLoading());

message.error(error.message);

}

};

const columns = [

{

title: "Title",

dataIndex: "Title",

},

{

title: "Comapany",

dataIndex: "company",

},

{

title: "Posted On",

dataIndex: "postedOn",

},

{

title: "Last Date to Apply",

dataIndex: "lastDateToApply",

},

{

title: "Status",

dataIndex: "status",

},

{

title: "Action",

dataIndex: "action",

render: (text, record) => (

<div className="d-flex gap-2 align-items-center">

<i

class="ri-delete-bin-line"

onClick={() => deleteJob(record.id)}

></i>

{record.status === "approved" && (

<span

className="underline"

onClick={() => changeStatus(record, "rejected")}

>

Reject

</span>

)}

{(record.status === "pending" || record.status === "rejected") && (

<span

className="underline"

onClick={() => changeStatus(record, "approved")}

>

{" "}

Approve{" "}

</span>

)}

</div>

),

},

];

React.useEffect(() => {

getData();

}, []);

return (

<div>

<div className="d-flex justify-content-between">

<PageTitle title="All Jobs" />

<Link to="/posted-jobs/new">

{" "}

<button className="primary-outlined-btn">New Job</button>

</Link>

</div>

<Table columns={columns} dataSource={data} />

</div>

);

}

export default Alljobs;

**Allusers.js**

import React from "react";

import { useDispatch } from "react-redux";

import { Link, useNavigate } from "react-router-dom";

import PageTitle from "../../components/PageTitle";

import { HideLoading, ShowLoading } from "../../redux/alertSlice";

import {

getPostedJobsByUserId,

deleteJob,

deleteJobById,

editJobDetails,

} from "../../apis/jobs";

import { Table, message } from "antd";

import { getAllUsers, updateUserProfile } from "../../apis/users";

function AllUsers() {

const navigate = useNavigate();

const dispatch = useDispatch();

const [data, setData] = React.useState([]);

const getData = async () => {

try {

dispatch(ShowLoading());

const response = await getAllUsers();

if (response.success) {

setData(response.data);

}

dispatch(HideLoading());

} catch (error) {

dispatch(HideLoading());

message.error(error.message);

}

};

const changeStatus = async (id,status) => {

try{

dispatch(ShowLoading());

const response = await updateUserProfile({id,status});

if(response.success){

setData(response.data);

getData();

}

dispatch(HideLoading());

}catch(error){

dispatch(HideLoading());

message.error(error.message);

}

}

const columns = [

{

title: "Name",

dataIndex: "Name",

},

{

title: "Email",

dataIndex: "email",

},

{

title: "User Id",

dataIndex: "id",

},

{

title:"Status",

dataIndex: "status",

},

{

title: "Action",

dataIndex: "action",

render: (text, record) => (

<div className="d-flex gap-2 align-items-center">

{record.status === "approved" && (

<span

className="underline"

onClick={() => changeStatus(record.id, "rejected")}

>

Reject

</span>

)}

{ (record.status === "pending" || record.status === "rejected") && (

<span

className="underline"

onClick={() => changeStatus(record.id, "approved")}

>

{" "}

Approve{" "}

</span>

)}

</div>

),

},

];

React.useEffect(() => {

getData();

}, []);

return (

<div>

<div className="d-flex justify-content-between">

<PageTitle title="All Users" />

</div>

<Table columns={columns} dataSource={data} />

</div>

);

}

export default AllUsers;

**AppliedCandidates.js**

import { Modal, Table, message } from "antd";

import React from "react";

import { useDispatch } from "react-redux";

import { Link, useNavigate } from "react-router-dom";

import { HideLoading, ShowLoading } from "../../../redux/alertSlice";

import { changeApplicationStatus } from "../../../apis/jobs";

function AppliedCandidates({

showAppliedCandidates,

setShowAppliedCandidates,

appliedCandidates,

reloadData,

}) {

const navigate = useNavigate();

const dispatch = useDispatch();

const user = JSON.parse(localStorage.getItem("user"))

const changeStatus = async (applicationData,status) => {

try {

dispatch(ShowLoading());

const response = await changeApplicationStatus({

...applicationData,

status,

link:`${user.id + '' + user.email}` ,

});

dispatch(HideLoading());

if (response.success) {

message.success(response.message);

reloadData(applicationData.jobId);

} else {

message.error(response.message);

}

} catch (error) {

message.error("Something went wrong");

dispatch(HideLoading());

}

};

const columns = [

{

title: "Id",

dataIndex: "id",

},

{

title: "Name",

dataIndex: "userName",

render: (text, record) => {

return <Link to={`/profile/${record.userId}`}> {text} </Link>;

},

},

{

title: "Email",

dataIndex: "email",

},

{

title: "Phone",

dataIndex: "phoneNumber",

},

{

title: "Status",

dataIndex: "status",

},

{

title: "Action",

dataIndex: "action",

render: (text, record) => {

return (

<div>

{record.status === "pending" && (

<>

{" "}

<span

className="underline"

onClick={() =>{

changeStatus(record, "approved"

}

}

>

Approve

</span>

<span

className="underline mx-2"

onClick={() =>

changeStatus(record, "rejected")

}

>

Reject

</span>

</>

)}

</div>

);

