# Software design document

Authored by:

# Zeshan hussain, Ahmed Qadoura, Ahmed ali, Shamas hussain

# Database Design

The database is structured for optimal data management. It uses a relational model using MySQL, known to offer great support for querying. The tables are designed to manage and maintain user profiles, job listings and saved job preferences. The database provides the backbone for employees to use the portal with ease., whilst ensuring data integrity

The database utilises three tables:

* Users
* Jobs
* Saved jobs

## Users Table

The Users Table is part of the system which is essential for storing users' details such as their emails, usernames, date of birth and passwords. User authentication is a vital feature of an online job portal system, and the system uses bcrypt hashing for password security. The table also contains timestamps of account creation and modification. There is also a unique constraint applied to the email column to ensure distinct account existence for each user to prevent duplicate registrations.

## Jobs Table

The Jobs table contains job postings with attributes such as title, company, location, employment type, description, requirements, and salary along with the skills needed. This level of detail enables a job seeker to filter and search for jobs aligned with their preferences. As it stands, the salary field is stored as a text string (e.g., “100k – 130k”) which will not be efficient for numeric comparisons. A better solution would be to store salary as two separate integer fields (min\_salary and max\_salary) which would facilitate better filtering and sorting.

In addition, the skills column is stored as a text field separated by commas, which is not the best way to query certain skills. A many-to-many relationship between jobs and skills, with the use of a separate Job\_Skills table, would offer a more effective means of filtering jobs by required skills. This would be more compliant with industry standards and allow for more granular skill-based searches.

## Saved Jobs Table

The Saved Jobs table allows users to save job postings that they may want to look at later. It creates a many-to-many association with users and jobs through the use of foreign keys, so every saved job record under a user corresponds to a legitimate user and job record. Users are also prevented from saving a job numerous times due to the unique constraint on the composite of user\_id and job\_id. While practical, the table could be improved by adding a timestamp detail to indicate when a job was saved so that user job preferences and engagements can be analyzed.

## Normalization, Indexing and Optimization

The database implements normalisation, which minimises duplication and ensures uniformity among the tables. Data integrity is ensured through the implementation of primary and foreign keys, while for the Saved Jobs table, there are indexes placed on frequently queried fields, such as job titles and company names, that facilitates faster searching within the table. For other fields such as location, skills, and salary, additional indexing can further improve query speed, especially for users performing job searches with many filters.

To conclude, the existing database structure accommodates job posting, user management, and saving a job effectively. Despite this, some changes, including enhanced salary storage, better representation of skills, and new user roles, can dramatically improve a system’s scalability and efficiency. An adoption of industry standards on relational database management would allow the job portal to refine its search features and overall usability.

