**JOB PORTAL WEBSITE**

**Introduction :**

Welcome to our Job Portal Website! Our platform is dedicated to job seekers to view the job vacancies and apply for their job. Our job portal website offers a user-friendly interface to help you secure the perfect job.

With an intuitive design, our job portal website allows you to effortlessly explore various jobs, view their company name, job role, required skills, job type, experience, and make registrations with just a few clicks. You can browse through a diverse range of job availability.

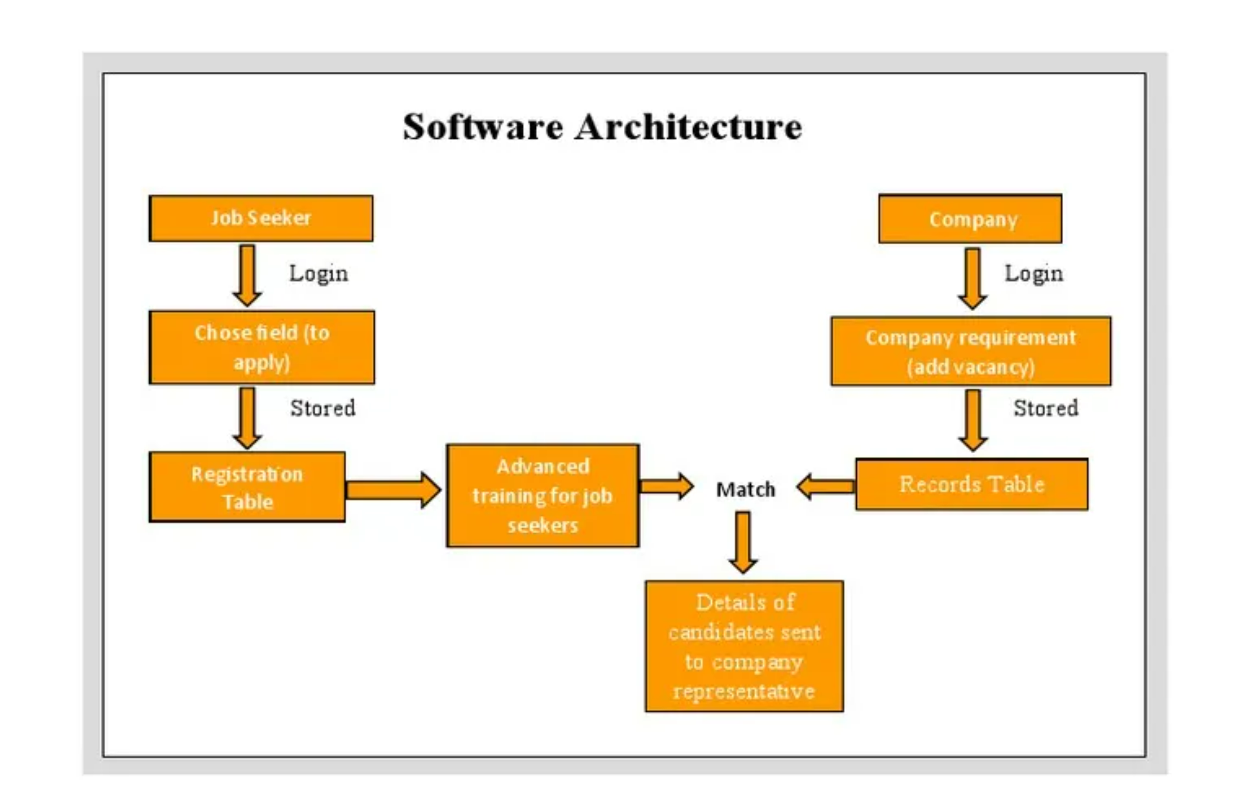
This project is Online Job Portal where Companies can post and view jobs. A Job seeker can view and apply the jobs. Both Job Seeker and Company can update their profile details.

The system is an online application that can be accessed throughout the organization and outside as well with proper login provided. This system can be used as an online job portal for the placement department of the organization to manage the student information with regards to placement.

Job Portal is developed for creating an interactive job vacancy for candidates. On the whole the objective of the project is to enable jobseekers to place their resumes and companies to publish their resume, search for jobs, view personal job listings.

It will provide various companies to place their vacancy profile on the site and also have an option to search candidate resumes. Apart from this there will be an admin module for the customer to make changes to the database content. It consists of Job Seeker and Job Provider.

**TECHNICAL ARCHITECTURE:**

****

The technical architecture of a job portal webite typically involves a client-server model, where the frontend represents the frontend and the backend serves as the server. The frontend is responsible for user interface, interaction, and presentation, while the backend handles data storage, business logic, and integration with external services like databases. Communication between the frontend and backend is typically facilitated through APIs, enabling seamless data exchange and functionality.