},

},

];

return (

<div>

<Modal

title="Applied Candidates"

open={showAppliedCandidates}

onCancel={() => setShowAppliedCandidates(false)}

footer={null}

width={1000}

>

<Table

columns={columns}

colums={columns}

dataSource={appliedCandidates}

rowkey="id"

/>

</Modal>

</div>

);

}

export default AppliedCandidates;

**index.js**

import React from "react";

import { useDispatch } from "react-redux";

import { Link, useNavigate } from "react-router-dom";

import PageTitle from "../../../components/PageTitle";

import { HideLoading, ShowLoading } from "../../../redux/alertSlice";

import { getPostedJobsByUserId ,deleteJob, deleteJobById, getApplicationsByJobId } from "../../../apis/jobs";

import { Table, message } from "antd";

import AppliedCandidates from "./AppliedCandidates";

function PostedJobs() {

const navigate = useNavigate();

const dispatch = useDispatch();

const [data, setData] = React.useState([]);

const [showAppliedCandidates,setShowAppliedCandidates] = React.useState(false)

const [appliedCandidates,setAppliedCandidates] = React.useState(false)

const InterviewRoom = JSON.parse(localStorage.getItem('user'))

const getData = async () => {

try {

dispatch(ShowLoading());

const user = JSON.parse(localStorage.getItem("user"));

const response = await getPostedJobsByUserId(user.id);

if (response.success) {

setData(response.data);

}

dispatch(HideLoading());

} catch (error) {

dispatch(HideLoading());

message.error(error.message);

}

};

const deleteJob = async (id) => {

try {

dispatch(ShowLoading());

const response = await deleteJobById(id);

if (response.success) {

setData(response.data);

getData()

}

dispatch(HideLoading());

} catch (error) {

dispatch(HideLoading());

message.error(error.message);

}

};

const getAppliedCandidates= async(id) =>{

try{

dispatch(ShowLoading());

const response = await getApplicationsByJobId(id);

if(response.success) {

setAppliedCandidates(response.data);

if(!showAppliedCandidates){

setShowAppliedCandidates(true);

}

}

dispatch(HideLoading());

}catch(error){

dispatch(HideLoading());

message.error(error.message);

}

}

const columns = [

{

title: "Title",

dataIndex: "Title",

},

{

title: "Comapany",

dataIndex: "company",

},

{

title: "Posted On",

dataIndex: "postedOn",

},

{

title: "Last Date to Apply",

dataIndex: "lastDateToApply",

},

{

title: "Status",

dataIndex: "status",

},

{

title:"Action",

dataIndex:'action',

render:(text,record) =>(

<div className='d-flex gap-3 align-items-center' >

<span className="underline" onClick={()=> getAppliedCandidates(record.id)}>

candidates

</span>

<i class="ri-pencil-line" onClick={()=> navigate(`/posted-jobs/edit/${record.id}`)} ></i>

<i class = "ri-delete-bin-line" onClick={() => deleteJob(record.id)} ></i>

</div>

)

}

];

React.useEffect(() => {

getData();

}, []);

return (

<div>

<PageTitle title="Posted Jobs" />

<div className="flex justify-content-between">

<Link to="/posted-jobs/new">

{" "}

<button className="primary-outlined-btn mt-2 mx-2">Post A New Job</button>

</Link>

<Link to= {`/Interview/${InterviewRoom.id + '' + InterviewRoom.email}`}>

<button className="primary-contained-btn"> <small>Interview Room</small> </button>

</Link>

</div>

<br></br>

<Table columns={columns} dataSource={data} />

{showAppliedCandidates &&

<AppliedCandidates

showAppliedCandidates={showAppliedCandidates}

setShowAppliedCandidates={setShowAppliedCandidates}

appliedCandidates={appliedCandidates}

reloadData={getAppliedCandidates}

/>}

</div>

);

}

export default PostedJobs;

**NewEditjob.js**

import React, { useEffect } from "react";

import { useNavigate, useParams } from "react-router-dom";

import PageTitle from "../../../components/PageTitle";

import { Form, Row, Col, message } from "antd";

import { useDispatch } from "react-redux";

import { HideLoading, ShowLoading } from "../../../redux/alertSlice";

import { addNewJobPost, editJobDetails, getJobById } from "../../../apis/jobs";

function NewEditJob() {

const params = useParams();

const navigate = useNavigate();

const dispatch = useDispatch();

const [jobData, setJobData] = React.useState(null);

const onFinish = async (values) => {

try {

dispatch(ShowLoading());

let response = null;

if (params.id) {

response = await editJobDetails({

...values,

id:params.id,

})

} else {

response = await addNewJobPost(values);

}

if (response.success) {

message.success(response.message);

navigate("/posted-jobs");

} else {

message.error(response.message);

}

dispatch(HideLoading());

} catch (error) {

dispatch(HideLoading());

message.error(error.message);

}

};

const getData = async () => {

try {

dispatch(ShowLoading());

const response = await getJobById(params.id);

dispatch(HideLoading());

if (response.success) {

setJobData(response.data);

} else {

message.error(response.message);

}

} catch (error) {

message.error(error.message);

}

};

useEffect(() => {

if(params.id){

getData();

}else{

setJobData({});

