



Page left blank intentionally

**INDEX**

[**1**](#_gjdgxs) **PURPOSE** 2

[**2**](#_30j0zll) **PROJECT SCOPE** 2

[**3**](#_1fob9te) **SYSTEM OVERVIEW** 2

[**4**](#_3znysh7) **DESIGN CONSIDERATIONS** 2

[4.1](#_2et92p0) Requirements 3

[4.2](#_tyjcwt) Assumptions 3

[4.3](#_3dy6vkm) Dependencies 3

[**5**](#_1t3h5sf) **SYSTEM ARCHITECTURE** 3

[5.1](#_4d34og8) Architectural Strategies 4

[5.2](#_2s8eyo1) Structure & Relationships 4

[**6**](#_17dp8vu) **DETAILED DESCRIPTION OF COMPONENTS** 4

[**7**](#_3rdcrjn) **INTEGRATION** 5

[**8**](#_26in1rg) **APPENDICES** 1

[8.1](#_lnxbz9) Appendix A – Detailed Description of Components 1

**General Instructions for using the Live Project POC Document**

* This template and the subsequent document created using this template is a confidential document and is the intellectual property of Cloud Counselage Pvt. Ltd. Circulating it outside of the organisation without the consent of Cloud Counselage Pvt. Ltd. is the breach of company policies and will lead to legal actions
* The Design Specification of a software forms the basis of development of software
* The **text between inequality (< >) is to be replaced** by relevant text
* Please **remove the yellow highlight on the Text** between the inequality (< >). This is done to help you notice the text to be changed/replaced
* The text in *italics* highlighted in grey is just for reference and should be removed after adding the relevant text

# **PURPOSE**

The purpose of the "Jobsearch" application is to facilitate a seamless and efficient connection between employers and job seekers in the recruitment process. By leveraging modern technologies such as React, Express, MongoDB Atlas, and Cloudinary, the application simplifies candidate sourcing and job hunting. Its user-friendly interface allows employers to easily find suitable candidates, while job seekers can search for relevant job opportunities tailored to their skills and preferences.

# **PROJECT SCOPE**

The scope of the JobSearch includes its distinct features, its benefits, and its limitations. The system's distinct features allow it to To make easy recruitment process by using MERN framework. The system enables the user to Easily appoint and recruit applicants.

# **SYSTEM OVERVIEW**

This section will provide an outline of the various components and subsystems of JobSearch

The "Jobsearch" project is a Fullstack MERN (MongoDB, Express.js, React, Node.js) application designed to streamline the recruitment process for both employers and job seekers. Leveraging React for the frontend and Express.js for the backend, along with MongoDB Atlas for database management and Cloudinary for cloud storage, the system provides a comprehensive solution for efficient candidate sourcing and job searching.

Employers can easily search for and find suitable candidates based on their requirements, while job seekers can search for relevant job opportunities tailored to their skills and preferences. The intuitive user interface and seamless integration of cloud storage technology make the recruitment process straightforward and hassle-free for all stakeholders involved. Overall, "Jobsearch" simplifies the recruitment process, making it easier for employers to find the right candidates and for job seekers to find suitable employment opportunities.

# **DESIGN CONSIDERATIONS**

This section describes requirements, assumptions and dependencies to be addressed to devise a complete design solution.

## Requirements

* **User Authentication:** Implement secure user authentication for both employers and job seekers.
* **Job Posting:** Allow employers to post job listings with detailed descriptions, including job title, responsibilities, requirements, and location.
* **Candidate Search**: Provide a search functionality for employers to easily find and filter potential candidates based on various criteria such as skills, experience, and location.
* **Job Search**: Enable job seekers to search for relevant job opportunities based on their skills, experience, and preferences.
* **Messaging System**: Facilitate communication between employers and job **seekers through** a messaging system for inquiries, scheduling interviews, etc.
* Responsive UI: Develop a responsive and user-friendly interface accessible across various devices and screen sizes.

## Assumptions

* **Users have basic knowledge** of how to navigate web applications.
* Employers have the necessary information ready to post job listings.
* Job seekers have updated profiles with accurate information.
* Users have access to a stable internet connection for seamless usage of the application.

## Dependencies

* **React**: Frontend library for building user interfaces.
* **Express**: Web application framework for Node.js used for backend development.
* **MongoDB Atlas**: Cloud-based database service for storing application data.
* **Cloudinary**: Cloud storage service for managing and serving media files such as images and documents.
* **Node.js**: JavaScript runtime environment for executing backend code.
* **NPM (Node Package Manager)**: Package manager for installing and managing project dependencies.