### ER DIAGRAM:

### 

The Entity-Relationship (ER) diagram for a job portal website visually represents the relationships between different entities involved in the system, such as users, admin, restaurants, and reviews. It illustrates how these entities are related to each other and helps in understanding the overall database structure and data flow within the application.

### KEY FEATURES:

**Job Listings:** The job portal website offers an extensive job catalog with various categories and subcategories. Job seekers can easily search, browse, and filter job listings based on their preferences and skills.

**Job Application and Resume upload:** The website includes a job application feature where users can apply for jobs directly through the platform. Additionally, job seekers can upload their resumes, making it convenient for employers to review applicant profiles.

**Reviews and Ratings:** Users, both employers and job seekers, can provide feedback and ratings. This feature helps in establishing a trustworthy community and aids employers in making informed hiring decisions.

**Job status Tracking:** Job seekers can track the status of their job applications in real-time. Updates on application processing, interview invitations, and job offers provide transparency and keep users informed.

**Admin Dashboard:** Administrators have access to a comprehensive dashboard to manage job listings, applicant profiles, employer information, and overall website performance. It provides insights into the number of job postings, applicant demographics, and other relevant analytics.

**Development Resources:** The portal may provide additional resources, such as articles, webinars, or training opportunities, to help job seekers enhance their skills and stay competitive in the job market.

**Search and Filtering:** The website may offer premium services for employers, such as featured job listings or enhanced visibility. Secure payment options ensure smooth transactions for these premium services.

**PRE REQUISITES:**

To develop a full-stack table booking app using AngularJS, Node.js, and MongoDB, there are several prerequisites you should consider. Here are the key prerequisites for developing such an application:

**Node.js and npm:** Install Node.js, which includes npm (Node Package Manager), on your development machine. Node.js is required to run JavaScript on the server side.

* Download: <https://nodejs.org/en/download/>
* Installation instructions: <https://nodejs.org/en/download/package-manager/>

**MongoDB:** Set up a MongoDB database to store hotel and booking information. Install MongoDB locally or use a cloud-based MongoDB service.

* Download: <https://www.mongodb.com/try/download/community>
* Installation instructions: <https://docs.mongodb.com/manual/installation/>

**Express.js:** Express.js is a web application framework for Node.js. Install Express.js to handle server-side routing, middleware, and API development.

* Installation: Open your command prompt or terminal and run the following command: **npm install express**

**Angular:** Angular is a JavaScript framework for building client-side applications. Install Angular CLI (Command Line Interface) globally to create and manage your Angular project.

Install Angular CLI:

* Angular provides a command-line interface (CLI) tool that helps with project setup and development.
* Install the Angular CLI globally by running the following command:

npm install -g @angular/cli

**Verify the Angular CLI installation:**

* Run the following command to verify that the Angular CLI is installed correctly: ng version You should see the version of the Angular CLI printed in the terminal if the installation was successful.

Create a new Angular project:

* Choose or create a directory where you want to set up your Angular project.
* Open your terminal or command prompt.
* Navigate to the selected directory using the cd command.
* Create a new Angular project by running the following command: ng new cl Wait for the project to be created:
* The Angular CLI will generate the basic project structure and install the necessary dependencies.

**Navigate into the project directory:**

* After the project creation is complete, navigate into the project directory by running the following command**: cd frontend**

###### Start the development server:

* To launch the development server and see your Angular app in the browser, run the following command: **ng serve / npm start**
* The Angular CLI will compile your app and start the development server.
* Open your web browser and navigate to http://localhost:4200 to see your Angular app running.

You have successfully set up Angular on your machine and created a new Angular project. You can now start building your app by modifying the generated project files in the src directory.

Please note that these instructions provide a basic setup for Angular. You can explore more ad- vanced configurations and features by referring to the official Angular documentation: https://angular.io

**HTML, CSS, and JavaScript:** Basic knowledge of HTML for creating the structure of your app, CSS for styling, and JavaScript for client-side interactivity is essential.

**Database Connectivity:** Use a MongoDB driver or an Object-Document Mapping (ODM) library like Mongoose to connect your Node.js server with the MongoDB database and perform CRUD (Create, Read, Update, Delete) operations.

**Front-end Framework:** Utilize Angular to build the user-facing part of the application, including products listings, booking forms, and user interfaces for the admin dashboard.

**Version Control**: Use Git for version control, enabling collaboration and tracking changes throughout the development process. Platforms like GitHub or Bitbucket can host your repository.