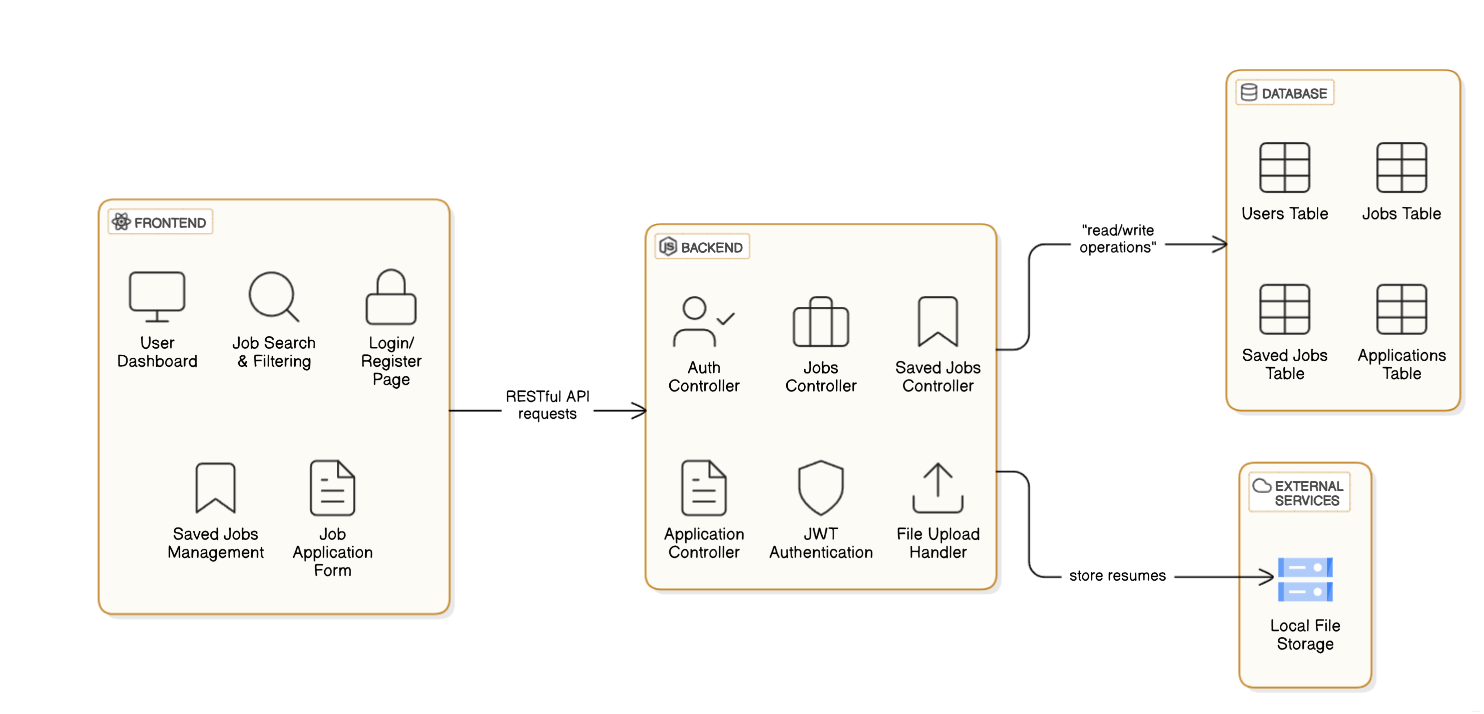
# Backend Implementation

Backend programming of the job portal is the most important part that deals with the business logic, database processing, authentication, and communication through APIs with the frontend and the database. The system is constructed with Node and Express which means that server side will be light and have the ability to scale. The backend has been designed to provide strong RESTful APIs through which users of the system, that is, employers and job seekers, can perform their functions conveniently. It also features strong mechanisms for authentication, validation of user inputs, and error handling to maintain the integrity of the stored information and ensure data security.

## System Architecture and Components

The backend is constructed integrally modular, separating the functionalities into services like authentication, job services, user services, and saved jobs service. Each module connects to MySql database with the help of mysql2/promise package which offers efficient query execution and connection pooling. The middleware layer authenticates users through JWT (JSON Web Token) and makes sure that restricted endpoints can only be accessed by authorized users.

The interaction of the different pieces is represented in Figure 1. The Frontend interacts with the Backend through RESTful API calls, which the Backend fulfills by querying or updating information in the MySQL database. The authentication module checks the users’ identities, while security policies are implemented by Middleware functions. The Job Management service fetches available jobs according to the user specified parameters while the Application service enables users to apply for jobs in a safe manner.



**Figure 1: Architecture Diagram**

## Authentication and Security

One of the most challenging areas in the implementation of the system is the backend authentication. The system implements JWT-based authentication. After a user logs in successfully, a JWT token is issued. This token has to be placed in the headers of all fetched API calls, and this is used by the backend to identify the user. Also, users’ passwords are hashed using bcrypt.js before they are saved into the database. In the case of database leakages, the credentials are still protected, something that users' credentials are not secured by will.

