Table of Contents

[Introduction 2](#_Toc185158529)

[1. Research and Planning 3](#_Toc185158530)

[Project Overview 3](#_Toc185158531)

[Objectives 3](#_Toc185158532)

[Requirements Analysis 3](#_Toc185158533)

[Tools and Technologies 4](#_Toc185158534)

[2. Choice of Framework and Technologies 4](#_Toc185158535)

[Front-end: React 4](#_Toc185158536)

[Back-end: Node.js and Express.js 5](#_Toc185158537)

[Database 5](#_Toc185158538)

[3. UX Design 6](#_Toc185158539)

[Design Approach 6](#_Toc185158540)

[Wireframe 6](#_Toc185158541)

[4. Web Services 8](#_Toc185158542)

[REST API Endpoints 8](#_Toc185158543)

[Example Request/Response 9](#_Toc185158544)

[Tools used for testing API endpoints: 9](#_Toc185158545)

[5. Security Threats and Measures 10](#_Toc185158546)

[Identified Threats 10](#_Toc185158547)

[Mitigation Strategies 11](#_Toc185158548)

[6. Features 11](#_Toc185158549)

[Employee Features 11](#_Toc185158550)

[Employer Features 11](#_Toc185158551)

[Future Enhancements 11](#_Toc185158552)

[7. Design Patterns 11](#_Toc185158553)

[Middleware for Authentication 12](#_Toc185158554)

[8. Deployment 12](#_Toc185158555)

[Deployment Process 12](#_Toc185158556)

[Challenges Faced 12](#_Toc185158557)

[9. Learning Experience 12](#_Toc185158558)

[10. GitHub Repository 13](#_Toc185158559)

[Conclusion 13](#_Toc185158560)

## **Introduction**

Our app is based on a job portal where there is involvement of two parties, employer and employee. This app has many features plus security features integrated into the web app, such as both parties having different interfaces to interact with the website. For employers, they can post their jobs with a few details such as job title, description, date posted, job type, etc., in which they can also perform tasks like editing their post, deleting their post, or posting a new job. For employees, they can view the job, apply for that job, search for jobs, and can also add a note based on a job they target for the future. This app is not limited to these features, as it also includes authentication where employers and employees can register themselves individually. The app also has some security features like password encryption using hashing and JSON Web Tokens, etc. We will discuss more of the technologies below to see how and what we have implemented in our web application. Also, there are some API integration from third party like Rapid API which is opensource for everyone through which integrated some of the jobs and posted on our web app.

## **1. Research and Planning**

### ****Project Overview****

Our Job portal have two parties known as employer and employee who can perform some of the tasks and also third party to integrate posted jobs:

* Employer: The employer can post, edit, delete the jobs
* Employee: View the jobs, also can add notes to remind him for target company and can delete notes.
* Rapid API: To integrate posted jobs through external api’s

**This web application is based on two frameworks one is react for frontend design and Nodejs for backend implementation. This application has three security features such as Json web token, cookies to store some user information, Bcrypt to encrypt user password whether it is employee or employer.**

### ****Objectives****

* **To create web application which benefitted both employer and employee**
* **To create web application where user information is confidential and secured**
* **To create a web application which is user friendly and fast in response to user.**
* **To create a web application which contains proper database integration using Schema.**
* **To create a web application which follows industrial standard to design code structure and designs.**
* **To create a web application which has attractive and frontend design that it can attracts more user.**

### ****Requirements Analysis****

#### ****User Roles****:

* Employer: Post, edit, delete job postings.
* Employee: View and search for job postings.

#### ****Features****:

* Authentication: Integrated json web token to securely login and registers
* Job Listings Use of crud operations to list a job post
* Note bases system: Employee can create notes regarding job post to create reminder for new jobs.

#### ****Constraints****:

* Standalone web application in a single storage.
* Follows monolithic architecture
* Scalability for future expansion.