* Git: Download and installation instructions can be found at: [https://git-](https://git-scm.com/downloads) [scm.com/downloads](https://git-scm.com/downloads)

**Development Environment:** Choose a code editor or Integrated Development Environment (IDE) that suits your preferences, such as Visual Studio Code, Sublime Text, or WebStorm.

* Visual Studio Code: Download from<https://code.visualstudio.com/download>
* Sublime Text: Download from <https://www.sublimetext.com/download>
* WebStorm: Download from <https://www.jetbrains.com/webstorm/download>

To Connect the Database with Node JS go through the below provided link:

* Link: [https://www.section.io/engineering-education/nodejs- mongoosejs-mongodb/](https://www.section.io/engineering-education/nodejs-%20mongoosejs-mongodb/)

###### **To run the existing Table Booking App project downloaded from github: Follow below steps:**

1. **Clone the Repository:**
   * Open your terminal or command prompt.
   * Navigate to the directory where you want to store the e-commerce app.
   * Execute the following command to clone the repository:

**git clone: https://github.com/Santhoshhkumar/Job\_Portal**

###### **Install Dependencies:**

* + Navigate into the cloned repository directory:

cd BookingApp

* + Install the required dependencies by running the following command:

npm install

###### **Start the Development Server:**

* + To start the development server, execute the following command:

npm start

###### **Access the App:**

* + You should see the e-commerce app's homepage, indicating that the installation and setup were successful.

**Project Repository Link: https://github.com/Santhoshhkumar/Job\_Portal**

Congratulations! You have successfully installed and set up the job portal website on your local machine. You can now proceed with further customization, development, and testing as needed.

###### **Roles and Responsibilities:**

The project has two types of users –Customer and Admin. The roles and responsibilities of these two types of users can be inferred from the API endpoints defined in the code. Here is a summary:

**JOB SEEKERS:**

1. Create an account and log in to the system using their email and password.
2. Browse and search for jobs available on the platform.
3. View detailed job information, including description and availability.
4. Apply for jobs.
5. Proceed to application page.
6. Confirmation of successful application.
7. Track application status.
8. Manage their profile information.
9. Provide feedback and reviews.
10. Access customer support.

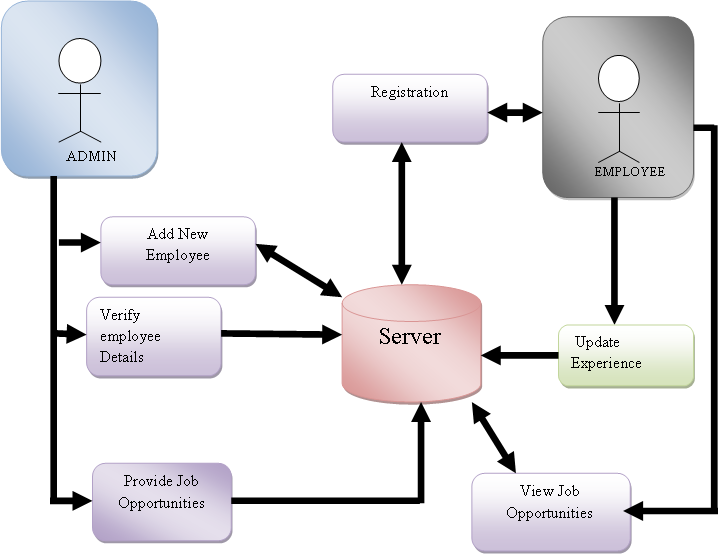
**ADMIN:**

1. Manage and monitor the job listings.
2. Approve and update job listings.
3. Monitor and moderate job listings.
4. Handle candidate disputes.
5. Manage user accounts and profiles.
6. Analyze platform performance .
7. Implement and enforce platform policies, terms of service, and privacy regulations.
8. Continuously improve the platform's functionality, user experience, and security measures.

These roles and responsibilities are aimed at ensuring a smooth and efficient operation of the jop portal, providing a seamless experience for Job seekers and administrators.

