



SOFTWARE DESIGN SPECIFICATION

FULLSTACK DEVELOPMENT  
  
DEVELOP A WEB APPLICATION TO FIND RIGHT INTERNSHIP AND JOBS

|  |  |  |  |
| --- | --- | --- | --- |
| **Created By:** | Chirag Yadav | **Approved By:** | Harshada Topale |
| **Created On:** | 17-07-2024 | **Approved On:** | 17-07-2024 |

Page left blank intentionally

**INDEX**

[**1** **PURPOSE** 2](#_Toc142418236)

[**2** **PROJECT SCOPE** 2](#_Toc142418237)

[**3** **SYSTEM OVERVIEW** 2](#_Toc142418238)

[**4** **DESIGN CONSIDERATIONS** 2](#_Toc142418239)

[4.1 Requirements 3](#_Toc142418240)

[4.2 Assumptions 3](#_Toc142418241)

[4.3 Dependencies 3](#_Toc142418242)

[**5** **SYSTEM ARCHITECTURE** 3](#_Toc142418243)

[5.1 Architectural Strategies 4](#_Toc142418244)

[5.2 Structure & Relationships 4](#_Toc142418245)

[**6** **DETAILED DESCRIPTION OF COMPONENTS** 4](#_Toc142418246)

[**7** **INTEGRATION** 5](#_Toc142418247)

[**8** **APPENDICES** 1](#_Toc142418248)

[8.1 Appendix A – Detailed Description of Components 1](#_Toc142418249)

**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**

This document is created based on the requirement specification document. The purpose of this Software Design Specification (SDS) Document is to break down the project into components to describe in detail what the purpose of each component is and how it will be implemented. The SDS will also serve as a tool for verification and validation of the final product.

# **PROJECT SCOPE**

The scope of the Job & Internship Finder Web Application includes its distinct features, its benefits, and its limitations. The system's distinct features allow it to streamline job and internship searches by using advanced filtering and matching algorithms. The system enables the user to find personalized job and internship opportunities, eliminating the hassle of browsing through irrelevant postings.

# **SYSTEM OVERVIEW**

This system consists of several interconnected components:

* **Frontend**: Built using React and Tailwind CSS to ensure responsiveness and an intuitive user experience.
* **Backend**: Powered by Node.js and Express for handling server-side operations and database interactions.
* **Database**: MongoDB stores user data, job postings, applications, and chats.

# **DESIGN CONSIDERATIONS**

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

## Requirements

A user-friendly interface for job and internship searches.

Secure user authentication and authorization.

Data storage for user profiles and job applications.

Integration with external job listing APIs.

## Assumptions

Users will provide accurate and complete profiles to enhance recommendation accuracy.

Recruiters will interact with the platform for direct candidate engagement.

Internet connectivity is available for optimal system performance.

## Dependencies

Dependency on external job listing platforms for data.

Compatibility with different browsers and devices.

# **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.

The Software architecture:

* Defines structure of a system
* Defines behaviour of a system
* Defines component relationship
* Defines communication structure
* Balances stakeholder’s needs
* Influences team structure
* Focuses on significant elements
* Captures early design decisions

Below some important characteristics which are commonly considered are explained.

**Operational Architecture Characteristics:**

* Availability
* Performance
* Reliability
* Low fault tolerance
* Scalability

**Structural Architecture Characteristics:**

* Configurability
* Extensibility
* Supportability
* Portability
* Maintainability

**Cross-Cutting Architecture Characteristics:**

* Accessibility
* Security
* Usability
* Privacy
* Feasibility

## Architectural Strategies

Frontend-Backend Communication: REST APIs handle client-server interactions.

Database: MongoDB Atlas ensures scalable and reliable data storage.

Authentication: Uses JSON Web Tokens (JWT) for secure user sessions.

Job Recommendation: Algorithm leveraging user data for personalized results.

## Structure & Relationships

Frontend: Communicates with backend through REST APIs.

Backend: Processes requests, interacts with the database.

Database: Stores user profiles, job postings, and application data.