Before allowing access to certain routes, authentication middleware makes sure to verify the existence and validity of JWT tokens to avert unauthorized access. If a user’s request is not valid, the server returns an error message.

## Job Management and Search Functionality

The backend infrastructure effectively manages job posting as well as searching and filtering. When a user looks for a job, the system constructs a search filter containing attributes such as title, company name, location, job type, salary, and skills required. This filter is sent to the MySQL database in the form of parameterized queries. The backend makes efficient use of several filters by including only the absolutely required conditions to the final query execution only to guarantee optimal performance of the database.

To make searching faster, frequently searched fields like job title, locality, and company name are indexed. The response times to the queries are reduced due to these indexes, which results in faster and more efficient searches for jobs.

## Interactions with Users: Job Applications and More

The job applications process is handled through a specific API endpoint that allows users with valid credentials to apply for specific jobs. Each application is defined by a user identifier, job identifier, applicant’s full name, email address, and a cover letter. The system is built in such a way that a user can only apply for a job once, ensuring that duplicate applications are not submitted.

Users are also able to upload files pertaining to job applications, such as resumes in either PDF or DOCX format. Multer.js is the package that enables secure file uploads, including advanced validation that specifies what file types may be uploaded.

## User Preferences and Saved Jobs

Users have the ability to bookmark jobs in the portal so that they may be viewed later. The Saved Jobs API maintains a many-to-many relationship between users and job listings, making it possible for every user to bookmark multiple jobs without them being saved more than once. Users are also able to save or remove jobs using a specific API endpoint which alters the database.

The saved jobs feature not only helps in tracking user activity but at the same time can help in future development iterations by implementing tailored job advertisements through personalized job recommendations.

## Error Handling and Performance Optimization

The backend utilises a centralised form of error handling, which guarantees that all API responding communications follow a distinctive pattern. Express.js middleware hooks capture mistakes at various points in time throughout the processing and serve a formulated error. For the purpose of optimising the performance of the backend, the following are implemented:

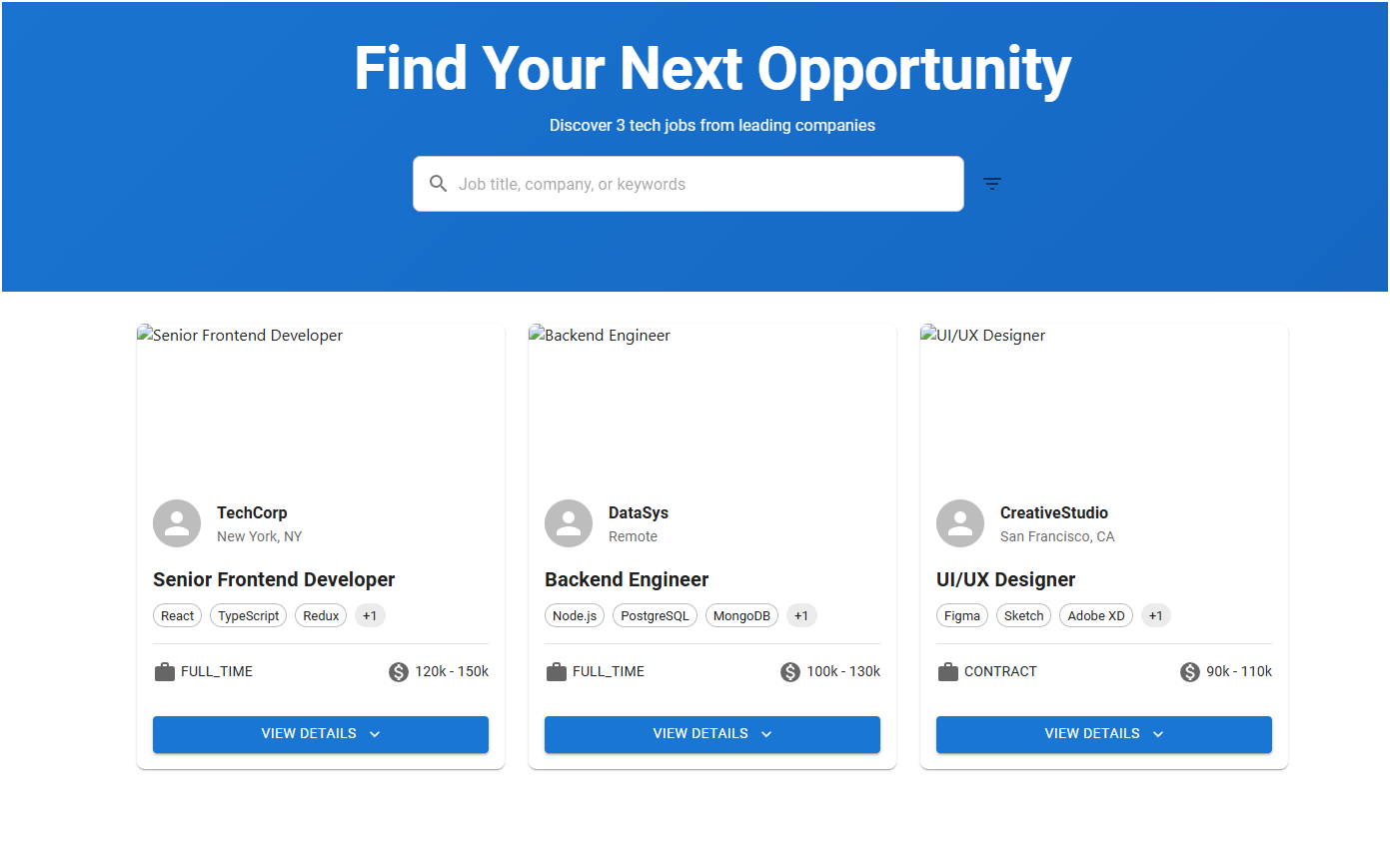
1. Connection Pooling: Using the mysql2/promise package allows for better handling of database connections which avoids the cost of opening new connections for every request.
2. Caching: Job listings that users access very often can be cached using Redis or other in-memory caches to prevent putting extra stress on the database.

