**carrerNest – Finding Job made easy**

**Introduction:**

"Discover limitless career possibilities at carrerNest, a cutting-edge job-seeking platform crafted with the robust MERN stack. Connecting job seekers and employers effortlessly, our secure authentication guarantees a trustworthy experience. Employers can post detailed job listings, while job seekers explore diverse opportunities, applying seamlessly. Elevate your career journey or find top talent—all in one place. Welcome to carrerNest, where opportunities abound, and connections flourish for a brighter professional future."

**Frontend Overview:**

1. **React:**

Library Choice: The frontend is built using React, a declarative and efficient JavaScript library for building user interfaces. React allows for the creation of reusable UI components.

1. **Authentication Components:**

Login and Registration Forms: React components for user authentication, including login and registration forms. These components interact with the backend authentication API.

1. **User Profile Components:**

Profile Display: Components for displaying user information

Edit Profile Form: A form to edit and update the user's profile information.

1. **Publish the Jobs by a Employer:** A employer can publish a job by giving the title ,description and many more details and we store this in database using mongodb.

**5. Routing:** React Router: Implements client-side routing to navigate between different pages and components of the application without refreshing the entire page.

**6.State Management:**

React Context : Manages the state of the application, ensuring efficient data flow between components.

**7. Responsive Design:**

Media Queries: Ensures a responsive design that adapts to various screen sizes and devices, providing a consistent user experience.

1. **Integration with Backend APIs:**

Axios or Fetch: Integration with backend APIs for user authentication, fetching jobs, managing jobs.

1. **Error Handling:**

Display Error Messages: Components for displaying error messages in case of failed operations, such as unsuccessful login attempts or failed job applications

**Backend Overview:**

1. **Node.js and Express.js:**

Node.js serves as the runtime, and Express.js is the web application framework for handling HTTP requests and responses.

1. **MongoDB and Mongoose:**

MongoDB is used as the NoSQL database to store data, while Mongoose serves as the ODM (Object Data Modeling) library to define schemas and interact with MongoDB.

1. **Models :**
2. **User Model:**

* **Schema Definition:**

Defines the structure of the User entity, including fields such as username, email, password (hashed with bcrypt), etc.

* **Mongoose Model:**

Represents the User entity

Manages CRUD operations related to users.

1. **Job Model:**

* **Schema Definition:**

Describes the structure of a job, including fields like Title, description, location, salary, publisher, createdAt, etc.

* **Mongoose Model:**

Represents the Post entity in the application code.

Handles CRUD operations related to posts.

1. **Application Model:**

* **Schema Definition:**

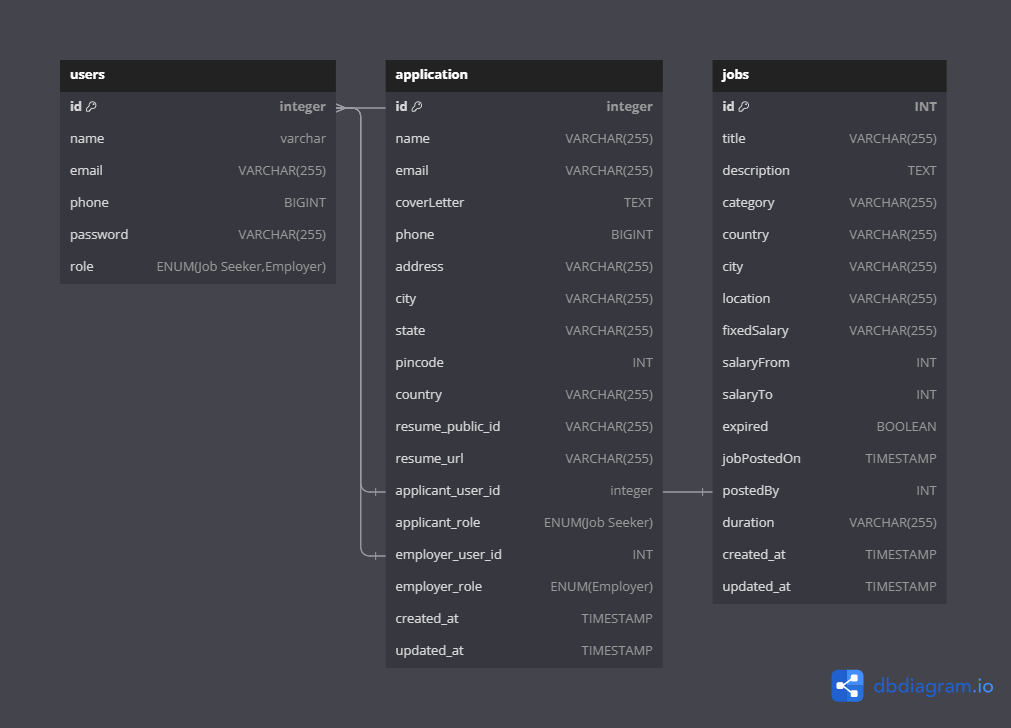
Describes the structure of a Application, including fields like id , name , email ,