### ****Tools and Technologies****

* **Front-end**: React, Fetch JavaScript based API calls, Cookies
* **Back-end**: Node.js, Express.js.
* **Database**: MongoDB (NoSQL) for database
* **Security**: Bcrypt for password hashing, JWT for authentication.
* **Storage**: GitHub repository to store website
* **Deployment**: Render web application.

# **2. Choice of Framework and Technologies**

## **Front-end: React**

REACT framework is developed by React which is a popular JavaScript library for building user interfaces specially for single page applications. It allows developers to reuse UI as components in its structure and development becomes faster and easy to maintain. Its component-based architecture makes modular development as each component takes care of its state and logic, making code reusable and scalable.

Some of the libraries we have used in react to implement logics and to connect API calls

**React Router**: For dynamic routing within the application.

* **useNavigate, useLocation : These are some functions which we have use to navigate the URL’s and fetch the query and params from URL address.**

**Fetch**: To handle API requests and responses efficiently.

* **React Hooks**: We have used usestate, useeffect to store the variables and calls the functions and logic as react render the components every time hooks calls up.

## **Back-end: Node.js and Express.js**

Node.js is a framework that developers use to run JavaScript code on the server side, outside of a web browser. Before time some developer’s user nodejs to create interactive websites but now Node.js is using to create server backend by the developers. Also, developer can create routes for pages, API calls which can interact with frontend or external API’s also it allows backend to interact with a dataset in which user can crud operations and manipulations with database. Nodejs have some libraries through which user can extend the functionalities of backend.

## **Database**

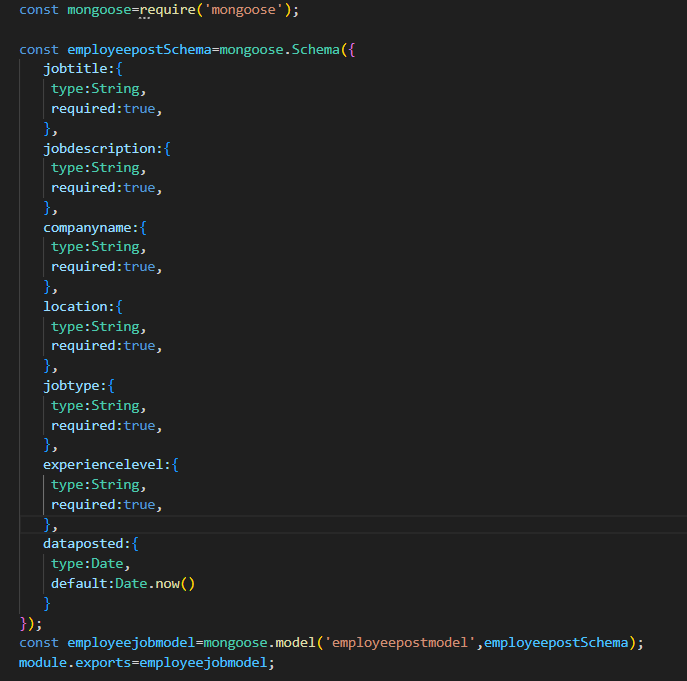
For our web application we chose MongoDB as a database which based on NoSQL which contains structured, semi structured and non-structured data in databased. Also, Mongo DB is easy to use and can use by coder who are beginner and have some knowledge of JSON known as Json object notation. So, for so many reasons we chose MongoDB as a database.

**Schema and Model**

Schema is something in which we define the fields and their datatype with some additional properties to define how our table adds data to it. It is a structure of table which stores on a database and model is the final one which creates the logic and interactions with the data while schema we can use to create read, update, and delete operation

