SOFTWARE REQUIREMENTS SPECIFICATION

For

JOB SEARCH APPLICATION

PREPARED BY

MADHUMITHA G

VANITHA LAKSHMI G

DEEPA LAKSHMI R

1. Introduction

1.1 Purpose

The purpose of this Software Requirements Specification (SRS) document is to provide a comprehensive outline of the requirements, functionalities, and constraints for the development of a Job Search Application. This application, built using Java for the back-end and React.js for the front-end, is designed to address the needs of job seekers and employers, offering a robust platform for job discovery and talent acquisition.

1.2 Scope

The Job Search Application aims to create a user-centric platform that bridges the gap between job seekers and employers. It facilitates job discovery, application management, and seamless communication between these two critical user groups. The scope of this document encompasses a detailed description of the application's features, interfaces, and performance expectations.

1.3 Definitions, Acronyms, and Abbreviations

Job Seeker: A user actively seeking employment opportunities.

Employer: An organization or individual posting job listings and managing the hiring process.

Java: A widely-used programming language for the application's back-end.

React.js: A JavaScript library for building dynamic and responsive user interfaces.

SRS: Software Requirements Specification, the document you are currently reading.

UI: User Interface.

API: Application Programming Interface.

1.4 References

Include any external documents, standards, or sources that were consulted during the development of this SRS.

1.5 Overview

This SRS will provide a detailed description of the Job Search Application's objectives, functionalities, and constraints. It is intended to serve as a guiding document for the development team, ensuring a common understanding of the project's scope and requirements. Additionally, it will aid stakeholders, including project managers, designers, developers, and quality assurance teams, in their respective roles throughout the project's lifecycle.

2. Overall Description

2.1 Product Perspective

The Job Search Application is a web-based platform that connects job seekers and employers. It operates independently, offering job seekers the ability to search for job listings and apply for positions, while employers can post job listings and manage the hiring process. The application does not integrate with external systems, but it may utilize third-party services for additional features such as authentication or analytics.

2.2 Product Functions

2.2.1 Job Seeker Functions

User Registration: Job seekers can create accounts by providing necessary information.

Job Search: Users can search for job listings based on various criteria, including location, industry, job type, and keywords.

Application Submission: Job seekers can apply for jobs by submitting their resumes and cover letters.

User Profile: Users can manage their profiles, including personal information and job application history.

Messaging: A messaging system enables communication between job seekers and employers.

2.2.2 Employer Functions

Employer Registration: Employers can create accounts, providing company details.

Job Posting: Employers can post job listings with descriptions, requirements, and application deadlines.

Application Management: Employers can view and manage job applications, including shortlisting and communication with applicants.

User Profile: Employers can manage their company profiles and job listings.

Messaging: Employers can communicate with job seekers who have applied for their positions.

2.3 User Classes and Characteristics

2.3.1 Job Seekers

Job seekers using the application are typically individuals actively looking for employment opportunities. They possess varying levels of experience and skills.

2.3.2 Employers

Employers represent organizations or individuals seeking to fill job positions. They may range from small businesses to large corporations.

2.4 Operating Environment

The Job Search Application will be hosted on web servers. Users can access it via modern web browsers on desktop and mobile devices. The application's front-end is developed using React.js, and the back-end is implemented in Java. The database management system (DBMS) used is MySQL.

2.5 Design and Implementation Constraints

The application should be responsive and accessible on a variety of devices and screen sizes.

Development will adhere to industry best practices for security and data protection.

Third-party services may be used for features such as user authentication, email notifications, and analytics.

2.6 User Documentation

Comprehensive user documentation will be provided to guide job seekers and employers in using the application effectively. This documentation will include user guides, FAQs, and tutorials.

2.7 Assumptions and Dependencies

It is assumed that users have access to the internet and modern web browsers.

The application will depend on external services for tasks such as email notifications and analytics.

The development team assumes access to development and production servers for deployment.