# Frontend Implementation

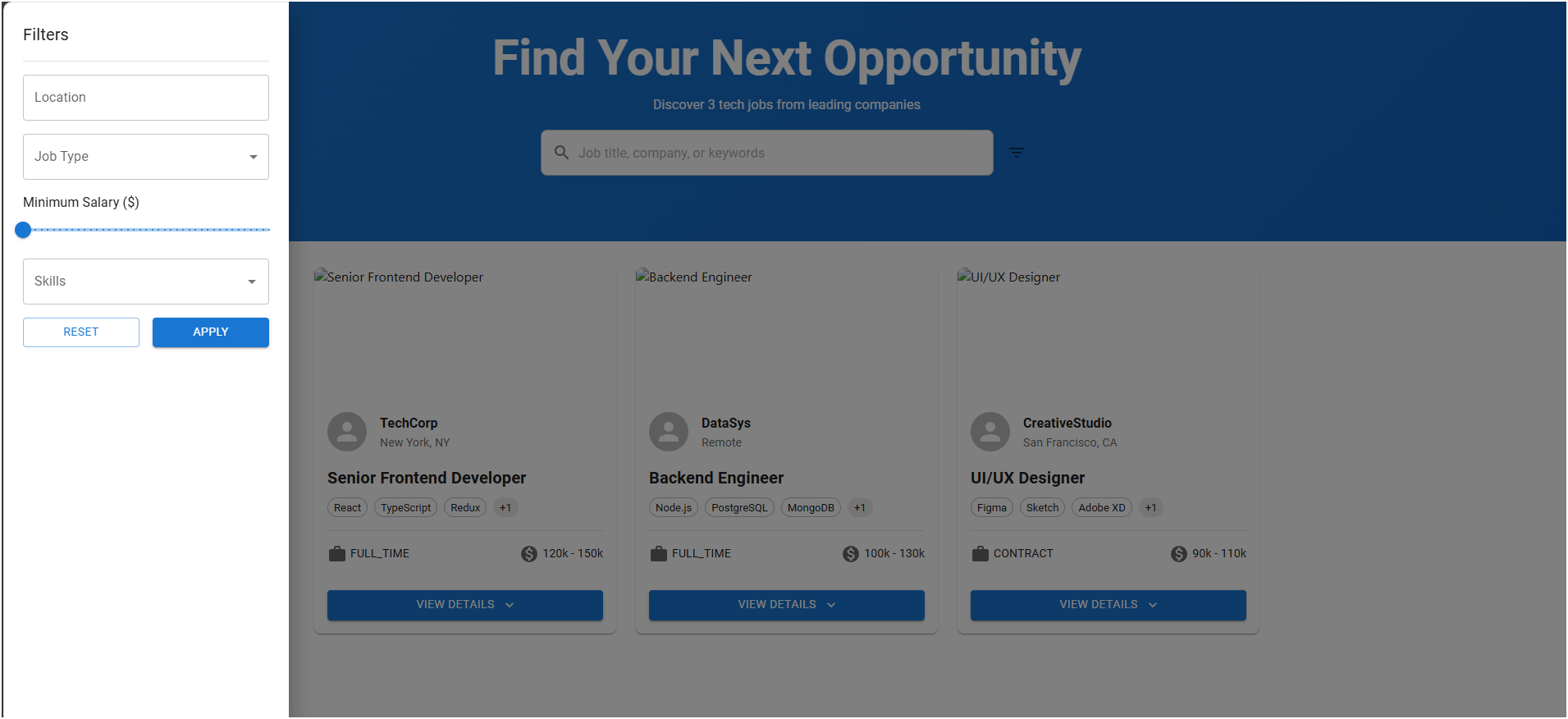
The job portal front end is interactive, modern, and user-friendly with the intention of helping job seekers easily look for job listings, filter results, and apply to jobs with ease. The implementation is done using React.js which is a highly responsive user interface component-based JavaScript library. To improve design and user experience, Material-UI (MUI) is incorporated. MUI provides pre-styled components to make the design and overall functionality of the platform better. The users and back-end interface through RESTful APIs, getting job listings, authenticating users, and applying for jobs are done concurrently.