****

**Schema to add notes by the employee**



**Post Model and schema**

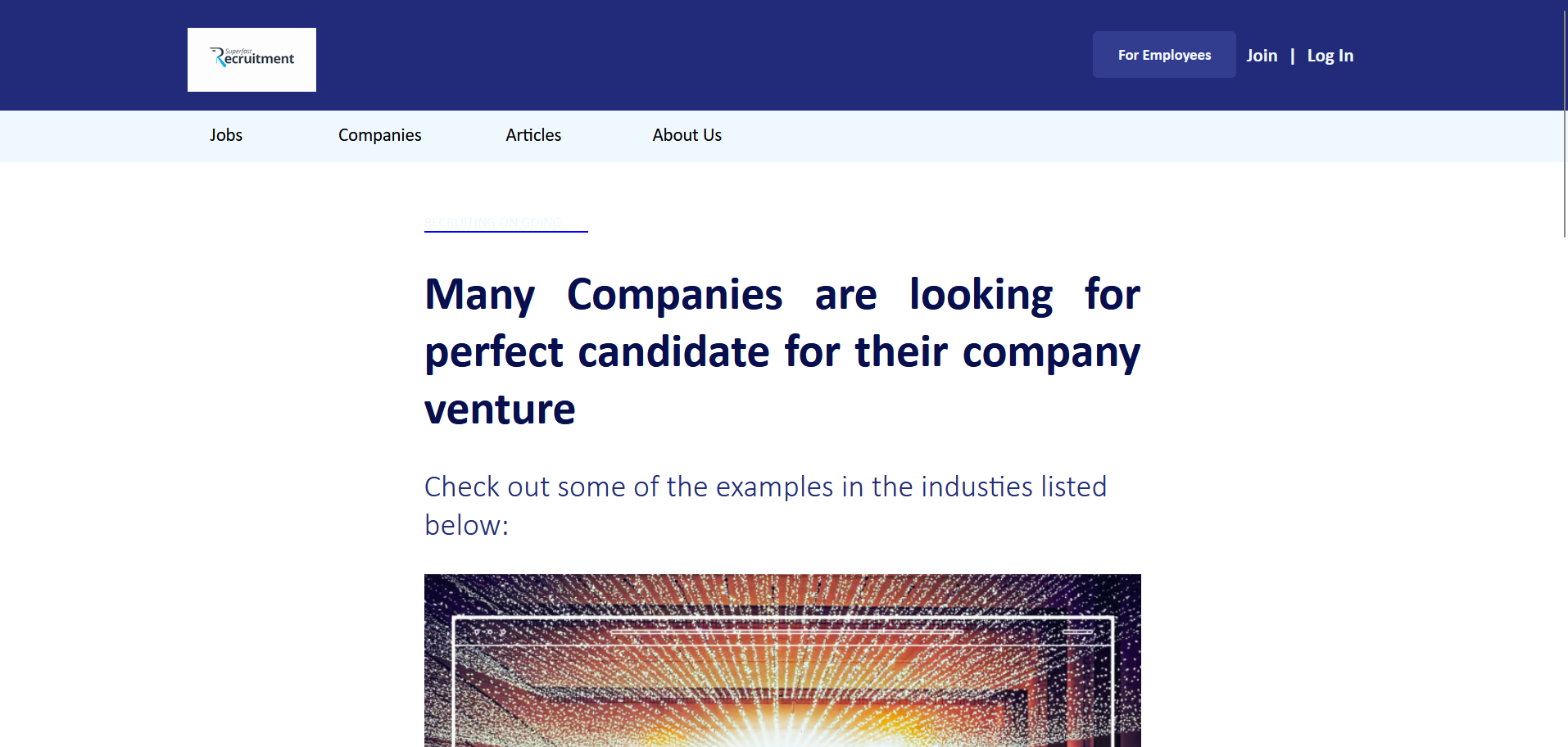
# **3. UX Design**

## **Design Approach**

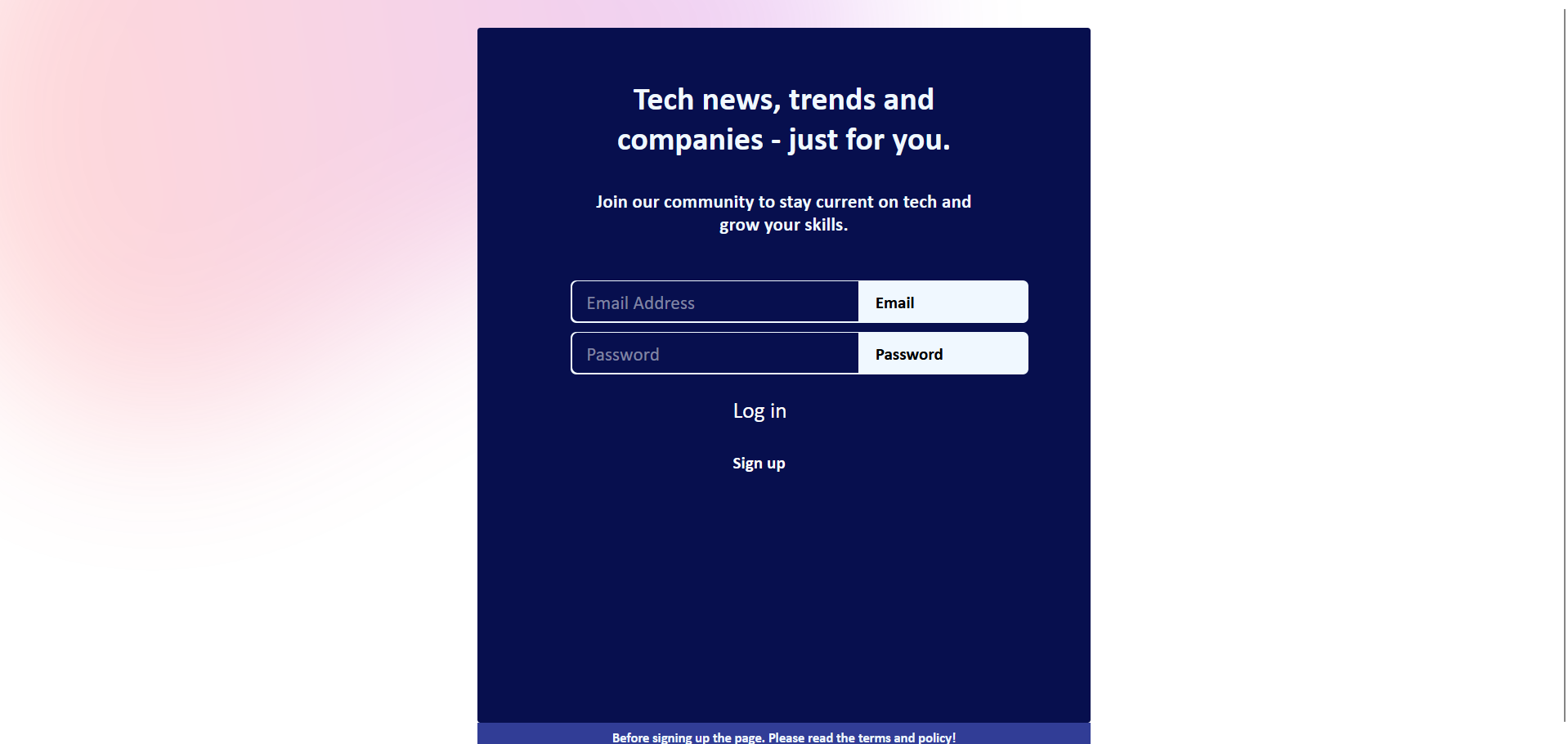
* **User-friendly**: This contains simple and user-friendly pages like for employees and employers like for employers have three pages posting, editing, and deleting of post where employee have three pages like homepage, adding notes, view posts which contains external and internal posts.
* **Responsive**: Website is responsive for some of the resolutions like HD and low-resolution laptops and screen size but for future we will enhance its availabilities for more devices