Cover letter , phone ,address ,city ,state , pincode ,country , resume , applicant user id , applicant role , employer user id and timestamps

* **Mongoose Model:**

Represents the Post entity

Handles CRUD operations related to application.

****

**JWT Verification is used:**

Middleware to verify and decode JWTs attached to protected routes.

Ensures that only authenticated users have access to certain endpoints.

Controllers:

**Api Design :**

**User routes:**

* user/register -> It register the user on the website as Job seeker

or by a Employer.

* user/login -> It logins the user in the database using authorization and authentication by jwt token
* user/logout -> it Logout the user by clearing the cookies
* user/getUser -> It gets the current user

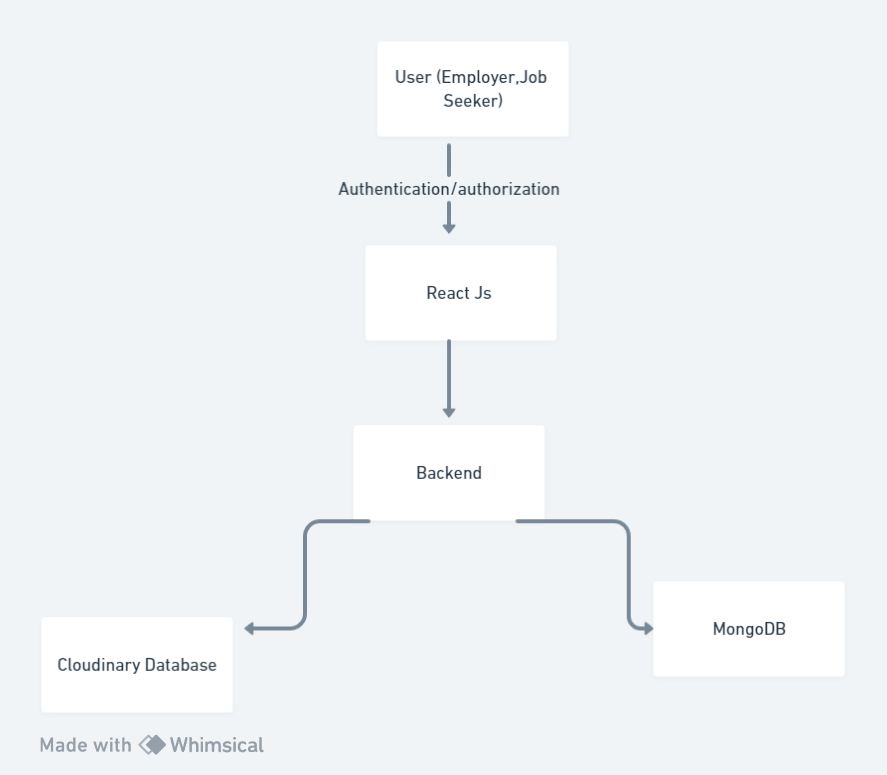
**Job Routes:**

* job/post -> It posts the job by checking if the user is employer
* job/getmyJobs -> It gets the jobs of the Employer
* job/updateJob/:id -> It updates the job by checking the id
* job/deleteJob/:id -> It Deletes the job by checking the id
* job/getalljobs ->It gets all the jobs that are currently now
* job/getSingleJob/:id -> It gets the single job by getting the id of job

**Application routes:**

* application/employer/employer/GetAllapps -> This will get all the applications that has came to the empoyer
* application/jobseeker/apps -> This will get all the applications that the job seeker has posted
* application/postapp -> This will post the application of the jobseeker
* application/jobseeker/getall -> This will get the all the applications of the job seeker
* application/delete/:id -> Job seeker can delete its application by this

**Data flow**



**Deployment :** on vercel