3. Specific Requirements

3.1 External Interfaces

3.1.1 User Interfaces (UI)

The user interface (UI) of the Job Search Application shall adhere to modern design principles to provide an exceptional user experience. It must be responsive, ensuring seamless functionality on various devices, including desktop computers, tablets, and mobile phones. The UI shall incorporate intuitive navigation, clear information presentation, and accessibility features to accommodate users with disabilities.

Users shall have the ability to register, log in, and log out from the application. Registration shall involve the submission of necessary information, including name, email address, and password, followed by email confirmation to verify the user's identity. Password reset functionality shall be available for users who forget their login credentials.

3.1.2 Application Programming Interfaces (APIs)

The application shall offer a robust and secure RESTful API for communication between the front-end, developed using React.js, and the back-end, implemented in Java. The API shall support Create, Read, Update, and Delete (CRUD) operations for critical entities, including job listings, job applications, and user profiles. Security measures such as token-based authentication shall be in place to protect API endpoints from unauthorized access.

3.2 Functional Requirements

3.2.1 User Registration and Authentication

The Job Search Application shall provide user registration functionality. Users, both job seekers and employers, shall be able to create accounts using their email addresses and creating password-protected profiles. Upon registration, a confirmation email shall be sent to the user's provided email address to verify their account.

For users who forget their passwords, a password reset mechanism shall be implemented. This mechanism will enable users to reset their passwords through a secure email-based process, ensuring that only authorized users can regain access to their accounts.

3.2.2 Job Search and Filtering

Job seekers shall have the capability to search for job listings based on a variety of criteria, including keywords, location, industry, and job type. Search results shall be displayed in an organized manner, providing essential details about each job listing. Users shall also have the option to save job listings and receive notifications when new positions matching their criteria become available.

3.2.3 Job Posting

Employers shall be able to create and manage job listings efficiently. When creating a job listing, employers shall provide detailed job descriptions, requirements, and application deadlines. Employers shall also have the ability to edit and remove job listings as needed. All job listings shall be presented with comprehensive information to attract potential candidates.

3.2.4 Application Submission and Tracking

Job seekers shall have the functionality to submit job applications through the platform. This process will involve uploading resumes and cover letters. Upon successful submission, users shall receive confirmation emails as proof of application submission. To enable efficient tracking, job seekers shall have access to a personalized dashboard where they can monitor the status and progress of their applications.

3.2.5 User Profile Management

Both job seekers and employers shall be able to manage their profiles within the application. This functionality includes editing personal information, contact details, and, in the case of job seekers, uploading and updating resumes. Employers shall also be able to manage their company profiles, providing information about their organizations, which will be visible to job seekers.

3.2.6 Messaging System

The Job Search Application shall facilitate communication between job seekers and employers through an integrated messaging system. Users shall be able to send and receive messages within the platform. Messages shall be organized and categorized by job listing, ensuring that communication remains contextually relevant. Users shall receive notifications to inform them of new messages, fostering timely and efficient communication.

3.3 Non-Functional Requirements

3.3.1 Performance

To provide an optimal user experience, the application shall load within a reasonable time frame, with page load times targeted to be under two seconds per page. The API shall be designed to support a high number of concurrent users, with a target of accommodating over 1000 concurrent requests to ensure scalability and responsiveness during peak usage.

3.3.2 Security

Security is paramount, and the application shall implement industry-standard security measures. User data shall be securely stored, following best practices for data protection. User authentication shall be implemented using secure protocols, and access to data shall be role-based to ensure data privacy.

3.3.3 Scalability

The application shall be designed with scalability in mind to handle an increasing number of users and job listings efficiently. The database shall be optimized for quick data retrieval and storage, ensuring performance as the application scales.

3.3.4 Availability

The application shall target a high availability rate, with an uptime goal of 99.9%. This high availability shall be achieved through robust server architecture and infrastructure. Regular backups and comprehensive disaster recovery procedures shall be in place to minimize downtime and data loss.