## **Wireframe**

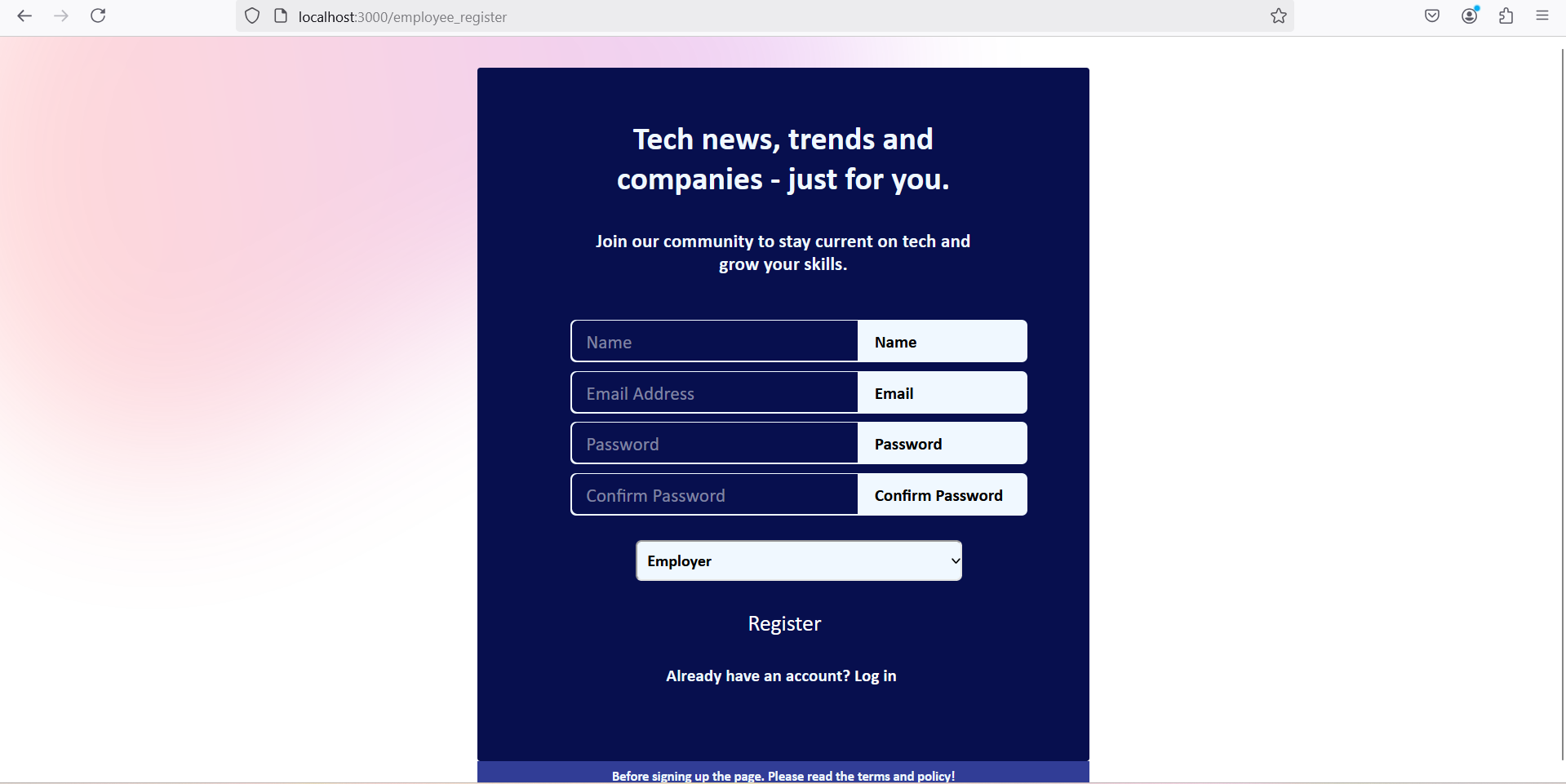
* **Homepage**: It contains the introduction of the website and some descriptions about the Job portal with some quotes and photos
* **Employer Dashboard**: It includes edit, delete, create post for the user.
* **Employee Dashboard**: Displays the job listing created by the employer and also contains some notes section the employee to get remind for future jobs.