**Flowchart:**

User → Frontend → Backend → Database

# **DETAILED DESCRIPTION OF COMPONENTS**

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

The below template will be used to specify the details of all the components

**Table 1: Detailed Design Specification Template**

|  |  |
| --- | --- |
| **Identification** | The unique name for the component and the location of the component in the system. |
| **Type** | A module, a subprogram, a form, a data file, a control procedure, a class, etc. |
| **Purpose** | Function and performance requirements implemented by the design component, including derived requirements. Derived requirements are not explicitly stated in the SRS - but are implied or adjunct to formally stated SDS requirements. |
| **Subordinates** | The internal structure of the component, the constituents of the component, and the functional requirements satisfied by each part. |
| **Dependencies** | How the component’s function and performance relate to other components. How this component is used by other components. The other components that use this component. Interaction details such as timing, interaction conditions (such as order of execution and data sharing), and responsibility for creation, duplication, use, storage, and elimination of components. |
| **Interfaces** | Detailed description of all external or internal interfaces as well as of any mechanism for communicating through messages, parameters, or common data areas. All error messages and error codes should be identified. All screen formats, interactive messages, and other user interface components (originally defined in the SRS) should be given here. |
| **Resources** | A complete description of all resources (hardware or software) external to the component but required to carry out its functions. |
| **Processing** | A full description of the functions presented in the Function subsection. Pseudocode can be used to document algorithms, equations, and logic. |
| **Data** | For the data internal to the component, describes the representation method, initial values, use, semantics, and format. |

# **INTEGRATIONS**

External Job APIs: Enhances job data by integrating listings from external platforms.

Authentication Services: Secure login with JWT.

# **APPENDICES**

## Appendix A – Detailed Description of Components

*For e.g.*

|  |  |
| --- | --- |
| *Identification* | ***Login Screen*** |
| *Type* | *Class/Form* |
| *Purpose* | *Ensures that only authenticated users (job seekers and employers) can access the application.* |
| *Subordinates* | *This screen contains links to the following screens:*  *- Register Screen*  *- Forgot Password Screen*  *- Home Screen (after successful login).* |
| *Dependencies* | *The following screens link to this screen:*  *- Register Screen (for new users).* |
| *Interfaces* | *The login form includes two text boxes (username and password), a "Login" button, and links to "Register" screens. It uses responsive design for mobile and desktop views.* |
| *Resources* | *Requires access to the authentication database to verify login credentials.* |
| *Processing* | *Validates input fields, sends API requests to the backend for authentication, and navigates the user to the Home Screen upon success. Displays error messages for incorrect credentials.* |
| *Data* | *The data for the Login Screen is the username and password entered by the user. It is validated with a query against the database.* |

|  |  |
| --- | --- |
| **Identification** | **Register** |
| **Type** | Class/Form/ |
| **Purpose** | Allows new users to create an account by entering their personal and professional details. |
| **Subordinates** | Links to:  - Login Screen (for existing users). |
| **Dependencies** | Linked to the Login Screen for users to switch if they already have an account. |
| **Interfaces** | Form fields include:  - Name  - Email  - Password  - Confirm Password  - Submit button. |
| **Resources** | Requires access to the database for storing new user information. |
| **Processing** | Validates form fields, ensures the password matches the confirmation field, and sends the data to the backend for account creation. |
| **Data** | Includes name, email, and hashed password stored in the database. |

|  |  |
| --- | --- |
| **Identification** | **Job Screen** |
| **Type** | Class/Form/ |
| **Purpose** | Enables users to search and filter jobs or internships based on preferences such as location, skills, and industry. |
| **Subordinates** | Links to:  - JobDetailsScreen (for viewing job details). |
| **Dependencies** | Requires job data from the backend and filter criteria input from the user. |
| **Interfaces** | Includes a search bar, filter dropdowns, and a results grid displaying job postings. |
| **Resources** | Access to the job database to retrieve and display matching results. |
| **Processing** | Sends user-input filter data to the backend API, processes the response, and dynamically displays results. |
| **Data** | User search input (keywords, location, filters) and job data fetched from the database. |