}

}, []);

return (

<div>

<PageTitle title={params.id ? "Edit Job" : "Add New Job Post"} />

{jobData && (

<Form layout="vertical" onFinish={onFinish} initialValues={jobData}>

<Row gutter={[10, 10]}>

<Col span={12}>

<Form.Item

label="Title"

name="Title"

rules={[{ required: true, message: "required" }]}

>

<input type="text" />

</Form.Item>

</Col>

<Col span={6}>

<Form.Item

label="Industry"

name="industry"

rules={[{ required: true, message: "required" }]}

>

<select name="" id="">

<option value="">Select</option>

<option value="it">IT</option>

<option value="finance">Finance</option>

<option value="marketing">Marketing</option>

<option value="realstate">Real Estate</option>

</select>

</Form.Item>

</Col>

<Col span={6}>

<Form.Item

label="Location"

name="location"

rules={[{ required: true, message: "required" }]}

>

<select name="" id="">

<option value="">Select</option>

<option value="india">India</option>

<option value="usa">USA</option>

<option value="uk">UK</option>

</select>

</Form.Item>

</Col>

<Col span={6}>

<Form.Item

label="Cpmpany Name"

name="company"

rules={[{ required: true, message: "required" }]}

>

<input type="text" />

</Form.Item>

</Col>

<Col span={6}>

<Form.Item

label="Salary"

name="salary"

rules={[{ required: true, message: "required" }]}

>

<input type="text" />

</Form.Item>

</Col>

<Col span={6}>

<Form.Item

label="Job Type"

name="jobType"

rules={[{ required: true, message: "required" }]}

>

<select name="" id="">

<option value="">Select</option>

<option value="fulltime">Full Time</option>

<option value="parttime">Part Time</option>

<option value="contract">Contract</option>

</select>

</Form.Item>

</Col>

<Col span={6}>

<Form.Item

label="Last Date To Apply"

name="lastDateToApply"

rules={[{ required: true, message: "required" }]}

>

<input type="date" />

</Form.Item>

</Col>

<Col span={6}>

<Form.Item

label="Experience"

name="experience"

rules={[{ required: true, message: "required" }]}

>

<input type="number" />

</Form.Item>

</Col>

<Col span={6}>

<Form.Item

label="Notice Period"

name="noticePeriod"

rules={[{ required: true, message: "required" }]}

>

<input type="text" />

</Form.Item>

</Col>

<Col span={24}>

<Form.Item

label="Job Description"

name="jobDescription"

rules={[{ required: true, message: "required" }]}

>

<textarea type="text" />

</Form.Item>

</Col>

</Row>

<div className="d-flex justify-content-end gap-2">

<button

className="primary-outlined-btn"

onClick={() => navigate("/posted-jobs")}

>

{" "}

Cancel

</button>

<button

className="primary-contained-btn"

type="submit"

>

Save

</button>

</div>

</Form>

)}

</div>

);

}

export default NewEditJob;

**Education.js**

import { Form, Input, Row, Col } from "antd";

import React from "react";

function Education() {

return (

<>

<Form.List name="education">

{(fields, { add, remove }) => (

<>

{fields.map(({ key, name, ...restField }) => (

<Row gutter={[10, 10]} align="middle">

<Col span={8}>

<Form.Item

{...restField}

name={[name, "institution"]}

rules={[

{

required: true,

message: "required",

},

]}

label="Institution"

>

<Input placeholder="First Name" />

</Form.Item>

</Col>

<Col span={8}>

<Form.Item

{...restField}

name={[name, "degree"]}

rules={[

{

required: true,

message: "required",

},

]}

label="Degree"

>

<Input placeholder="" />

</Form.Item>

</Col>

<Col span={4}>

<Form.Item

{...restField}

name={[name, "percentage"]}

rules={[

{

required: true,

message: "required",

},

]}

label="Percentage"

>

<Input placeholder="" />

</Form.Item>

</Col>

<i class="ri-delete-bin-line" onClick={() => remove(name)}></i>

</Row>

))}

<Form.Item>

<button className="primary-outlined-btn " onClick={() => add()}>

ADD EDUCATION

</button>

</Form.Item>

</>

)}

</Form.List>

<Form.List name="skills">

{(fields, { add, remove }) => (

<>

{fields.map(({ key, name, ...restField }) => (

<Row gutter={[10, 10]} align="middle">

<Col span={8}>

<Form.Item

{...restField}

name={[name, "technology"]}

rules={[

{

required: true,

message: "required",

},

]}

label="Technology"

>

<Input placeholder="" />

</Form.Item>

</Col>

<Col span={8}>

<Form.Item

{...restField}

name={[name, "rating"]}

rules={[

{

required: true,

message: "required",

},

]}

label="Rating"

>

<Input placeholder="" />

</Form.Item>

</Col>

<i class="ri-delete-bin-line" onClick={() => remove(name)}></i>

</Row>

))}

<Form.Item>

<button className="primary-outlined-btn " onClick={() => add()}>

ADD SKILLS

</button>

</Form.Item>

</>

)}

</Form.List>

</>

);

}

export default Education;

/\*

<Form.List name="education">

{(fields, { add, remove }) => (

<>

{fields.map(({ key, name, ...restField }) => (

<Row gutter={[10, 10]} align="middle">

<Col span={8}>

<Form.Item

{...restField}

name={[name, "institution"]}

rules={[

{

required: true,

message: "required",

},

]}

label="Institution"

>

<Input placeholder="First Name" />

</Form.Item>

</Col>

<Col span={8}>

<Form.Item

{...restField}

name={[name, "degree"]}

rules={[

{

required: true,

message: "required",

},

]}

label="Degree"