3.3.5 Usability

Usability is a key consideration, and the application shall be accessible to users with disabilities. User interfaces shall adhere to UX best practices, providing intuitive navigation and a user-friendly experience for all users.

3.4 System Interfaces

3.4.1 Front-end (React.js)

The front-end, developed using React.js, shall communicate with the back-end through API requests. It shall be designed to be responsive, adapting to various screen sizes and devices. The user interface components shall be optimized for an intuitive user experience.

3.4.2 Back-end (Java)

The back-end, implemented in Java, shall handle API requests, process data, and manage user authentication. It shall be configured to efficiently handle concurrent connections and provide high-performance responses to front-end requests.

3.5 Database Requirements

3.5.1 Data Schema

The database shall include well-defined tables to store user profiles, job listings, job applications, and messages. Appropriate relationships between tables shall be established to maintain data integrity and support efficient queries.

3.5.2 Data Storage

User data, including profiles and application histories, shall be stored securely in the database. The database management system (DBMS) shall be optimized for efficient data retrieval and storage to ensure the application's responsiveness.

4. System Design

4.1 Architectural Overview

The Job Search Application shall adopt a client-server architecture, with the following key components:

4.1.1 Client-Side (Front-end)

The client-side, developed using React.js, is responsible for presenting the user interface to job seekers and employers. It communicates with the server-side through RESTful API calls for data retrieval, submission, and updates. The client-side ensures a responsive and user-friendly experience.

4.1.2 Server-Side (Back-end)

The server-side, implemented in Java, is responsible for processing requests from the client-side, managing the application's business logic, and interacting with the database. It includes the following modules:

4.1.2.1 API Layer

The API layer handles incoming HTTP requests from the client-side and routes them to the appropriate controllers for processing. It ensures secure communication between the client-side and server-side.

4.1.2.2 Controller Layer

The controller layer contains the business logic of the application. It processes requests, validates data, and coordinates interactions with other components, such as the database. The controller layer enforces access control and authentication.

4.1.2.3 Service Layer

The service layer contains the application's core services, including user management, job listing management, and messaging services. It encapsulates the application's logic, promoting modularity and maintainability.

4.1.2.4 Database

The database layer stores and manages data essential for the application's operation. It utilizes the MySQL database management system (DBMS) to store user profiles, job listings, job applications, and messaging data. The schema shall be designed to support efficient data retrieval and storage.

4.2 Front-end (React.js) Design

The front-end design focuses on creating an intuitive and responsive user interface. Key design elements include:

4.2.1 User Authentication

Users shall be able to register, log in, and reset passwords securely.

User sessions shall be managed to maintain authentication state.

4.2.2 User Profile Management

Job seekers and employers shall be able to create and update their profiles.

Job seekers shall be able to upload and manage their resumes.

4.2.3 Job Search and Listing

Job seekers shall access a search feature to find relevant job listings.

Job listings shall display comprehensive details, including job descriptions and application deadlines.

4.2.4 Job Application

Job seekers shall submit applications by uploading resumes and cover letters.

A dashboard shall provide an overview of application statuses.

4.2.5 Messaging

Users shall have access to an internal messaging system for communication.

Messages shall be organized by job listing, enabling contextual discussions.

4.3 Back-end (Java) Design

The back-end design focuses on performance, security, and scalability. Key design elements include:

4.3.1 API Design

The API shall follow RESTful principles for clear and structured communication.

Endpoints shall be secured using token-based authentication.

4.3.2 Business Logic

Controllers shall validate input data and enforce access control.

Services shall implement core application functionalities, such as user management and job handling.

4.3.3 Security

Data shall be encrypted during transmission using industry-standard protocols.

Passwords shall be securely hashed and salted before storage.

Role-based access control shall restrict data access as per user roles.

4.3.4 Scalability

The application shall be designed to handle increased load gracefully.

