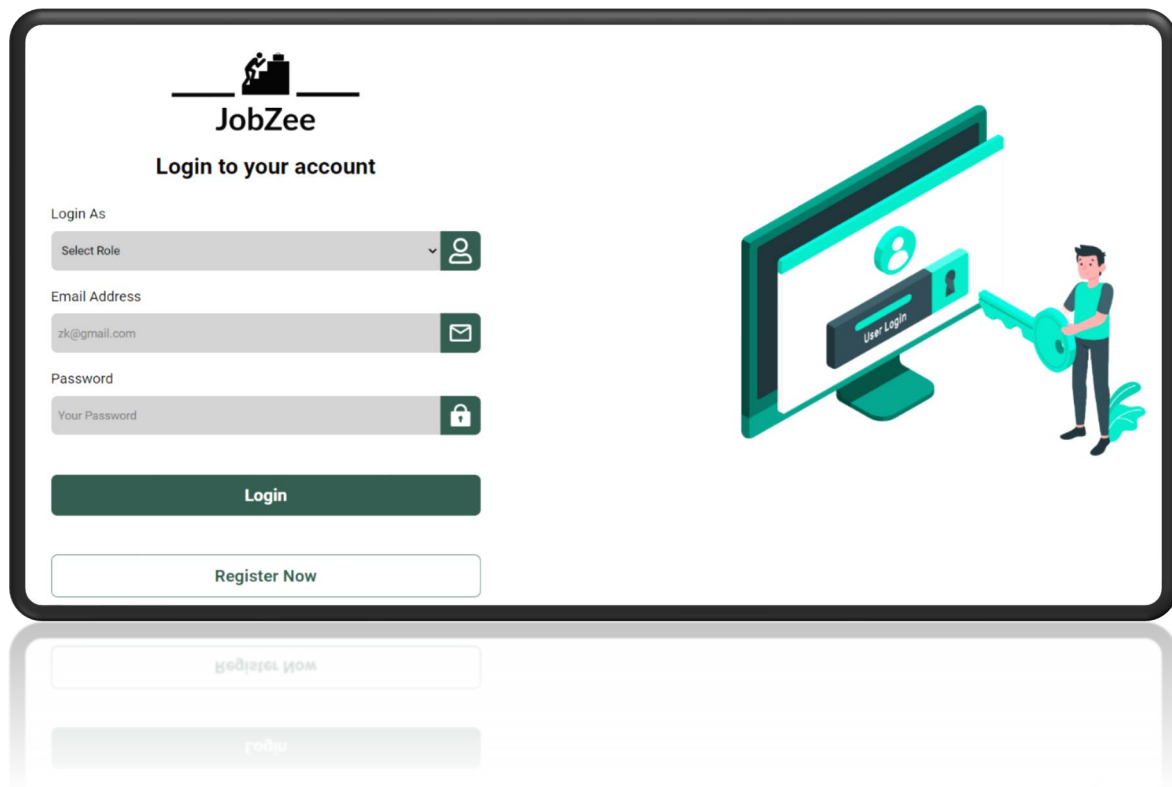


# DATABASE LAB PROJECT DOCUMENTATION:



The image displays a web application interface for 'JobZee'. At the top left, there is a logo consisting of a stylized person climbing a ladder next to the text 'JobZee'. Below the logo, the text 'Login to your account' is centered. The interface features a 'Login As' section with a dropdown menu labeled 'Select Role' and a user icon. Below this are three input fields: 'Email Address' with the placeholder 'zk@gmail.com', 'Password' with the placeholder 'Your Password', and a 'Login' button. A 'Register Now' button is located at the bottom. To the right of the login form is an illustration of a person holding a large key next to a computer monitor displaying a 'User Login' screen. The entire interface is enclosed in a rounded rectangle with a dark border, and a reflection of the interface is visible below it.