>

<Input placeholder="First Name" />

</Form.Item>

</Col>

<Col span={4}>

<Form.Item

{...restField}

name={[name, "percentage"]}

rules={[

{

required: true,

message: "required",

},

]}

label="Percentage"

>

<Input placeholder="First Name" />

</Form.Item>

</Col>

<i class="ri-delete-bin-line" onClick={() => remove(name)}></i>

</Row>

))}

<Form.Item>

<button className="primary-outlined-btn " onClick={() => add()}>

ADD EDUCATION

</button>

</Form.Item>

</>

)}

</Form.List>

\*/

**Experience.js**

import { Form, Input, Row, Col } from "antd";

import React from "react";

function Experience() {

return (

<>

<Form.List name="experiences">

{(fields, { add, remove }) => (

<>

{fields.map(({ key, name, ...restField }) => (

<Row gutter={[10, 10]} align="middle">

<Col span={7}>

<Form.Item

{...restField}

name={[name, "company"]}

rules={[

{

required: true,

message: "required",

},

]}

label="Company"

>

<Input placeholder="First Name" />

</Form.Item>

</Col>

<Col span={7}>

<Form.Item

{...restField}

name={[name, "designation"]}

rules={[

{

required: true,

message: "required",

},

]}

label="Designation"

>

<Input placeholder="" />

</Form.Item>

</Col>

<Col span={4}>

<Form.Item

{...restField}

name={[name, "duration"]}

rules={[

{

required: true,

message: "required",

},

]}

label="Duration"

>

<Input placeholder="" />

</Form.Item>

</Col>

<Col span={4}>

<Form.Item

{...restField}

name={[name, "location"]}

rules={[

{

required: true,

message: "required",

},

]}

label="Location"

>

<Input placeholder="" />

</Form.Item>

</Col>

<i class="ri-delete-bin-line" onClick={() => remove(name)}></i>

</Row>

))}

<Form.Item>

<button className="primary-outlined-btn " onClick={() => add()}>

ADD EXPERIENCE

</button>

</Form.Item>

</>

)}

</Form.List>

<Form.List name="project">

{(fields, { add, remove }) => (

<>

{fields.map(({ key, name, ...restField }) => (

<Row gutter={[10, 10]} align="middle">

<Col span={7}>

<Form.Item

{...restField}

name={[name, "title"]}

rules={[

{

required: true,

message: "required",

},

]}

label="Title"

>

<Input placeholder="" />

</Form.Item>

</Col>

<Col span={11}>

<Form.Item

{...restField}

name={[name, "Description"]}

rules={[

{

required: true,

message: "required",

},

]}

label="Description"

>

<textarea placeholder="" />

</Form.Item>

</Col>

<Col span={4}>

<Form.Item

{...restField}

name={[name, "duration"]}

rules={[

{

required: true,

message: "required",

},

]}

label="Duration"

>

<Input placeholder="" />

</Form.Item>

</Col>

<i class="ri-delete-bin-line" onClick={() => remove(name)}></i>

</Row>

))}

<Form.Item>

<button className="primary-outlined-btn " onClick={() => add()}>

ADD PROJECT

</button>

</Form.Item>

</>

)}

</Form.List>

</>

)

}

export default Experience

**index.js**

import React from "react";

import PageTitle from "../../../components/PageTitle";

import { Tabs ,message} from "antd";

import PersonalInfo from "./PersonalInfo";

import Education from "./Education";

import Experience from "./Experience";

import { Form } from "antd";

import { useDispatch } from "react-redux";

import {getUserProfile, updateUserProfile} from '../../../apis/users'

import { HideLoading, ShowLoading } from "../../../redux/alertSlice";

import { useNavigate, useParams } from "react-router-dom";

const { TabPane } = Tabs;

function Profile() {

const dispatch = useDispatch();

const navigate = useNavigate();

const [userData , setUserData] = React.useState(null);

const params = useParams();

const loggedInUser = JSON.parse(localStorage.getItem('user'));

const onFinish= async(values) =>{

try{

dispatch(ShowLoading());

const response = await updateUserProfile(values);

dispatch(HideLoading());

if(response.success){

message.success(response.message);

}else{

message.error(response.message);

}

}catch(error){

dispatch(HideLoading());

message.error(error.message);

}

}

const getData = async() =>{

try{

dispatch(ShowLoading());

const user = JSON.parse(localStorage.getItem('user'));

const response = await getUserProfile(params.id);

dispatch(HideLoading());

if(response.success){

setUserData(response.data);

}else{

message.error(response.message);

}

}catch(error){

message.error(error.message);

}

}

React.useEffect(()=>{

getData();

},[])

return (

<div>

<PageTitle title="Profile" />

{userData && <Form layout="vertical" onFinish={onFinish}

initialValues={userData}

>

<Tabs defaultActiveKey="1">

<TabPane tab="Personal Info" key="1">

{" "}

<PersonalInfo />

</TabPane>

<TabPane tab="Education" key="2">

{" "}

<Education />

</TabPane>

<TabPane tab="Experience" key="3">

{" "}

<Experience />

</TabPane>

</Tabs>

<div className="d-flex justify-content-end">

{params.id === loggedInUser.id && <button className="primary-contained-btn" type="submit">SAVE</button>}