****

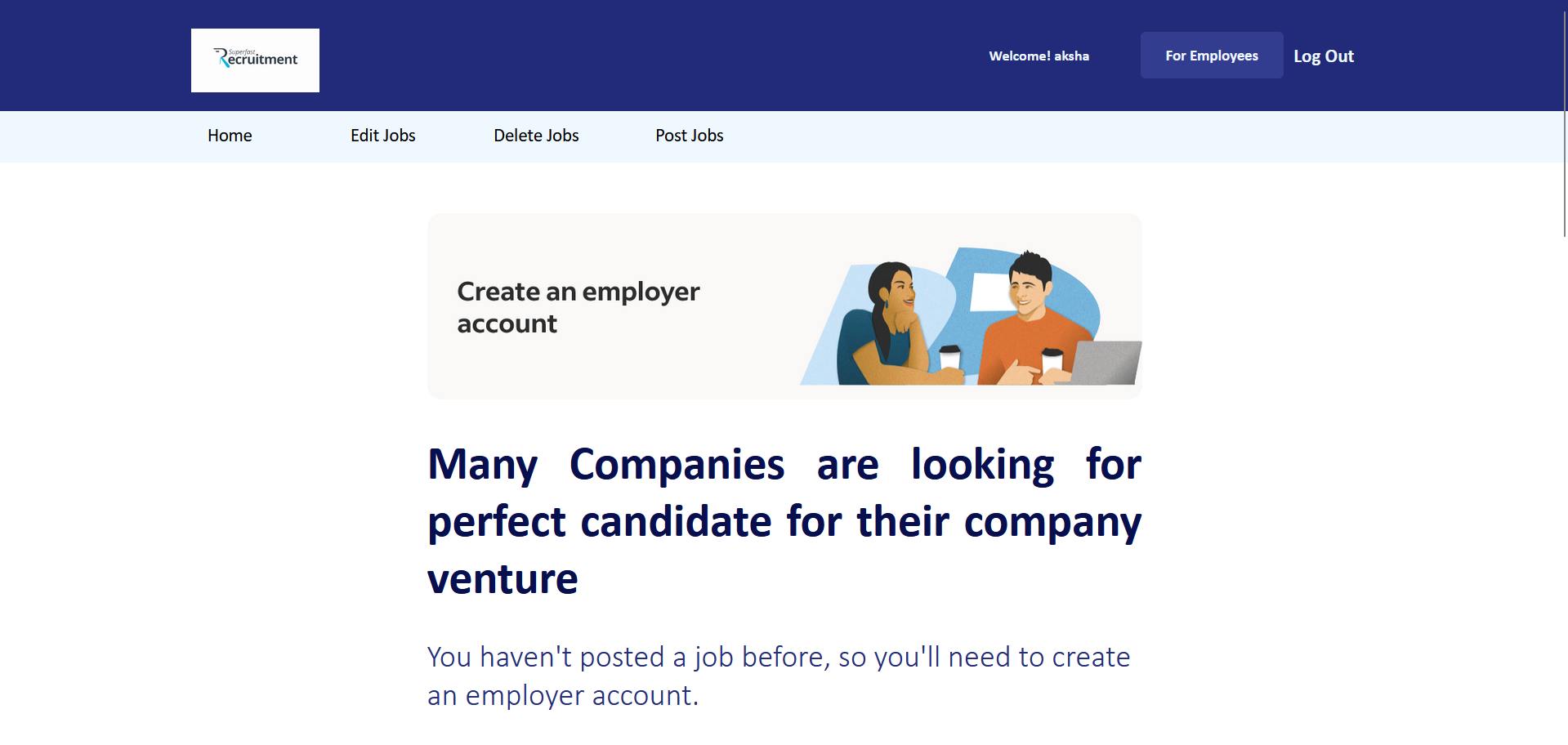
**Front Page**

****

**Login Page**

****

**Register Page**

****

**Employer Page**

## **4. Web Services**

## **REST API Endpoints**

Our backend works on the basis restful API which connects both frontend and backend and can communicate and exchange information through API using methods post, put, delete and get. As we have used crud operations in a web application so to create information, updating and manipulating, this API required.