JobZee

Login to your account

Login As

Select Role

Email Address

zk@gmail.com

Password

Your Password

Login

Register Now

## GROUP MEMBERS:

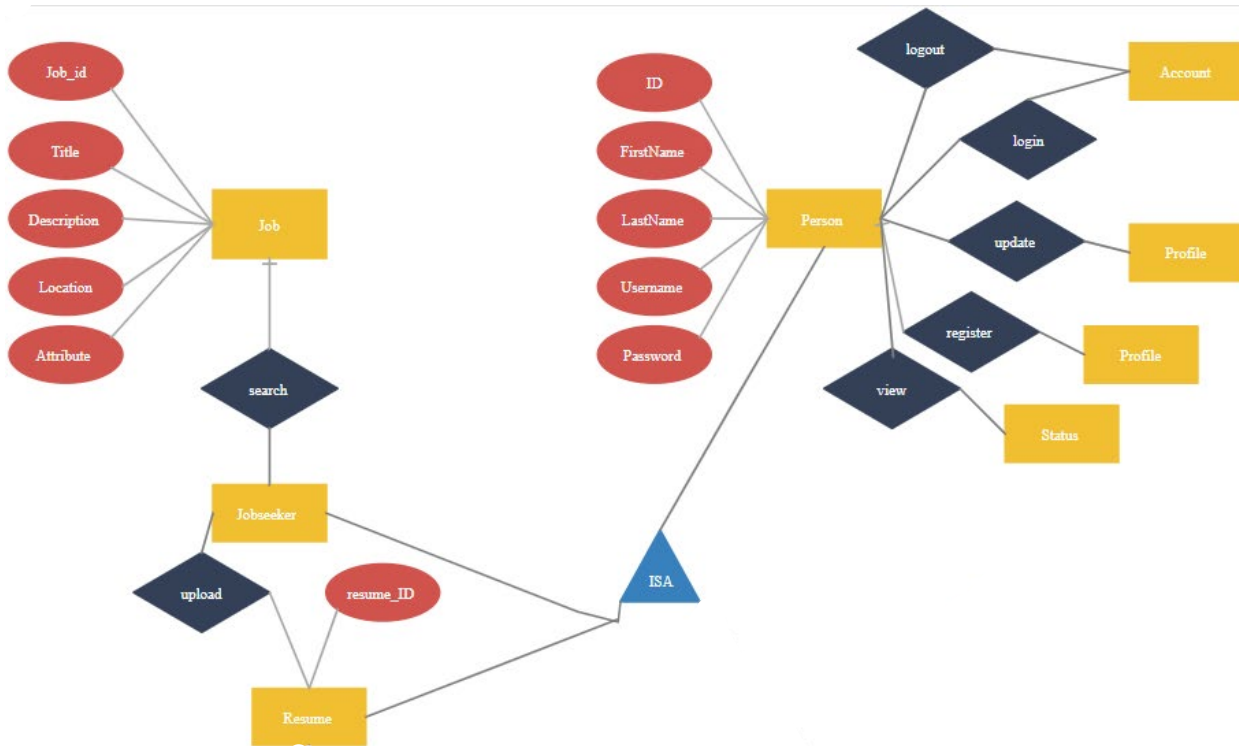
Muhammad Hassan Jamshaid (231574)

Mujeeb Ahmad (231700)

# Introduction:

Welcome to the documentation for the MERN Stack Job Seeking Web Application. This project is designed to connect job seekers with potential employers, providing a platform for job listings, applications, and management. The application leverages the powerful combination of MongoDB, Express.js, React.js, and Node.js to deliver a seamless user experience for both job seekers and employers.

## ERD Diagram:



## Prerequisites:

To run this project locally, you will need to have the following software installed:

1. **Node.js**: JavaScript runtime built on Chrome's V8 JavaScript engine.
2. **npm (Node Package Manager)**: Comes with Node.js, used for managing dependencies.
3. **MongoDB**: NoSQL database used for storing application data.

# Backend (Node.js with Express.js)

**express:** Fast, unopinionated, minimalist web framework for Node.js.

**mongoose:** Elegant MongoDB object modeling for Node.js.

**dotenv:** Module to load environment variables from a .env file.

**bcryptjs:** Library to hash passwords.

**jsonwebtoken:** Implementation of JSON Web Tokens.

**cookie-parser:** Parse Cookie header and populate req.cookies with an object keyed by the cookie names.

**cloudinary:** Library to manage image and video files in Cloudinary.

# Frontend (React.js)

**react:** JavaScript library for building user interfaces.

**react-dom:** Entry point of the DOM-related rendering paths.

**react-router-dom:** DOM bindings for React Router.

**axios:** Promise based HTTP client for the browser and node.js.

**redux:** Predictable state container for JavaScript apps.

**react-redux:** Official React bindings for Redux.

**redux-thunk:** Thunk middleware for Redux.

# Development Tools

**nodemon:** Utility that will monitor for any changes in your source and automatically restart your server.

## Database

This project uses MongoDB as its database. MongoDB is a NoSQL database which stores data in JSON-like documents with flexible schemas. This allows for easy and fast storage and retrieval of data, making it a great choice for web applications.

# Setting Up the Database

1. **Install MongoDB:** Follow the [official MongoDB installation guide) to install MongoDB on your machine.

2. **Start MongoDB:** Run the following command to start MongoDB.

```
mongosh
```

## Connecting to the Database

The connection to the MongoDB database is handled using Mongoose, an ODM (Object Data Modeling) library for MongoDB and Node.js. Mongoose provides a straightforward, schema-based solution to model your application data.

## MongoDB Compass Interface:

The screenshot displays the MongoDB Compass interface. The top section, titled "New Connection", allows users to connect to a MongoDB deployment. It includes a URI field with the value "mongodb://localhost:27017/" and a "Connect" button. Below this, there are "Advanced Connection Options" and "Save" and "Save & Connect" buttons.

The bottom section shows the MongoDB Compass interface for the "MERN\_STACK\_JOB\_SEEKING" database. It lists three collections: "applications", "jobs", and "users". Each collection has a table with columns for "Storage size", "Documents", "Avg. document size", "Indexes", and "Total index size".