<button className="primary-outlined-btn" type="button" onClick={()=>navigate('/')}>

CANCEL

</button>

</div>

</Form>}

</div>

);

}

export default Profile;

**PersonalInfo.js**

import { Col, Form, Row } from "antd";

import React from "react";

function PersonalInfo() {

return (

<Row gutter={[10, 10]}>

<Col span={8}>

<Form.Item

label="First Name"

name="firstName"

rules={[{ required: true, message: "required" }]}

>

<input type="text" />

</Form.Item>

</Col>

<Col span={8}>

<Form.Item

label="Last Name"

name="lastName"

rules={[{ required: true, message: "required" }]}

>

<input type="text" />

</Form.Item>

</Col>

<Col span={8}>

<Form.Item

label="Email"

name="email"

rules={[{ required: true, message: "required" }]}

>

<input type="text" />

</Form.Item>

</Col>

<Col span={8}>

<Form.Item

label="Phone Number"

name="phoneNumber"

rules={[{ required: true, message: "required" }]}

>

<input type="text" />

</Form.Item>

</Col>

<Col span={8}>

<Form.Item

label="Portfolio"

name="portfolio"

rules={[{ required: true, message: "required" }]}

>

<input type="text" />

</Form.Item>

</Col>

<Col span={24}>

<Form.Item

label="Carrier Objective"

name="carrierObjective"

rules={[{ required: true, message: "required" }]}

>

<textarea type="text" rows={4} />

</Form.Item>

</Col>

<Col span={24}>

<Form.Item label="Address" name="address">

<textarea type="text" rows={2} />

</Form.Item>

</Col>

</Row>

);

}

export default PersonalInfo;

**ApplliedJobs.js**

import React from "react";

import { useDispatch } from "react-redux";

import { Link, useNavigate } from "react-router-dom";

import PageTitle from "../../components/PageTitle";

import { HideLoading, ShowLoading } from "../../redux/alertSlice";

import { getPostedJobsByUserId ,deleteJob, deleteJobById, getApplicationsByUserId } from "../../apis/jobs";

import { Table, message } from "antd";

function AppliedJobs() {

const navigate = useNavigate();

const dispatch = useDispatch();

const [data, setData] = React.useState([]);

const getData = async () => {

try {

dispatch(ShowLoading());

const user = JSON.parse(localStorage.getItem("user"));

const response = await getApplicationsByUserId(user.id);

if (response.success) {

setData(response.data);

}

dispatch(HideLoading());

} catch (error) {

dispatch(HideLoading());

message.error(error.message);

}

};

const columns = [

{

title: "Job",

dataIndex: "jobTitle",

},

{

title: "Comapany",

dataIndex: "company",

},

{

title: "Applied On",

dataIndex: "appliedOn",

},

{

title: "Status",

dataIndex: "status",

},

];

React.useEffect(() => {

getData();

}, []);

return (

<div>

<div className="d-flex justify-content-between">

<PageTitle title="Applied Jobs" />

</div>

<Table columns={columns} dataSource={data} />

</div>

);

}

export default AppliedJobs;

**Home.js**

import React, { useState } from "react";

import { useDispatch } from "react-redux";

import { useNavigate } from "react-router-dom";

import { HideLoading, ShowLoading } from "../redux/alertSlice";

import { getAllJobs } from "../apis/jobs";

import { Col, Row, message } from "antd";

import PageTitle from "../components/PageTitle";

import Filters from "../components/Filters";

function Home() {

const navigate = useNavigate();

const dispatch = useDispatch();

const [data, setData] = useState([]);

const [filters, setFilters ] = useState({

location:'',

industry:'',

experience:'',

});

const getData = async () => {

try {

dispatch(ShowLoading());

const response = await getAllJobs();

if (response.success) {

const approvedJobs = response.data.filter((job) => job.status === 'approved')

setData(approvedJobs);

}

dispatch(HideLoading());

} catch (error) {

dispatch(HideLoading());

message.error(error.message);

}

};

React.useEffect(() => {

getData();

}, []);

return (

<div>

<Filters

filter={filters}

setFilter={setFilters}

SetData={setData}

/>

<br></br>

<hr></hr>

<Row gutter={[15, 15]}>

{data.map((job) => (

<Col span={8}>

<div className="job-card">

<h5 className="uppercase">{job.Title}</h5>

<hr></hr>

<div className = "d-flex flex-column gap-1">

<div className="d-flex justify-content-between mt-1">

<span>Company</span>

<span>{job.company}</span>

</div>

<div className="d-flex justify-content-between">

<span>Location</span>

<span>{job.location}</span>

</div>

<div className="d-flex justify-content-between">

<span>Salary</span>

<span>{job.salary}</span>

</div>

<div className="d-flex justify-content-between">

<span>Posted On</span>

<span>{job.postedOn}</span>

</div>

<div className="d-flex justify-content-between">

<span>Last Date to Apply</span>

<span>{job.lastDateToApply}</span>

</div>

</div>

<button className="primary-outlined-btn w-100 mt-2"

onClick={()=> navigate(`job-description/${job.id}`)}

>

Apply Now

</button>

</div>

</Col>

))}

</Row>

</div>

);

}

export default Home;