## User Interface and Dashboard

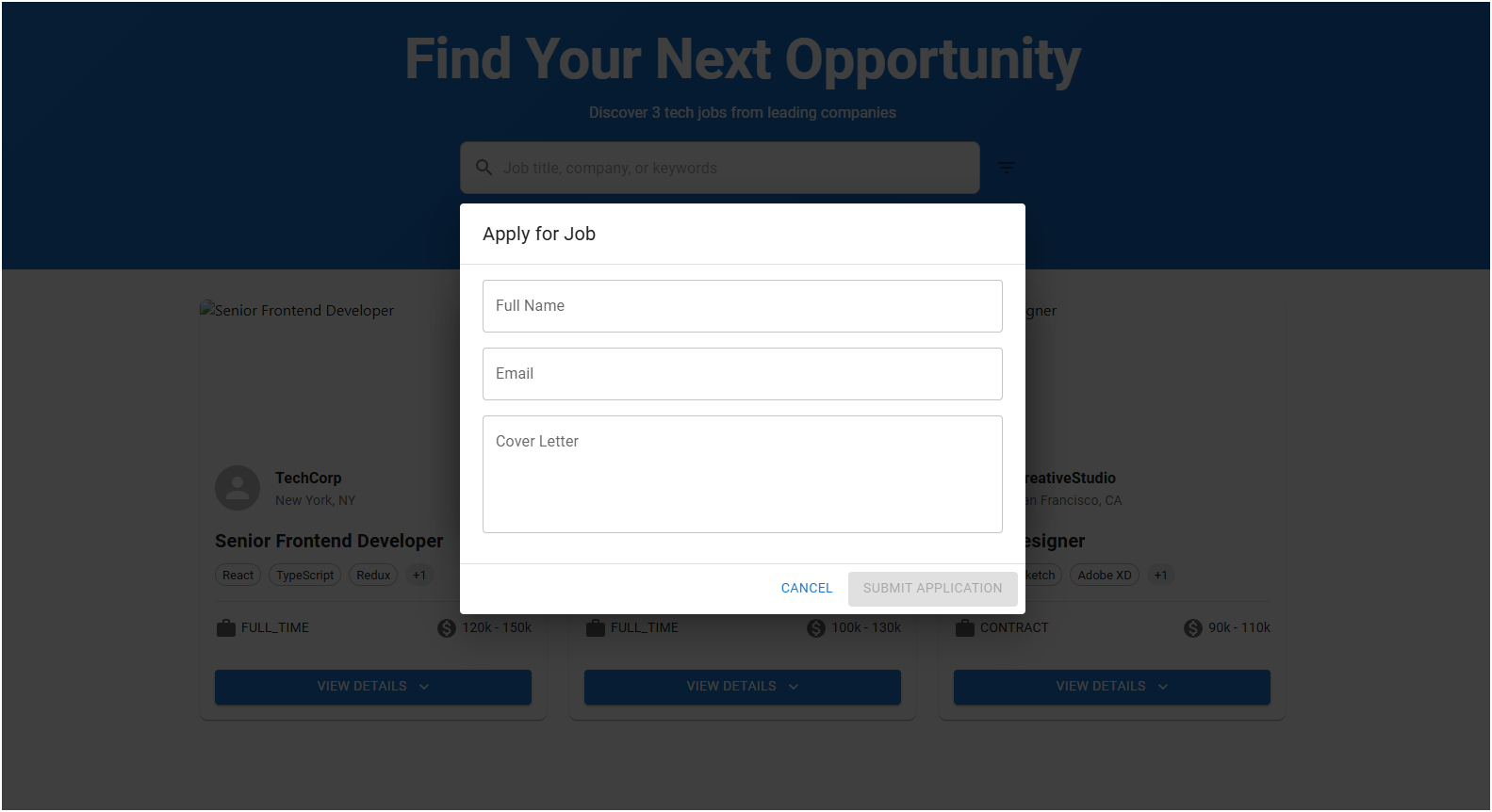
The dashboard is the main view for users to search and see the jobs available. It is organised and modern, displaying the jobs in cards with company information, job titles, salaries, and required skills in a clean way. The interface has a search bar and enables subsequent filtering of the job listings based on keywords, job type, location, and the salary range for the position being advertised. The filtering system updates on real time user input.



To aid in user interaction, the dashboard is enhanced with additional animated effects using Framer Motion, such as hover animations and motion transitions. Users utilising the platform across various devices are also catered to through the mobile responsive feature of the dashboard.