#### Endpoints used to exchange information between client(ReactJS) and server(NodeJS):

1. **Authentication**:
   * POST /joining/login: User login: User login
   * POST /joining/employerRegister: User registration.
   * Get /joining/logout: for logout and expire json web token and delete cookies
2. **Job Management**:
   * POST /joining/jobpostform: Create a job post (Employer)
   * PUT /joining/jobpostform: Edit a job post (Employer).
   * DELETE /joining/jobdeletepost/:id: Delete a job post (Employer).
   * GET /joining/jobpostview: To fetch job listings (Employee).
   * POST /joining/fetchnotes :To add notes to the database of employee using specific id of employee which saved at the time of token exchange while login
3. **Search and Filter**:
   * GET /search?name=${name} :To search the item in API calls
4. **Extenal API**

https://jobs-api14.p.rapidapi.com/v2/list?query=Web%20Developer&location=United%20States&autoTranslateLocation=false&remoteOnly=false&employmentTypes=fulltime%3Bparttime%3Bintern%3Bcontractor

This is the external API used to fetch job post using Job API on RapidAPI platform. This API need API key and host name to access the content for job post on our portal

## **Example Request/Response**

#### ****POST /joining/employeRegister****

**Request**:

{

"name": "John Doe",

"email": "john@example.com",

"password": "password123"

“option” : “Employer”

}

**Response**:

{

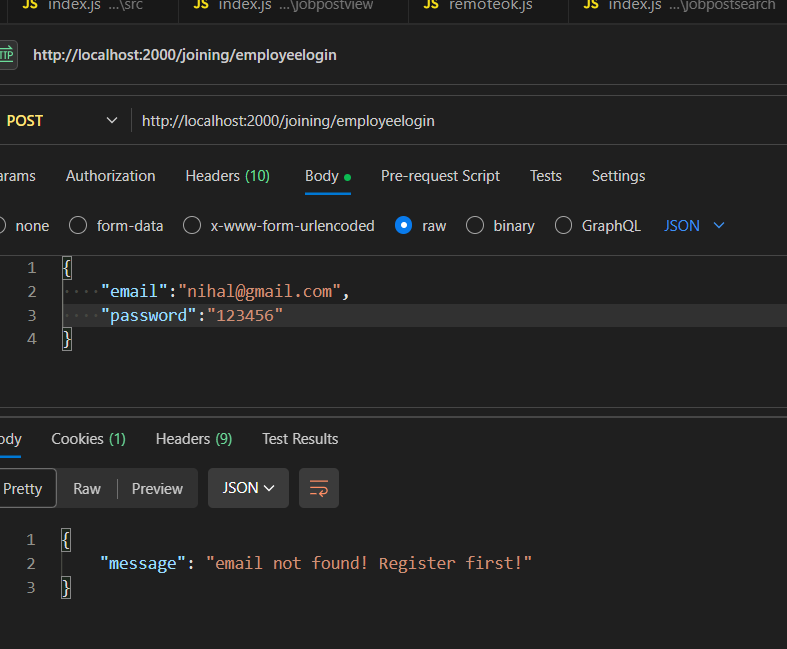
"message": "User registered successfully",

"token": "<JWT\_TOKEN>"

}

## **Tools used for testing API endpoints:**

* **Postman**: Postman is the tool which internally installed in visual studio code as a plugin where we can add API endpoints by adding raw data, headers, or different formats to exchange information through server. So, we have used this tool to test the endpoints created in server to check whether authentication or API calls are working or not. Also, we have tested the external API using this by adding API key using headers in postman.