# **SYSTEM ARCHITECTURE**

The software system architecture refers to the logical organization of a distributed system into software components. It defines how components of a software system are assembled, their relationship and communication between them. It serves as a blueprint for software application and development basis for developer team. An effective architecture serves as the conceptual glue that holds every phase of the project together for all of its stakeholders, enabling agility, time and cost savings, and early identification of design risks.

**5.1 Operational Architecture Characteristics:**

* Availability: Ensures that the application is consistently accessible to both employers and job seekers, minimizing downtime and disruptions.
* Performance: Optimizes the speed and responsiveness of the application, ensuring quick loading times for job listings and candidate profiles.
* Reliability: Guarantees that the system functions correctly and consistently, providing accurate search results and reliable communication between users.
* Low fault tolerance: Minimizes the occurrence of errors or system failures, enhancing the overall reliability of the application.
* Scalability: Allows the application to handle increasing numbers of users and data efficiently, ensuring that it can grow with the demand without compromising performance.

**2.2 Structural Architecture Characteristics:**

* Configurability: Offers flexibility in configuring various aspects of the application, such as search filters and user preferences.
* Extensibility: Supports the addition of new features or functionalities in the future, allowing for continuous improvement and adaptation to changing requirements.
* Supportability: Facilitates easy maintenance and support of the application, with clear documentation and modular design.
* Portability: Enables the application to run smoothly across different devices and platforms, ensuring accessibility for a diverse user base.
* Maintainability: Simplifies the process of maintaining and updating the application over time, making it easier for developers to manage and enhance.

**2.3 Cross-Cutting Architecture Characteristics:**

* Accessibility: Ensures that the application is accessible to users with disabilities, adhering to accessibility standards and guidelines.
* Security: Implements robust security measures to protect user data and privacy, preventing unauthorized access or data breaches.
* Usability: Focuses on providing a user-friendly interface and intuitive navigation, enhancing the overall user experience for both employers and job seekers.
* Privacy: Safeguards the confidentiality of user information and ensures compliance with data protection regulations, respecting user privacy rights.
* Feasibility: Ensures that the architectural design is feasible within the constraints of time, budget, and available resources, balancing stakeholder needs and project goals.

# **DETAILED DESCRIPTION OF COMPONENTS**

For detailed description of the components, please refer **Appendix A – Detailed Description of Components**

Design specification outlines the components of the "Jobsearch" application, including the frontend, backend API, database, and Cloudinary integration. Each component serves a specific purpose and has dependencies on other components to function effectively. The interfaces, resources, processing, and data aspects of each component are also described to provide a comprehensive understanding of their functionalities.

# **INTEGRATIONS**

**Cloudinary Integration:**

The "Jobsearch" application utilizes Cloudinary for cloud storage of media files such as candidate resumes, company logos, and job-related images.

Cloudinary's API will be integrated into the application's backend to facilitate seamless uploading, storage, and retrieval of media files.

The integration will enable users (both employers and job seekers) to upload and manage media files associated with their profiles, job listings, and applications.

**MongoDB Atlas Integration:**

MongoDB Atlas is used as the database management system for storing application data such as user profiles, job listings, and candidate information.

The application's backend, built with Express.js, will utilize MongoDB Atlas to perform CRUD (Create, Read, Update, Delete) operations on the database.

Integration with MongoDB Atlas ensures scalability, reliability, and efficient management of data for the "Jobsearch" application.

**React Frontend Integration:**

The frontend of the "Jobsearch" application is built using React, providing a dynamic and responsive user interface for both employers and job seekers.

React components will be integrated with the backend API endpoints to enable seamless communication between the frontend and backend of the application.

Integration with React allows for interactive features such as real-time job search, filtering, and user authentication within the application.

**Express.js Backend Integration:**

Express.js serves as the backend framework for the "Jobsearch" application, handling server-side logic, routing, and API endpoints.

Integration with Express.js enables the implementation of authentication mechanisms, data validation, and CRUD operations on the MongoDB Atlas database.

Express.js middleware will be utilized for error handling, logging, and security features to enhance the robustness of the backend infrastructure.

# **APPENDICES**

## Appendix A – Detailed Description of Components

| **Identification** | **Login ,Register, home screen, error 404 screen, job portal screen** |
| --- | --- |
| **Type** | Class/Form/ |
| **Purpose** | to fill the details and information and provide all the CRUD functionalities to the users |
| **Subordinates** | This screen contains links to the following screens:  Main Menu Screen  New User Account Screen  Login Screen  job search screen  job board screen |
| **Dependencies** | login/register screen and job portal screen to home screen |
| **Interfaces** | this application is designed to cater all the UI and UX requirements needs and easy to navigate from screens to screens |
| **Resources** | Database access needed, Cloud Storage API needed to working of application |
| **Processing** | only type of processing is done on the database and client side processing |
| **Data** | database data stored in mongodb , resume images stored on cloudiary , blobs data etc |