Collection	Storage size	Documents	Avg. document size	Indexes	Total index size
applications	4.10 kB	0	0 B	1	4.10 kB
jobs	20.48 kB	2	373.00 B	1	36.86 kB
users	20.48 kB	2	206.00 B	1	36.86 kB

# Sample Data:

MERN\_STACK\_JOB\_SEEKING

applications

jobs

users

admin

config

local

ADD DATA

EXPORT DATA

UPDATE

DELETE

\_id: ObjectId('664592238ae11670d3148f22')

title: "Dexter code software"

description: "hassanjamashiadsibtainhaidermujeebahmedshozababbasumantariq"

category: "samplecategory"

country: "Spain"

city: "Isl"

location: "islamabadpakistanairuniversityopp.bahriaunisectorf8"

salaryFrom: 1244

salaryTo: 124567

expired: false

postedBy: ObjectId('66458da58ae11670d3148f14')

jobPostedOn: 2024-05-16T04:57:07.863+00:00

\_\_v: 0

\_id: ObjectId('664ac26a36fc446ebc02c402')

title: "Flutter Developer"

description: "We are looking for a professional Flutter developer with good programm..."

category: "Mobile App Development"

country: "Pakistan"

city: "Islamabad"

location: "Tech-Solutions, Sector F-10, Islamabad"

salaryFrom: 150000

salaryTo: 250000

expired: false

postedBy: ObjectId('664ac1312e6996013310305a')

jobPostedOn: 2024-05-20T03:24:26.578+00:00

\_\_v: 0

MERN\_STACK\_JOB\_SEEKING

applications

jobs

users

admin

config

local

ADD DATA

EXPORT DATA

UPDATE

DELETE

\_id: ObjectId('664ac1312e6996013310305a')

name: "Hassan"

email: "hassan@gmail.com"

phone: 3123456789

password: "\$2b\$10\$P90kQXkHv8fk4vEPmZzBEeuEvmb7bBaSmH/t8ykHpF0Wn9mrYnq20"

role: "Employer"

createdAt: 2024-05-20T03:19:13.219+00:00

\_\_v: 0

\_id: ObjectId('664acafecbe67b241ec3f8ef')

name: "Hassan"

email: "hassan123@gmail.com"

phone: 3123456789

password: "\$2b\$10\$y8yWLn6oz8CJ3/WA0/Sdju/rLSx4905UMBvUxB7ZyB43y0bBwG7hK"

role: "Job Seeker"

createdAt: 2024-05-20T04:01:02.983+00:00

\_\_v: 0

## Project Structure:

The project is structured as follows:

- backend:** Contains the server-side code (Express.js).
- controllers:** Functions to handle routes logic.
- models:** Mongoose schemas and models.

**routes:** Express route definitions.

**middlewares:** Custom middleware functions.

**config:** Configuration files (e.g., database connection).

**frontend:** Contains the client-side code (React.js).

**components:** React components.

**pages:** Different pages of the application.

**redux:** Redux store setup and slices.

## How to Run the Project

### Install Dependencies

#### For the backend:

```
cd backend
```

```
npm install ....
```

#### For the frontend:

```
cd frontend
```

```
npm install ....
```

## Set Up Environment Variables

Create a `.env` file in the root of your project and add the following variables:

```
MONGO_URI
```

```
JWT_SECRET
```

```
CLOUDINARY_NAME
```

```
CLOUDINARY_API_KEY
```

```
CLOUDINARY_API_SECRET
```

### Run the Server:

**Start the backend server:**

```
cd backend
```

```
npm run dev
```

### **Start the frontend server:**

```
cd frontend
```

```
npm run dev
```

### **Access the Application**

Open your browser and navigate to ``http://localhost:5173`` to access the application.

### **Conclusion:**

This documentation provides a comprehensive overview of the MERN Stack Job Seeking Web Application, outlining the project's objectives, prerequisites, and setup instructions. By following the steps detailed above, you should be able to successfully set up and run the application on your local machine, gaining a clear understanding of its core functionalities and architecture.

The project leverages the powerful MERN stack—MongoDB, Express.js, React.js, and Node.js—to deliver a robust and scalable solution for job seekers and employers. The backend, built with Node.js and Express.js, handles data management, user authentication, and business logic. It connects seamlessly to MongoDB, a flexible and scalable NoSQL database, ensuring efficient data storage and retrieval. The use of Mongoose as an ODM simplifies interactions with MongoDB, providing a schema-based solution to model application data.

On the frontend, React.js offers a dynamic and responsive user interface, ensuring a smooth user experience. The integration of Redux for state management and Axios for HTTP requests ensures that the application's state is predictable and interactions with the backend are efficient.

The project also incorporates essential security features, such as password hashing with bcrypt.js and token-based authentication with JSON Web Tokens (JWT). This ensures that user data is protected and secure. The inclusion of Cloudinary for managing resume uploads further enhances the application's functionality, allowing users to upload and manage their resumes with ease.

To facilitate development and debugging, tools like nodemon are used to automatically restart the server on code changes, streamlining the development workflow.

By understanding the project's structure and the technologies involved, you can appreciate the modular and maintainable approach taken in its development. Each

component, from user authentication to job application management, is designed to be reusable and extensible, allowing for future enhancements and scalability.