**Admin and User Flow:**

**Admin:**

****

UserFlow:

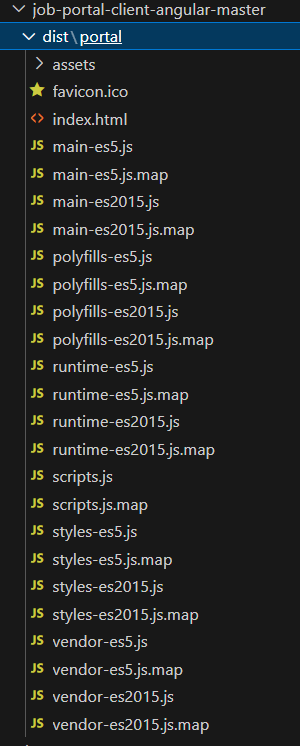
User:

A diagram of a job portal

Description automatically generated

**PROJECT STRUCTURE:**

**Front End:**

****

**A screenshot of a computer

Description automatically generated**

This structure assumes an Angular app and follows a modular approach. Here's a brief explanation of the main directories and files:

* + src/app/components: Contains components related to the customer app, such as register, login, job, dashboard, job details and more.
  + src/app/modules: Contains modules for different sections of the app. In this case, the admin module is included with its own set of components.
  + src/app/app-routing.module.ts: Defines the routing configuration for the app, specifying which components should be loaded for each route.
  + src/app/app.component.ts, src/app/app.component.html, `src.

**Back End:**

**A screen shot of a computer

Description automatically generated**

###### **Project Flow:**

**Milestone 1: Project Setup and Configuration:**

1. Install required tools and software:
   * Node.js.
   * MongoDB.
   * Angular CLI.
2. Create project folders and files:
   * Frontend folders.
   * Backend folders.

###### Milestone 2: Backend Development:

Setup express server:

* + Install express.
  + Create app.js file.
  + Define API’s

Configure MongoDB:

* + Install Mongoose.
  + Create database connection.
  + Create Models.

Implement API end points:

* + Implement CRUD operations.
  + Test API endpoints.

**Backend:**

1. Set Up Project Structure:
   * Create a new directory for your project and set up a package.json file using npm init command.
   * Install necessary dependencies such as Express.js, Mongoose, and other required packages.
2. Database Configuration:
   * Set up a MongoDB database either locally or using a cloud-based MongoDB service like MongoDB Atlas.
   * Create a database and define the necessary collections for users relevant data.
3. Create Express.js Server:
   * Set up an Express.js server to handle HTTP requests and serve API endpoints.
   * Configure middleware such as body-parser for parsing request bodies and cors for handling cross-origin requests.
4. Define API Routes:
   * Create separate route files for different API functionalities such as hotels, users, bookings, and authentication.
   * Define the necessary routes for listing jobs, handling user registration and login, managing bookings, etc.
   * Implement route handlers using Express.js to handle requests and interact with the database.

**Implement Data Models:**

* + Define Mongoose schemas for the different data entities like jobs, users, and applications.
  + Create corresponding Mongoose models to interact with the MongoDB database.
  + Implement CRUD operations (Create, Read, Update, Delete) for each model to perform database operations.

**API Design and Development:**

* + Identify the necessary functionality and data required by the frontend.
  + Design a set of RESTful APIs using a framework like Express.js or Django REST Framework.
  + Define API endpoints for user management, product catalog, shopping cart, order management, payment gateway integration, shipping integration, etc.
  + Implement the API routes, controllers, and data models to handle the corresponding operations.
  + Ensure that the APIs follow best practices, are secure, and provide appropriate re- sponses.

**User Management and Authentication:**

* + Implement user registration and login functionality.
  + Choose an authentication mechanism like session-based authentication or token-based authentication (e.g., JWT).
  + Store and hash user credentials securely.
  + Implement middleware to authenticate API requests and authorize access to protected routes

**Catalog and Inventory Management:**

* + Design the database schema to store product details, pricing, availability, and invento- ry levels.
  + Create APIs to retrieve product information, update inventory quantities, and handle search and filtering.
  + Implement validations to ensure data integrity and consistency.

**Booking Management:**

* + Design the database to store customer details and booking information.
  + Create APIs to handle cart operations like adding hotels, modifying quantities, and booking tables.

**Database Integration:**

* + Choose a suitable database technology (e.g., MySQL, PostgreSQL, MongoDB) based on your application's requirements.
  + Design the database schema to efficiently store and retrieve e-commerce data.
  + Establish a connection to the database and handle data persistence and retrieval.

**External Service Integration:**

* + Identify third-party services like email service providers, analytics services, or CRM systems that are required for your application.
  + Utilize the APIs or SDKs provided by these services to exchange data and perform necessary operations.
  + Implement the integration logic to send order confirmations, track user behavior, or manage customer relationships.

**Security and Data Protection:**

* + Apply appropriate security measures like encryption techniques for secure data transmission and storage.
  + Implement input validation and sanitization to prevent common security vulnerabili- ties.
  + Implement access control to ensure authorized access to sensitive data.

**Error Handling and Logging:**

* + Implement error handling mechanisms to handle exceptions and provide meaningful error messages to the frontend.
  + Use logging frameworks to record application logs for monitoring and troubleshooting purposes.