Horizontal scaling shall be considered to accommodate growing user and job listing volumes.

4.3.5 Data Storage

The MySQL database shall maintain referential integrity and be optimized for performance.

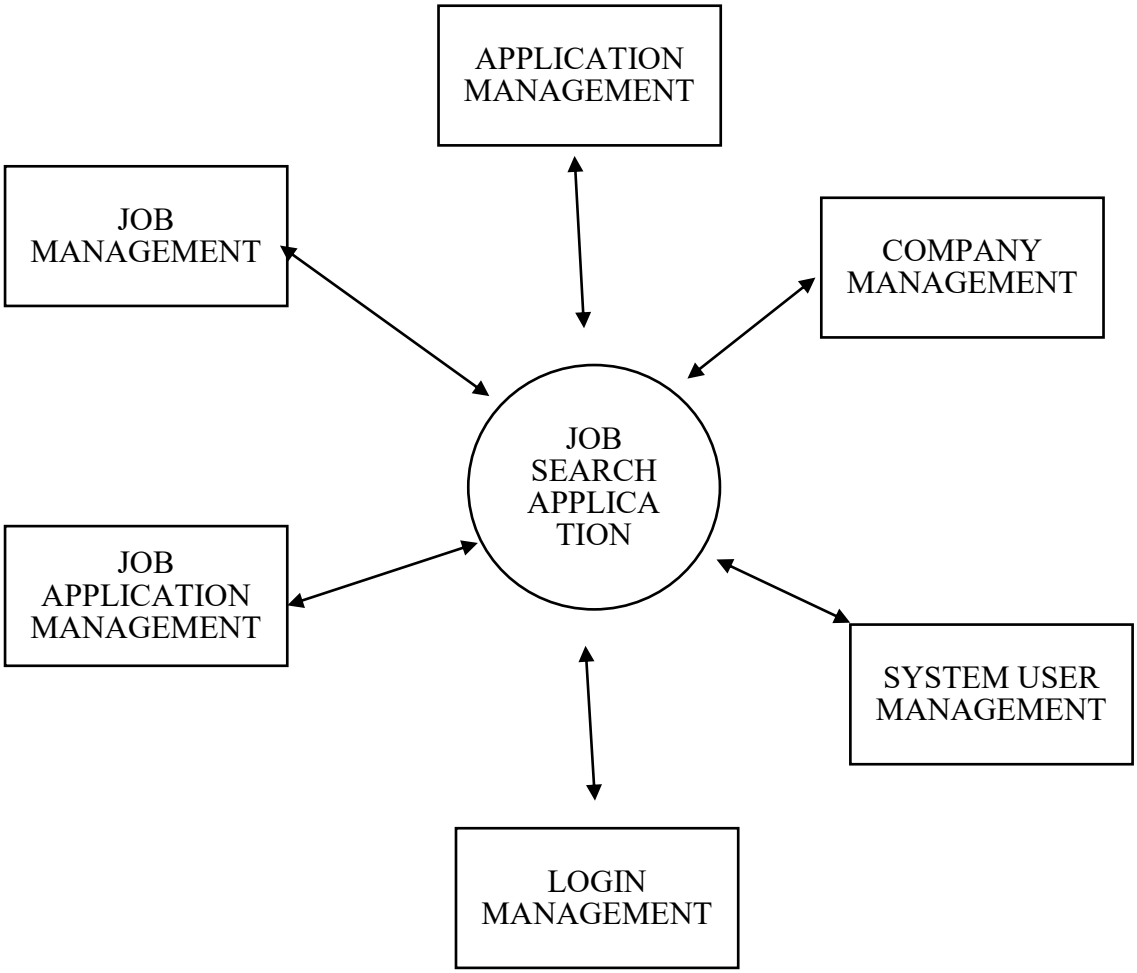
Data backup and recovery procedures shall be in place to prevent data loss.