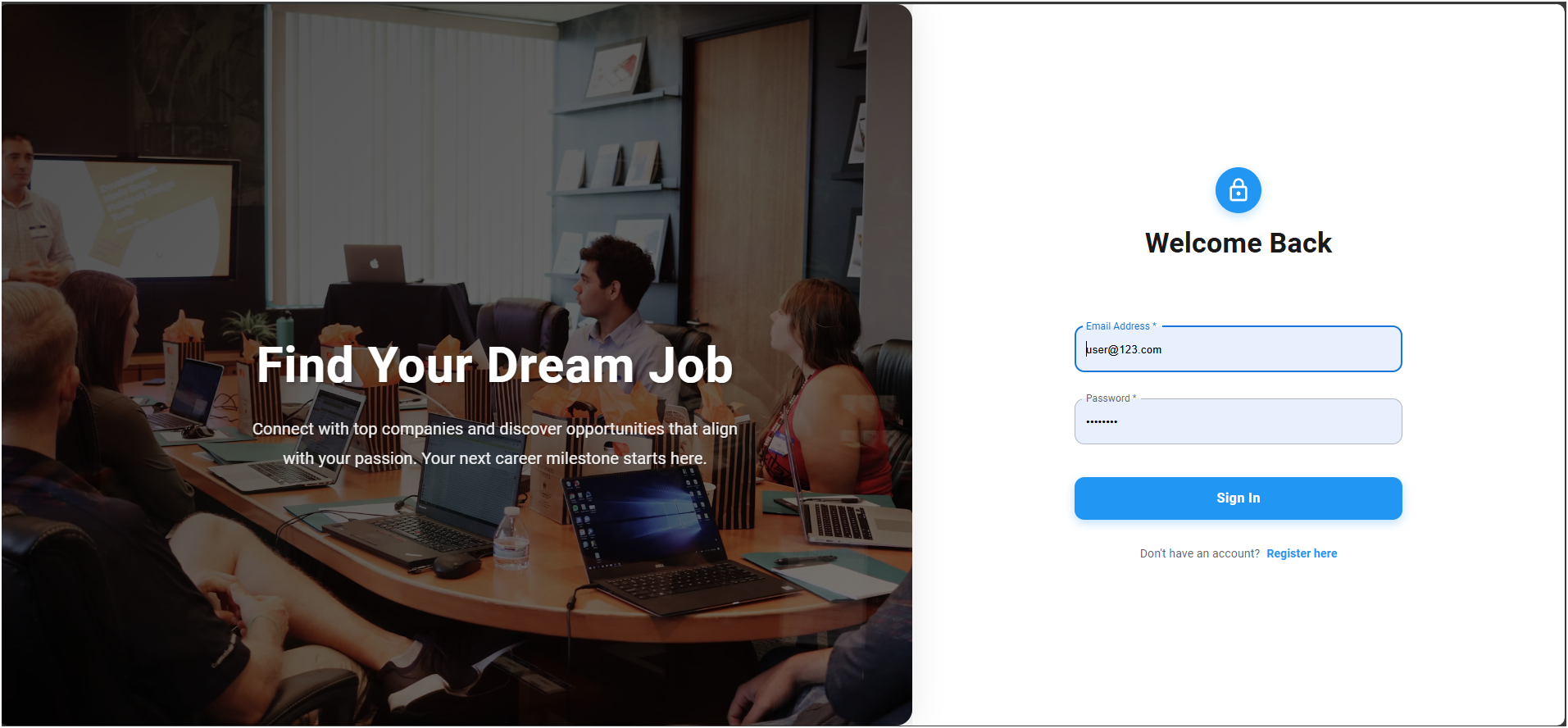
As seen on the screenshot above, we can see what the job listing dashboard looks like. Notice how clean and simple the interface is.

