**Company Website & Job Application Portal (MERN Stack)**

**Introduction**

This document outlines the architectural and technical specifications for the development of a dynamic and modern web application built using the MERN stack—**MongoDB**, **Express.js**, **React.js**, and **Node.js**. The goal of the project is to create a professional website for the organization that not only highlights the services provided but also includes an integrated **job application portal** for publishing job openings and accepting candidate applications online.

The application is designed with scalability, security, and usability in mind, catering to both public visitors (applicants) and internal administrators (recruiters or HR personnel).

A screenshot of a computer

AI-generated content may be incorrect.

**Architecture Diagram**

**Features**

**Informational Website Pages**: Including landing page, services overview, contact information, and about the company.

**Careers Portal**:

* Public listing of job openings
* Filtering and searching for roles
* Detailed view for each job

**Job Application Module**:

* Resume and details submission form
* Confirmation and success pages

**Admin Dashboard**:

* Secure login for internal users
* Add, update, or delete job postings
* View and manage applications

**Resume Upload and Storage**:

* Secure file upload
* Resumes stored in cloud (AWS S3 or Cloudinary)
* File URLs saved in the backend database

**Underlying Tech Stack**

**Frontend (Client-Side)**

The frontend is built using React.js, providing a dynamic and responsive user interface.

* React.js: Enables a single-page application (SPA) with reusable components and efficient rendering.
* React Router DOM: Facilitates client-side navigation between views like Home, Services, Careers, Apply, and Admin.
* Axios: Used for API requests from the frontend to the backend server.
* Tailwind CSS / Material UI: Provides a utility-first or component-based design system to ensure clean, responsive, and professional UI styling.
* Formik + Yup (optional): Handles form validations, especially for the job application forms.
* Protected Routes: Used to restrict access to admin features unless the user is authenticated.

**Backend (Server-Side)**

The backend uses Node.js and Express.js, providing a RESTful API for handling business logic and integrations.

* Node.js: JavaScript runtime for server-side scripting.
* Express.js: Minimalist web framework for defining HTTP routes and middleware.
* JWT (JSON Web Tokens): Implements secure authentication and session management for admin users.
* Multer: Middleware to handle multipart/form-data (used for uploading resumes).
* Bcrypt: Used for hashing and comparing passwords for secure login handling.
* CORS Middleware: Ensures proper cross-origin communication between frontend and backend during deployment.
* Helmet (optional): Provides security hardening for HTTP headers.

**Database (MongoDB with Mongoose)**

Data is persisted using MongoDB Atlas, a fully-managed NoSQL cloud database, accessed and structured via the Mongoose ORM.

* Collections and Schemas:
  + jobs: Stores job metadata such as title, description, location, and requirements.
  + applications: Captures applicant data including name, email, resume link, and applied job ID.
  + users: Contains admin account credentials and role info.
* Mongoose provides:
  + Schema enforcement and data modeling
  + Input validation at the schema level
  + Built-in query methods and population handling

**File Storage (Resume Uploads)**

Resumes uploaded via the job application form are not stored locally. Instead, a cloud-based object storage system is used.

* Options:
  + AWS S3: Integrates using the AWS SDK. Resumes are uploaded to a secured bucket with access policies.
  + Cloudinary: A simpler alternative that offers automatic file management and secure URLs.
* Integration Flow:
  + Resume is uploaded via the frontend form
  + Backend handles the file using multer and uploads to cloud storage
  + Cloud returns a secure URL
  + The URL is saved in the applications collection in MongoDB

**Deployment**

Each tier of the stack can be deployed independently using cloud platforms:

* Frontend:
  + Platforms: Vercel, Netlify, or Firebase Hosting
  + Build using npm run build and deploy static files
* Backend:
  + Platforms: Render, Heroku, Railway, or AWS EC2
  + .env used for environment variables (API keys, DB URLs)
* Database:
  + MongoDB Atlas (cloud database with access control and IP whitelisting)
* File Storage:
  + AWS S3 (with public-read or signed URL policies)
  + Cloudinary (with pre-configured upload presets and API keys)

**Security Practices**

* Passwords are hashed before storage (bcrypt)
* Admin APIs are protected using JWT-based authentication
* File uploads are sanitized and limited in size
* CORS policy is restricted to only known frontend origins
* HTTPS enforced for secure data transfer