4.4 Integration of Front-end and Back-end

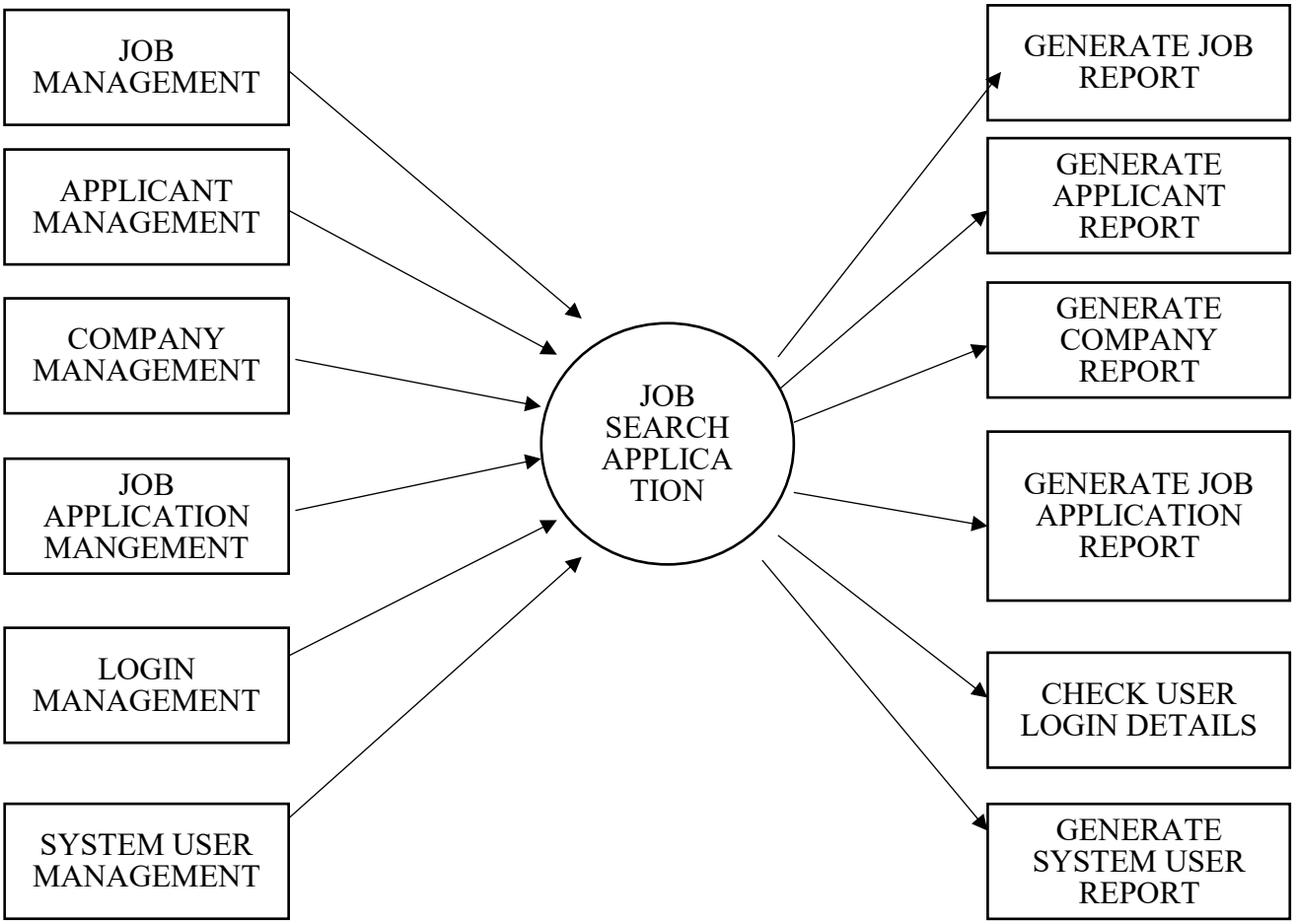
The integration of the React.js front-end and Java back-end shall be seamless, ensuring efficient data exchange and communication between client and server. RESTful API endpoints shall facilitate data transmission while adhering to security measures.

