**Job Portal**

**Requirements Document**

**1. Introduction**

The MERN Stack Job Portal Website is a full-stack application designed to connect job seekers with open job positions. Built using MongoDB, Express.js, React, and Node.js, this platform provides users with features such as viewing job listings, applying for jobs, and managing job listings through an admin panel.

**2. User Roles**

The website supports two types of users:

* **Job Seekers:** Individuals seeking employment who can view and apply for job positions.
* **Admins:** Users responsible for managing job listings, including adding, editing, and removing job postings.

**3. Functional Overview**

**3.1 Job Listings:**

* Users can browse through a list of available job positions.
* Each job listing displays essential details such as job title, company name, location, and job description.

**3.2 Job Details:**

* Users can click on a job listing to view detailed information about the job, including qualifications, responsibilities, and application instructions.

**3.3 Apply for Job:**

* Job seekers can apply for a job by submitting their application, which includes uploading a resume.

**3.4 Admin Panel:**

* Admins can log in to access the admin panel.
* The admin panel allows admins to manage job listings, including adding new job postings, editing existing ones, and deleting listings.

**3.5 Authentication**:

* JWT tokens are used for authentication to ensure secure access to user accounts and the admin panel.

**3.6 Image Upload:**

* Cloudinary is integrated for storing job images, allowing admins to upload images to accompany job listings.

**3.7 PDF and CSV:**

* Users can upload PDF files containing their resumes during the application process.
* Admins can export data to CSV format for reporting purposes.

**3.8 Responsive Design:**

* The website is designed to be responsive, ensuring a seamless user experience across various devices and screen sizes.

**Design Document**

1. **Architecture Overview**

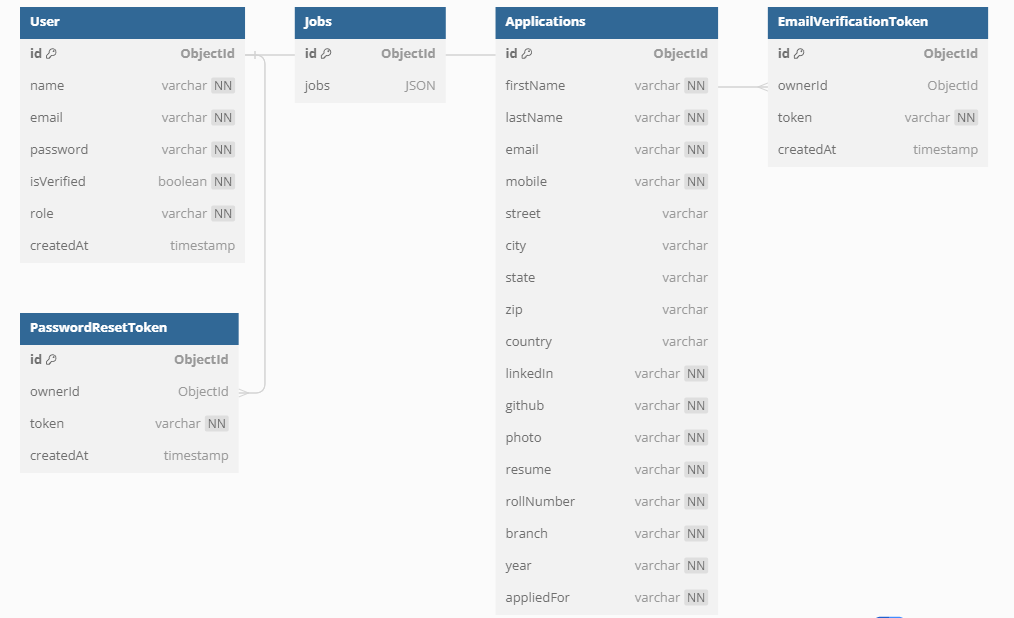
The MERN Stack Job Portal Website follows a modular architecture, with components including:

* Frontend: Developed using React.js and styled using Tailwind CSS and Material UI.
* Backend: Built using Node.js and Express.js, providing RESTful APIs for communication with the frontend and data management.
* Database: MongoDB is used to store user data, job listings, and other relevant information.

1. **Database Design**

The MongoDB database consists of collections for:

* Users: Stores user information including authentication credentials and profile details.
* Jobs: Contains job listings with details such as title, company, location, description, and requirements.



1. **API Design**

The backend provides RESTful APIs for:

* Authentication APIs:

1. POST /api/auth/register: Register a new user.
2. POST /api/auth/login: Authenticate user credentials and generate a JWT token.
3. GET /api/auth/user: Get current user details.

* Authentication APIs:

1. POST /api/auth/register: Register a new user.
2. POST /api/auth/login: Authenticate user credentials and generate a JWT token.
3. GET /api/auth/user: Get current user details.