**jobDescription.js**

import React, { useEffect } from "react";

import { useDispatch } from "react-redux";

import { useNavigate, useParams } from "react-router-dom";

import { HideLoading, ShowLoading } from "../redux/alertSlice";

import {

applyJobPost,

getApplicationByJobId,

getApplicationsByJobId,

getJobById,

} from "../apis/jobs";

import { Col, Row, message } from "antd";

import PageTitle from "../components/PageTitle";

function JobDesciption() {

const params = useParams();

const navigate = useNavigate();

const dispatch = useDispatch();

const [jobData, setJobData] = React.useState(null);

const [showApplyButton, setShowApplyButton] = React.useState(true);

const [alreadyApplied,setAlreadyApplied] = React.useState(false);

const user = JSON.parse(localStorage.getItem("user"))

const getData = async () => {

try {

dispatch(ShowLoading());

const response = await getJobById(params.id);

if (

response.data.postedByUserId ===

JSON.parse(localStorage.getItem("user")).id

) {

setShowApplyButton(false);

}

const applicationsResponse = await getApplicationsByJobId(params.id); //(params.id);

if (

applicationsResponse.data.filter(

(item) => item.userId === user.id

).length > 0

) {

setShowApplyButton(false);

setAlreadyApplied(true);

}

dispatch(HideLoading());

console.log(response);

if (response.success) {

setJobData(response.data);

} else {

message.error(response.message);

}

} catch (error) {

message.error(error.message);

}

};

const applyNow = async () => {

try {

dispatch(ShowLoading());

const response = await applyJobPost(jobData);

dispatch(HideLoading());

if (response.success) {

message.success(response.message);

navigate("/");

} else {

message.error(response.message);

}

} catch (error) {

message.error(error.message);

dispatch(HideLoading());

}

};

useEffect(() => {

getData();

}, []);

return (

jobData && (

<div>

<PageTitle title={jobData.Title} />

<Row>

<Col span={24}>

<div className="d-flex flex-column gap-1">

<div className="d-flex justify-content-between mt-1">

<span>Company</span>

<span>{jobData.company}</span>

</div>

<div className="d-flex justify-content-between">

<span>Location</span>

<span>{jobData.location.toUpperCase()}</span>

</div>

<div className="d-flex justify-content-between">

<span>Salary</span>

<span>{jobData.salary}</span>

</div>

<div className="d-flex justify-content-between">

<span>Experience</span>

<span>{jobData.experience} Years</span>

</div>

<div className="d-flex justify-content-between">

<span>Notice Period</span>

<span>{jobData.noticePeriod} Days</span>

</div>

<div className="d-flex justify-content-between">

<span>Posted On</span>

<span>{jobData.postedOn}</span>

</div>

<div className="d-flex justify-content-between">

<span>Job Type</span>

<span>{jobData.jobType}</span>

</div>

<div className="d-flex justify-content-between">

<span>Industry</span>

<span>{jobData.industry.toUpperCase()}</span>

</div>

<div className="d-flex justify-content-between">

<span>Last Date to Apply</span>

<span>{jobData.lastDateToApply}</span>

</div>

<div className="d-flex justify-content-between">

<span>Posted By</span>

<span>{jobData.postedByUserName}</span>

</div>

</div>

<hr></hr>

<h4 className="underline"> Job Description </h4>

<br></br>

<textarea className="pt-2" readOnly style={{ cursor: "default" }}>

{jobData.jobDescription}

</textarea>

{alreadyApplied && (

<div className="already-applied">

<span> You have already applied for this job. You can view your application status in the applied job section.</span>

</div> )}

<div className="d-flex gap-2">

<button

className="primary-outlined-btn"

onClick={() => navigate("/")}

>

CANCEL

</button>

{showApplyButton && (

<button className="primary-contained-btn" onClick={applyNow}>

APPLY NOW

</button>

)}

</div>

</Col>

</Row>

</div>

)

);

}

export default JobDesciption;

**Login.js**

import React from "react";

import { Form, message } from "antd";

import { Link } from "react-router-dom";

import { Button } from "antd";

import { LoginUser } from "../apis/authentication";

import { useDispatch } from "react-redux";

import { ShowLoading } from "../redux/alertSlice";

import { HideLoading } from "../redux/alertSlice";

function Login() {

const dispatch = useDispatch();

const onFinish = async (values) => {

try {

dispatch(ShowLoading())

const response = await LoginUser(values);

if (response.success) {

dispatch(HideLoading())

message.success(response.message);

localStorage.setItem("user", JSON.stringify(response.data));

window.location.href = "/";

} else {

dispatch(HideLoading())

message.error(response.message);

}

} catch (error) {

dispatch(HideLoading());

message.error(error.message);

}

};

return (

<div className="h-screen d-flex justify-content-center align-items-center bg-primary">

<div className="bg-white p-2 w-400">

<h3>LOGIN</h3>

<hr></hr>

<Form layout="vertical" onFinish={onFinish}>

<Form.Item name="email" label="Email" required>

<input type="text" required></input>

</Form.Item>

<Form.Item name="password" label="Password" required>

<input type="password" required></input>

</Form.Item>

<button className="primary-contained-btn w-100 mt-2" type="submit">

Login

</button>

<Link to="/register" className="d-block mt-2">

Not a member? Click here to register

</Link>

</Form>

</div>

</div>

);

}

export default Login;