4.5 Data Flow Diagrams (DFD)

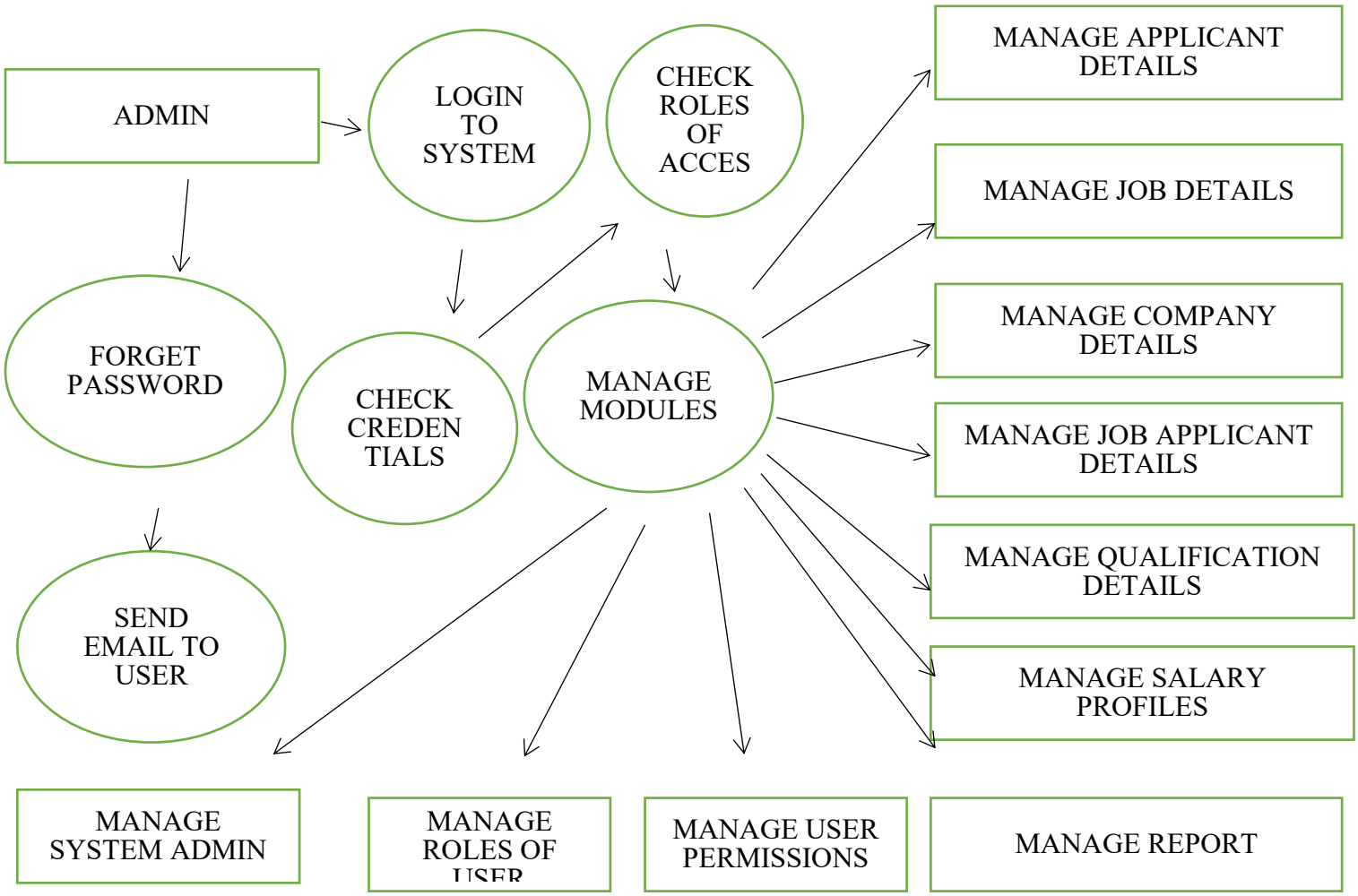
Data Flow Diagrams shall be developed to illustrate the flow of data within the system, outlining data sources, processes, and destinations. This visual representation shall aid in understanding the system's data dynamics.