* Admin Panel APIs:

1. GET /api/admin/jobs: Get all job listings (admin access).
2. GET /api/admin/jobs/:id: Get a specific job listing by ID (admin access).
3. POST /api/admin/jobs: Create a new job listing (admin access).
4. PUT /api/admin/jobs/:id: Update a specific job listing by ID (admin access).
5. DELETE /api/admin/jobs/:id: Delete a specific job listing by ID (admin access).
6. **Frontend Design**

The frontend is designed using React.js and styled with Tailwind CSS and Material UI. Key components include:

* Job Listings Page: Displays available job positions with relevant details.
* Job Details Page: Shows detailed information about a selected job listing.
* Application Form: Allows users to apply for jobs by submitting their application, including resume upload.
* Admin Panel: Provides access to admin features for managing job listings.

1. **Deployment**

The website can be deployed on cloud platforms such as AWS, Google Cloud Platform, or Heroku. Frontend hosting services like Netlify or Vercel can be used for hosting the client-side application, while the backend can be deployed on platforms supporting Node.js applications.

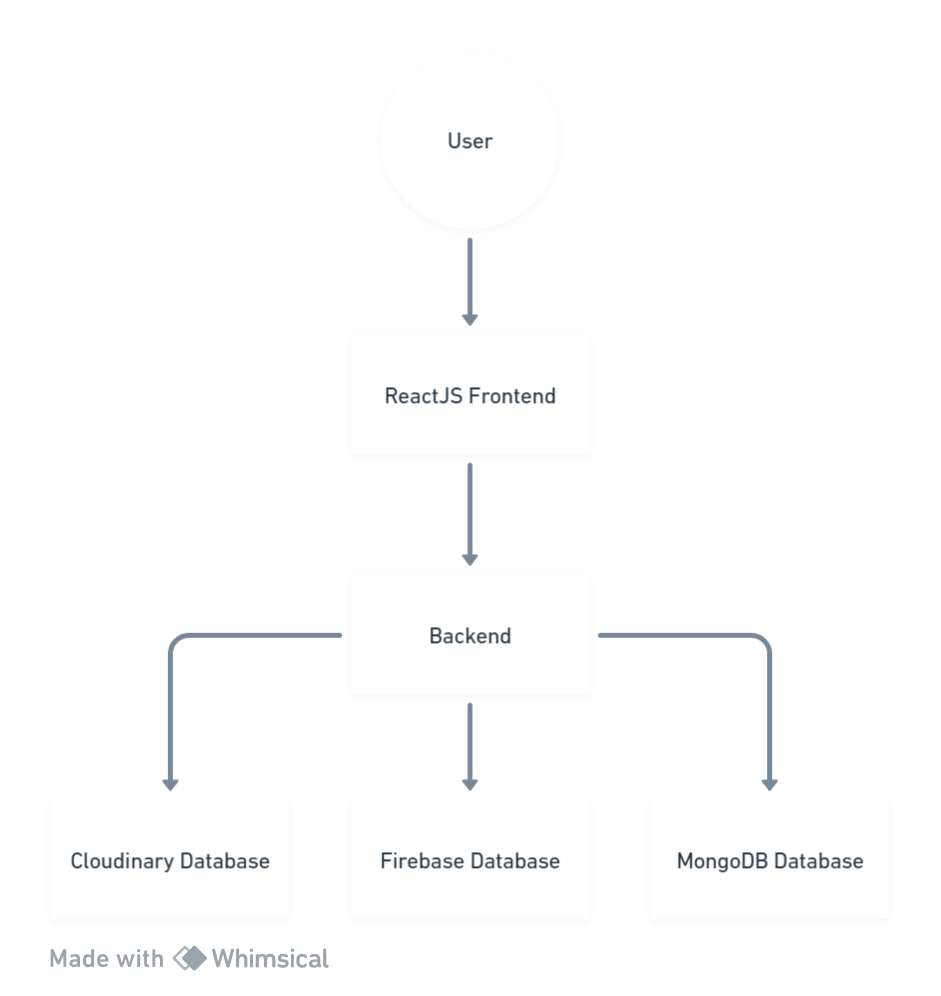
1. **Security**

Security measures include encryption of sensitive data, robust authentication using JWT tokens, input validation, and protection against common web vulnerabilities.

1. **Maintenance and Support**

Regular maintenance should be performed to keep the website updated with the latest security patches and technologies. A support system should be in place to address user inquiries and bug reports promptly.

1. **Data FLoe**

****

1. **Conclusion**

The MERN Stack Job Portal Website aims to provide a user-friendly platform for job seekers to find and apply for job positions efficiently. By following the outlined design principles and implementing the suggested features, the website aims to enhance the job search experience for users.