**Notifications.js**

aimport React from "react";

import { useDispatch, useSelector } from "react-redux";

import PageTitle from "../components/PageTitle";

import { Tabs, Alert, message } from "antd";

import { HideLoading, ShowLoading } from "../redux/alertSlice";

import { SetReloadNotifications } from "../redux/notificationsSlice";

import { changeNotificationStatus } from "../apis/users";

import { useNavigate } from "react-router-dom";

const { TabPane } = Tabs;

function Notifications() {

const { readNotifications, unreadNotifications } = useSelector(

(state) => state.notifications

);

const navigate = useNavigate();

const dispatch=useDispatch();

const changeStatus = async(id,status)=>{

try{

dispatch(ShowLoading())

const response = await changeNotificationStatus(id,status);

if(response.success){

message.success(response.message)

dispatch(SetReloadNotifications(true))

}

dispatch(SetReloadNotifications(true));

}catch(error){

dispatch(HideLoading())

}

}

return (

<div>

<PageTitle title="Notifications" />

<Tabs defaultActiveKey="1">

<TabPane tab="Unread" key="1">

{unreadNotifications.map((notification, index) => (

<Alert

key={index}

message={

<div className="d-flex justify-content-between align-items-center">

<div className = 'd-flex flex-column'

onClick={() => navigate(notification.onClick)}

>

<span> {notification.title} </span>

<span>{notification.createdAt}</span>

<br></br>

{notification.title[0] === '#' && <small>Click on this blue banner to navigate to Interview</small>}

</div>

<span className="underline" onClick={()=>changeStatus(notification.id , 'read')}>Mark as read</span>

</div>

}

/>

))}

</TabPane>

<TabPane tab="Read" key="2">

{readNotifications.map((notification, index) => (

<Alert

key={index}

message={

<div className="d-flex justify-content-between align-items-center">

<div className = 'd-flex flex-column'>

<span> {notification.title} </span>

<span>{notification.createdAt}</span>

</div>

<span className="underline" onClick={()=>changeStatus(notification.id , 'unread')}>Mark as unread</span>

</div>

}

/>

))}

</TabPane>

</Tabs>

</div>

);

}

export default Notifications;

**Register.js**

import React from 'react'

import { Form , message } from "antd";

import {Link} from 'react-router-dom'

import { Button } from "antd";

import {RegisterUser} from '../apis/authentication';

import { ShowLoading } from '../redux/alertSlice';

import { HideLoading } from '../redux/alertSlice';

import {useDispatch } from 'react-redux';

import { useNavigate } from 'react-router-dom';

function Register() {

const dispatch = useDispatch();

const navigate = useNavigate();

const onFinish = async(values) =>{

try{

dispatch(ShowLoading());

const response = await RegisterUser(values);

if(response.success){

dispatch(HideLoading());

message.success(response.message);

navigate('/login');

}else{

dispatch(HideLoading());

message.error(response.message);

}

}catch(error){

dispatch(HideLoading());

message.error(error.message);

}

}

return (

<div className="h-screen d-flex justify-content-center align-items-center bg-primary">

<div className="bg-white p-2 w-400">

<h3>REGISTER</h3>

<hr></hr>

<Form layout="vertical" onFinish={onFinish}>

<Form.Item name="email" label="Email" required>

<input type="text" required></input>

</Form.Item>

<Form.Item name="Name" label="Name" required>

<input type="text" required></input>

</Form.Item>

<Form.Item name="password" label="Password" required>

<input type="password" required></input>

</Form.Item>

<button className="primary-contained-btn w-100 mt-2" type="submit">

Register

</button>

<Link to ="/login" className = "d-block mt-2" >

Already a member? Click here to login

</Link>

</Form>

</div>

</div>

)

}

export default Register

**AlertSlice.js**

import {createSlice} from "@reduxjs/toolkit";

export const alertSlice = createSlice({

name :'alert',

initialState:{

loading:false,

},

reducers:{

ShowLoading : (state) => {

state.loading = true;

},

HideLoading : (state) => {

state.loading = false;

}

}

})

export const {ShowLoading , HideLoading} = alertSlice.actions;

export default alertSlice.reducer;

**notificationSlice.js**

import {createSlice} from "@reduxjs/toolkit";

export const notificationsSlice = createSlice({

name :'alert',

initialState:{

readNotifications:[],

unreadNotifications:[],

reloadNotifications:true,

},

reducers:{

SetReadNotifications:(state,action) =>{

state.readNotifications=action.payload;

},

SetUnreadNotifications:(state,action) =>{

state.unreadNotifications=action.payload;

},

SetReloadNotifications:(state,action) =>{

state.reloadNotifications = action.payload;

}

}

})

export const {SetReadNotifications , SetUnreadNotifications , SetReloadNotifications} = notificationsSlice.actions;

export default notificationsSlice.reducer;

**Store.js**

import {configureStore} from '@reduxjs/toolkit';

import alertReducer from './alertSlice';

import notificationsReducer from './notificationsSlice';

const store = configureStore({

reducer:{

alert:alertReducer,

notifications: notificationsReducer,

}

})

export default store;

**Stylesheets**

**Custom-components.css**

.loader-parent {

position: absolute;

inset: 0;

background-color: rgba(31, 31, 31, 0.685);

display: flex;

justify-content: center;

align-items: center;

z-index: 989;

}

.page-title h1{

font-size:22px;

text-transform:uppercase;