ZERO LEVEL DFD-JOB SEARCH APPLICATION



FIRST LEVEL DFD – JOB SEARCH APPLICATION



SECOND LEVEL DFD-JOB APPLICATION SEARCH

4.6 Sequence Diagrams

Sequence diagrams shall be created to depict interactions between system components during use cases such as user registration, job application submission, and messaging.

5. User Interface Design

5.1 Wireframes and Mockups

Creating wireframes and mockups for a job search application is an essential step in the design process. It helps you visualize the layout and user interface of your application before development begins.

5.2 UI Components

- Navigation Bar:

Provides access to different sections of the app, such as Home, Jobs, Saved Jobs, Profile, and Settings. May include a search bar for quick job searches.

- Search Bar:

Allows users to enter keywords, location, job type, or other search criteria.

May include a "Search" button or offer real-time suggestions as users type.

- Filtering Options:

Enables users to refine search results based on parameters like location, job type, industry, salary range, and experience level.

May include dropdowns, checkboxes, or sliders.

- Job Listings:

Displayed in a grid or list format on the search results page.

Each listing includes essential information like job title, company name, location, publication date, and a brief description.

The UI components should be designed with a consistent visual style, including typography, color schemes, and spacing, to create a cohesive and user-friendly job search application. Additionally, conducting usability testing and gathering user feedback can help refine and improve these components for a better user experience.

5.3 Navigation Design

Effective navigation design in job search application is crucial for providing a positive user experience and encouraging users to explore your app's features. A well-designed navigation system makes it easy for users to find and access the various features and sections of your app. Regularly evaluate and refine your navigation based on user feedback and evolving user needs.

5.4 User Experience (UX) Guidelines

Ensure that users creating an exceptional user experience (UX) is crucial for a job search application, as it can significantly impact user satisfaction, engagement, and retention. Here are some key UX guidelines to consider when designing your job search application:

1. Simple and Intuitive User Interface:

Clear Information Hierarchy: Organize information logically, with the most critical elements like job listings and search options prominently displayed. an easily find what they're looking for User-Friendly Forms: Keep registration, login, and profile-editing forms straightforward and user-friendly. Use inline validation to provide immediate feedback on user input.

2. Advanced Search and Filtering Options:

Robust Search Functionality: Develop a robust search engine that allows users to search for jobs using keywords, location, job type, industry, salary range, and more.

Advanced Filters: Offer advanced filters that enable users to refine their search results based on criteria such as experience level, education requirements, company size, and publication date.

3. Personalization and Recommendations:

User Profiles: Encourage users to create profiles where they can save job searches, track applications, and set preferences. Use this data to provide personalized job recommendations.

Recommended Jobs: Implement a recommendation engine that suggests relevant job listings based on a user's search history, saved jobs, and profile information.

Customized Alerts: Allow users to set up customized email alerts for specific job criteria and updates on their applications.

6. Maintenance and Support

6.1 Bug Tracking and Resolution

Bug tracking and resolution are crucial aspects of maintaining a job search application. Bugs can negatively impact user experience and application

functionality. When a bug is identified, it should be reported by users, testers, or developers. Bug reports should include detailed information about the issue, such as steps to reproduce, the environment it occurred in, and any relevant screenshots or error messages.

Bug tracking and resolution should be an ongoing process throughout the lifecycle of the job search application to ensure its reliability, performance, and user satisfaction.

6.2 Feature Requests and Enhancements

Managing feature requests and enhancements for a job search application is essential for its continued improvement and to meet the evolving needs of users.