## **Security Threats and Measures**

## **Identified Threats**

* **SQL Injection: Hacker can manipulate queries written within application using input fields**
* **Cross-Site Scripting (XSS): Hacker can use malicious scripts to inject within a web application.**
* **Data Breaches: Sensitive information like user credentials can be compromised**

## **Mitigation Strategies**

* **Input Validation: Preventable injection attack using user inputs.**
* **Authentication: JWT tokens are used for API control of authorization.**
* **Password Encryption: Hash passwords with bcrypt.**
* **HTTPS: We make sure that between server and client all communication is encrypted.**
* **CORS Policy: Keep your API only accessed by trusted domains.**

## **6. Features**

### ****Employee Features****

1. **View Job Listings**: Employee can fetch jobs in job portal through internal employer and external job posts.
2. **Apply for Jobs** (Optional): Employee can submit job application through jobs posted.
3. **Add Reminder: Employee can add reminder to save job for future use or will target in future**

### ****Employer Features****

1. **Post Jobs**: Employer can create job post using title, type, date, description, etc.
2. **Manage Jobs**: Employer can edit, delete, and post job and these operations called as crud operation

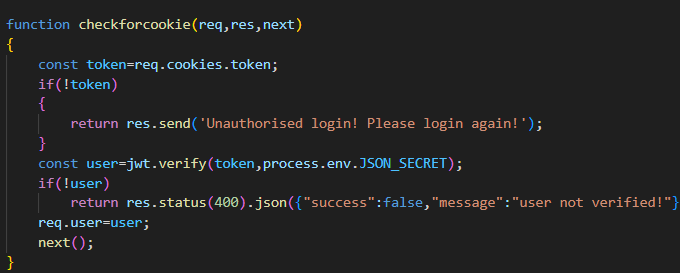
### ****Future Enhancements****

* Will recommend jobs to the employee using AI
* Notifications can be sent to user on real time basis
* Enhance more security level within a web application like using of cipher to encrypt sensitive information and can only decode using secret keys.

## **7. Design Patterns**

* **Model-View-Controller (MVC)**: To meet the industrial standard we used MVC to integrate code and logics to store them in different files so that functions can easily manipulate and readable by developer
* **Authentication Middleware**: Created a middleware like JSON web token verification every time user try to access sensitive page.
* **CRUD Pattern**: Crud Operations has been to edit, delete ,create and update codes.

### ****Middleware for Authentication****



## **8. Deployment**

### ****Deployment Process****

1. Build the React front-end and serve it using NodeJS server integrated express
2. Deploying of web application like frontend and backend on render platform.
3. Render automates the process of deployment by applying web hook whenever github will be updated

### ****Challenges Faced****

1. Debugging issues during API integration.
2. As app is based on monolithic architecture so everything is in single code architecture and if changes have been made to single then it is urge to correct other files as well.

## **9. Learning Experience**

This project enhanced my understanding of:

1. React and Node.js full stack web development.
2. Secure authentication and session management.
3. Scaling and maintainable APIs.
4. On cloud platforms, deployment, and hosting.
5. Testing and deploying with the tool called Postman.

## **10. GitHub Repository**

Both frontend and backend has been uploaded in a single repository on GitHub which will be further deployed on render platform. As render provide as a webhook whenever GitHub will be updated in future it will be automatically updated. The repository URL is: <https://github.com/developerthomasmk/superfast_job_portal.git>

## **Conclusion**

The Job Portal web application effectively satisfies the requirements of both employers and employees by serving as a secured, efficient, user-friendly platform. On using modern day techies such as React, Node, Express for frontend, backend and database as MongoDB, the project guarantees its scalability and maintainability. It provides a security features such as JWT authentication, bcrypt password hashing and CORS policies to ensure that any user data is protected and has trust in the platform. In building this, it not only handles the fundamental job posting and application processes, but delivers note taking for employees and easy integration with Rapid API for external job listing relationships. Using the MVC design pattern, and RESTful APIs, clean code structure is achieved and communication from client to server is efficient.AI based job recommendations and real time notifications are some of the future enhancements that will make the user experience better. Each part of this project represents modern web development practices regarding usability, security, and scalability. In conclusion it can be said that it is an excellent starting point for a thriving and engaging employment portal.