}

.underline{

text-decoration: underline;

cursor:pointer;

}

.job-card{

border: 1px solid rgb(188, 182, 182) !important;

padding: 10px !important;

border-radius: 0px !important;

}

.job-card h3{

text-transform:uppercase;

font-size: 18px;

}

.already-applied{

background-color:green;

font-size :16px;

padding:15px;

color:rgb(255, 255, 255);

}

.ri-notification-line{

color: white !important;

}

layout.css

.layout{

display:flex;

padding:15px;

width:100%;

height:100vh;

gap:15px;

}

.sidebar{

display: flex;

align-items:center;

justify-content:center;

border-radius: 5px;

background-color: #171717;

color:#fff;

padding:15px;

}

.menu{

display:flex;

flex-direction: column;

gap:15px;

width: 150px;

}

.menu-item{

display:flex;

align-items: center;

gap:10px;

padding:10px;

border-radius:5px;

cursor:pointer;

}

.active-menu-item {

background-color:#504d4d69 !important;

color:white;

}

.content{

width:100% !important;

display:flex;

flex-direction: column;

gap:20px;

}

.header{

background-color:#171717;

padding : 20px;

color:#fff !important;

border-radius: 5px;

width:100%;

font-size:20px;

}

.logo{

font-size:25px !important;

}

.body{

background-color:#fff;

padding:15px;

border: 1px solid rgba(111,111,111);

border-radius: 5px;

width:100%;

height:85vh;

overflow:auto;

}

i{

font-size: 20px !important;

cursor:pointer;

}

h1, h2 , h3 , h4 , h5, h6{

margin-bottom:0 !important;

}

**index.css**

@import url("https://fonts.googleapis.com/css2?family=Montserrat:wght@500&display=swap");

\* {

font-family: "Montserrat", sans-serif;

}

.h-screen {

height: 100vh;

}

.bg-primary {

background-color: #171717 !important;

}

.bg-white {

background-color: #fff !important;

}

.w-400 {

width: 400px !important;

}

input, select ,

textarea {

border: 1px solid gray !important;

width: 100% !important;

padding: 8px 20px;

}

input:focus,

textarea:focus {

outline: none !important;

border: 1px solid #171717 !important;

}

textarea {

min-height: 250px !important;

max-height: 1000px !important;

}

.primary-contained-btn {

background-color: #171717 !important;

border-radius: 0 !important;

height: 40px;

color: white !important;

text-transform: uppercase !important;

padding: 0 20px !important;

margin-right:5px;

margin-top:10px;

}

.primary-outlined-btn {

border: 1px solid #171717 !important;

border-radius: 0 !important;

height: 40px;

color: #171717 !important;

text-transform: uppercase !important;

background-color: white;

padding: 0 20px !important;

margin-top:10px;

}

.ant-form-iten {

margin-bottom: 10px !important;

}

a {

color: #171717 !important;

text-decoration: none;

}

# **Chapter 7 Conclusion and Future Work**

## 7.1 Conclusion

In conclusion, the development of the job searching and sharing platform using React, Redux, and Firebase has been successful. The platform provides a user-friendly interface for job seekers to search for job listings and for employers to share job opportunities. The integration of Firebase services ensures secure authentication and efficient storage of job listings. Through the testing process, the platform has been thoroughly validated and meets the specified requirements.

The job searching and sharing platform offers several key features, including job search functionality, job listing submission, user authentication, user profiles, and notifications. These features provide a comprehensive and intuitive experience for both job seekers and employers. The platform's performance and scalability have been optimized to handle a large volume of data and user interactions.

## 7.2 Future Work:

While the current implementation of the job searching and sharing platform is functional and meets the initial requirements, there are several areas for potential future enhancements and expansion:

7.2.1 Enhanced Search Functionality**:**

The search feature can be further improved by implementing advanced search filters, such as location-based search, salary range, and job category filters. This would provide users with more refined and personalized search results.

## 7.2.2 Recommendation System:

Implementing a recommendation system that suggests relevant job listings based on a user's profile, search history, and preferences can greatly enhance the user experience and help users discover new opportunities.

## 7.2.3 Advanced User Profiles:

Enhance user profiles by allowing users to add more detailed information about their skills, experiences, and portfolios. This can help employers get a better understanding of candidates and assist job seekers in showcasing their abilities.

Bibliography

1. Firebase Documentation:

Firebase. "Firebase Documentation." Firebase, Google, URL: <https://firebase.google.com/docs>.

2. React Documentation:

React. "React Documentation." React, Facebook, URL: <https://reactjs.org/docs>.

3. LinkedIn:

LinkedIn. "LinkedIn." LinkedIn, LinkedIn Corporation, URL: [https://www.linkedin.com](https://www.linkedin.com/).

4. Redux Documentation:

Redux. “Redux”. URL: <https://redux.js.org/>

5. CryptJS Documentation:

CryptJS. URL: [github.com/brix/crypto-js](http://github.com/brix/crypto-js)

6. MomentJS Documentation:

MomentJS. URL: <https://momentjs.com/>

7. Google.[URL : https://google.com/](file:///C:\Users\91765\AppData\Local\Microsoft\Windows\INetCache\IE\R9VI1ZHQ\URL%20:%20https:\google.com\)

8. <https://www.uptech.team/blog/how-to-make-a-social-media-app>

9. <https://www.youtube.com/@Fireship>