Feature Request Management:

- Collection of Feature Requests: Encourage users to submit feature requests through the application, website, or customer support channels .Maintain a centralized repository to collect and categorize requests.
- Prioritization: Evaluate each feature request based on factors like user demand, potential impact on the application, alignment with the application's goals, and available resources. Categorize requests into priority levels, such as high, medium, or low priority.
- User Feedback: Consider gathering user feedback and conducting surveys to validate the importance of certain feature requests. Engage with users to understand the specific use cases and pain points related to the requested features.
- Communication: Keep users informed about the status of their feature requests, whether they are under consideration, planned, or declined. Maintain transparency about the reasons behind the decisions.
- Documentation: Document each feature request, including its description, priority, and any related user feedback. Use a consistent format to capture essential details.

Feature Enhancement Process:

- Requirement Gathering: Collaborate with stakeholders, including users, product managers, and developers, to define detailed requirements for the enhancement.
- Design and Prototyping: Create design mockups or prototypes to visualize the proposed enhancement. Seek feedback from stakeholders to refine the design.
- Development: Assign the enhancement to a development team or developer.

Implement the enhancement according to the defined requirements and design.

- Testing and Quality Assurance: Thoroughly test the enhancement to identify and resolve any issues or bugs .Ensure that the enhancement works seamlessly with existing features.
- User Acceptance Testing (UAT):Involve users or a representative group to conduct UAT and validate that the enhancement meets their expectations.
- Documentation and Training: Update user documentation, help guides, or onboarding materials to reflect the new enhancement .Provide training or tutorials if necessary.
- Release Planning: Schedule the enhancement for release based on its priority and development timeline .Consider grouping related enhancements into the same release.
- Communication: Notify users about the upcoming enhancement, its benefits, and any changes they need to be aware of. Provide release notes detailing the new features and improvements.

6.3 Backup and Recovery Procedures

Implementing robust backup and recovery procedures for a job search application is crucial to ensure data integrity and minimize downtime in case of unforeseen events such as system failures, data corruption, or disasters.

6.4 System Monitoring.

Effective system monitoring ensures that your job search algorithm continues to provide value to users, maintains high performance, and can quickly respond to issues or changes in usage patterns. It's an ongoing process that should be integrated into the overall system management strategy.

7. Appendix

7.1 Glossary

The following are the list of conventions and acronyms used in this document and the project as well:

- **Job Search Application:** A software platform that helps users find employment opportunities.
- **User:** An individual who interacts with the job search application, including job seekers and employers.
- **Job Seeker:** A user of the application who is searching for job opportunities.
- **Employer**: A user of the application who posts job listings and manages the hiring process.
- **Job Listing:** A posting created by an employer to advertise a job opportunity.
- **Resume/CV:** A document that job seekers upload to apply for jobs, containing their qualifications, skills, and experience.
- **Application:** The process of a job seeker expressing interest in a specific job listing and submitting their resume.
- **Profile**: A user's personal account information, including their resume, contact details, and preferences.
- **Authentication:** The process of verifying a user's identity through methods such as username/password or biometrics.
- **Authorization:** Granting or denying access to specific features or data based on a user's role and permissions

- **API (Application Programming Interface):** A set of rules and protocols that allow different software systems to communicate and exchange data.
- **Privacy Policy:** A document outlining how user data is collected, used, and protected within the application.
- **Terms of Service:** Rules and agreements that users must accept to use the application.
- **Feedback:** Mechanisms for users to provide comments, suggestions, or bug reports to the development team.
- **Responsive Design:** Design principles that ensure the application functions properly and looks good on various devices and screen sizes.
- **Security:** Measures and protocols in place to protect user data and the application from unauthorized access and threats.
- **Maintenance:** Ongoing updates, bug fixes, and improvements to keep the application running smoothly.
- **Deployment**: The process of making the application available for users, often on a web server or app store.