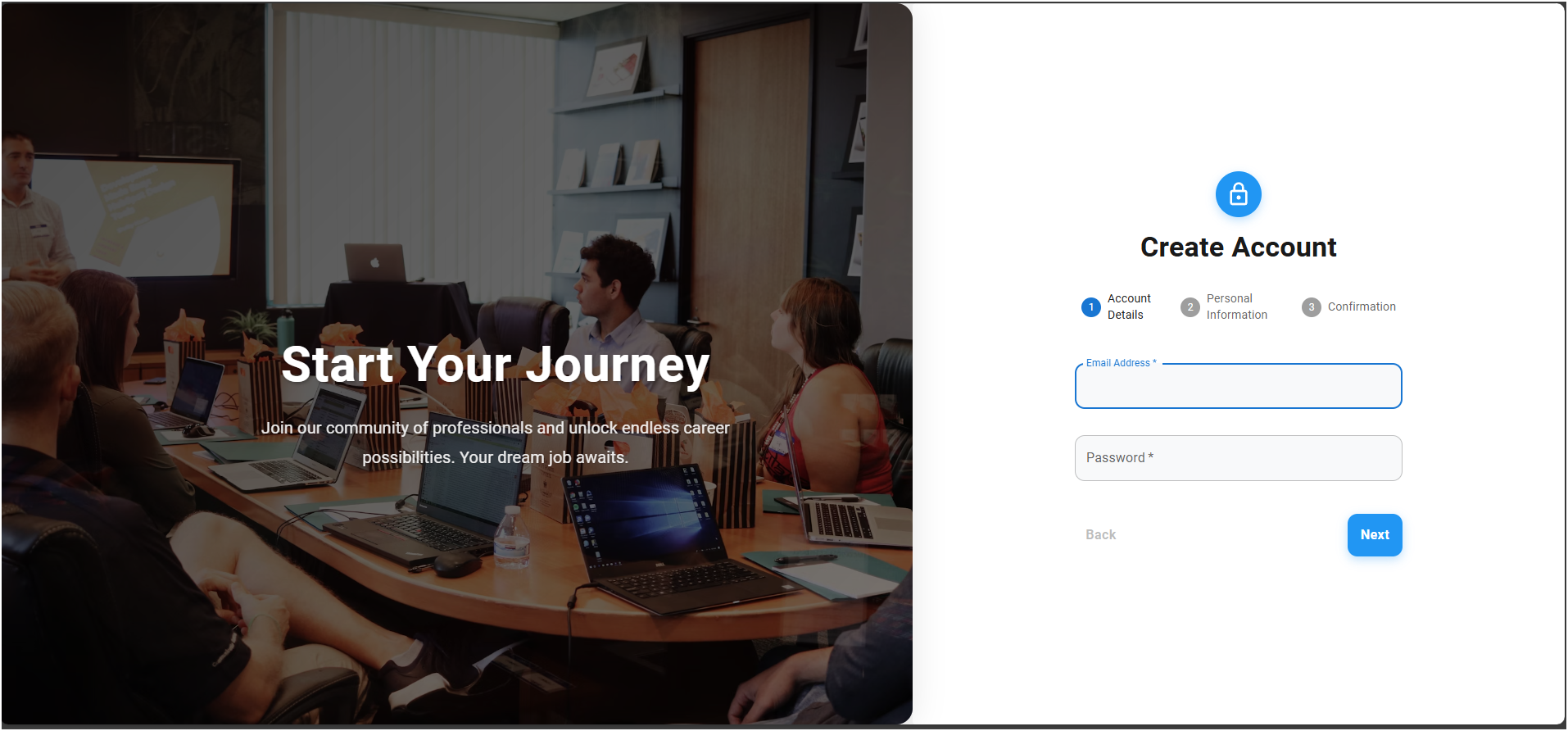


# Authentication and User Sessions

In the frontend functionality, user authentication is important to deal with. The login and registration pages use React Forms and also have some form of validation to make sure the users actually fill in the correct details. User authentication includes logging into the platform via account credentials, which saves a user’s JWT token in local storage and allows users to remain logged in. The validation process also comes with error handling when dealing with invalid credentials or failure to register and this information gets displayed to the user in form of messages on the front-end.

The screenshot of the user login page below shows how visually uncluttered the user has to interact with when filling in their account details. The layout of the form itself is clean and allows for easy input of the user’s credentials.





The user simply follows steps to fill out the account setup form with necessary details which include email address, password, username, and date of birth.

# Job Applications and User Engagement

Upon locating an appropriate job, users can click on the job card to view a full description of the job, including any qualifications that may be attached to it and the job responsibilities that are required. The modal has an “Apply Now” button which provides an application form to capture the user’ basic information and upload their resumes. The application form was developed with React-controlled components that ensure managed input and validation before submission.

The job application form is user friendly with a basic outline that asks for a user’ first and last names, email address, and cover letter. It also allows for resume uploads in PDF or DOCX formats, which is enabled by Multer.js at the backend. A screenshot of the job application modal is provided below: