Submitted by Rinshad

Sl.	Table of Contents	PAGE NO.

INTRODUCTION

A job portal is a website dedicated for online information about recruiters as well as job seekers. A job portal helps both the job seekers and recruiters finding the right organisation for the employees.

PURPOSE

The purpose of this SRS document is to provide a detailed overview of our software product, its parameters and goals. This document describes the project's target audience and its user interface, hardware and software requirements.
Documenting functional requirements and non functional requirements helps to avoid disappointment with a new system. It is also important in keeping the project on track, to budget and delivered on time.

SCOPE

- The system provides jobs catalogue and information to members and helps them decide on the jobs to apply.

- Features are the “tools” you use within a system to complete a set of tasks or actions. Functionality is how those features actually work to provide you with a desired outcome.

- When it comes to limitations or exclusions in terms of features or functionality, it is important to consider the specific context or domain in question. Different systems, software, or products may have their own unique limitations or exclusions.

- The system is designed to cater to various user roles or types of users, including job seekers, employers, and administrators. Each user role has specific needs and functionalities within the system.

OBJECTIVES

The primary objectives of a job portal system are to provide a seamless and efficient platform for job seekers to find suitable employment opportunities and for employers to find qualified candidates.

The System Requirements Specification (SRS) is a document that outlines the goals and problems that a system aims to address. It serves as a blueprint for the development team, providing a clear understanding of what needs to be achieved and how the system should function. The SRS is typically created during the early stages of the software development life cycle and acts as a reference for all stakeholders involved in the project.

FUNCTIONAL REQUIREMENTS

The admin module is an essential component of a software application that is designed to provide administrative functionalities and privileges to authorised users. It allows administrators to manage and control various aspects of the system, such as user management, content management, system configuration, and security settings.In this section, we will discuss the use cases, features, and user stories of the admin module in detail.

Use Cases:
1. User Management: The admin module enables administrators to create, update, and delete user accounts. It also allows them to assign roles and permissions to users, manage user profiles, and handle password resets.

2. Content Management: Administrators can use the admin module to manage the content of the application. They can create, edit, and delete various types of content such as articles, blog posts, images, videos, and documents. The module may also provide features like version control, content approval workflows, and content scheduling.

3. System Configuration: The admin module allows administrators to configure various settings of the system. This includes defining system-wide parameters like email server settings, database connections, API integrations, and third-party services. Administrators can also customize the look and feel of the application by configuring themes, templates, and layouts.

4. Security Management: Administrators have the responsibility to ensure the security of the application. The admin module provides features for managing user access controls, authentication mechanisms (e.g., single sign-on), password policies, and session management. It may also include features like audit logs and security reports for monitoring and analyzing system activities.

5. Reporting and Analytics: The admin module can offer reporting and analytics capabilities for administrators to gain insights into system usage, performance metrics, user behavior patterns, and other relevant data. This helps in making informed decisions regarding system improvements or optimizations.

Features:
1. User Interface: The admin module typically provides a user-friendly interface for administrators to perform their tasks efficiently. It may include features like dashboards, menus, search functionality, and customizable layouts to enhance usability.

2. Role-based Access Control: The admin module supports role-based access control (RBAC), allowing administrators to define different roles with specific permissions. This ensures that only authorized users can access and perform certain actions within the system.

3. Workflow Management: The admin module may include workflow management capabilities to streamline business processes. Administrators can define and manage workflows for content approval, user onboarding, or any other custom processes specific to the application.

4. Notifications and Alerts: Administrators can receive notifications and alerts through the admin module for important events or system updates. This helps them stay informed about critical issues, such as security breaches, system failures, or pending tasks.

5. Data Management: The admin module provides functionalities for managing data within the system. This includes importing/exporting data, performing bulk operations (e.g., batch user creation), and data backup/restore capabilities.

User Stories:
1. As an administrator, I want to be able to create new user accounts and assign appropriate roles and permissions based on their responsibilities within the organization.

2. As an administrator, I want to have a centralized dashboard where I can monitor system performance, user activities, and generate reports for analysis and decision-making purposes.

3. As an administrator, I want to be able to customize the application's appearance by selecting themes, modifying templates, and arranging layouts according to our branding guidelines.

PRIMARY ACTORS OF THE SYSTEM

In a system, there are various actors that play different roles and interact with the system in different ways. These actors can be categorized into primary actors, such as users, administrators, and external systems. Let's explore each of these primary actors in detail:

1. Users: Users are individuals or entities who directly interact with the system to perform specific tasks or access its functionalities. They can be further classified into different types based on their roles and permissions within the system. Some common types of users include:

- End Users: These are the individuals who utilize the system to accomplish their goals or tasks. They interact with the system through user interfaces, such as web browsers, mobile applications, or desktop software.

- Power Users: Power users are advanced users who have a deeper understanding of the system's functionalities and often have elevated privileges. They may have additional capabilities or access to certain administrative features.

- System Administrators: System administrators are responsible for managing and maintaining the system. They have elevated privileges and can perform tasks such as user management, configuration changes, security settings, and overall system maintenance.

2. Administrators: Administrators are individuals who have higher-level access and control over the system compared to regular users. They are responsible for managing and configuring the system to ensure its smooth operation. Some common types of administrators include:

- System Administrators: As mentioned earlier, system administrators have overall responsibility for managing and maintaining the system. They handle tasks such as user management, security configurations, backups, updates, and troubleshooting.

- Database Administrators: In systems that rely on databases, database administrators (DBAs) play a crucial role in managing and optimizing the database infrastructure. They handle tasks like database design, performance tuning, backup and recovery, and ensuring data integrity.

- Network Administrators: Network administrators focus on managing the network infrastructure that supports the system. They handle tasks such as network configuration, monitoring, troubleshooting network issues, and ensuring network security.

3. External Systems: External systems are entities or software components that interact with the system but exist outside of its boundaries. These systems can exchange data, trigger actions, or provide services to the primary system. Some examples of external systems include:

- APIs (Application Programming Interfaces): Many systems integrate with external APIs to exchange data or access services provided by other systems. These APIs allow seamless communication and interaction between different software applications.

- Third-party Services: Systems often rely on third-party services for various functionalities, such as payment gateways, authentication providers, messaging services, or cloud storage providers. These external services enhance the capabilities of the primary system.

- Legacy Systems: In some cases, older or legacy systems may still be in use and need to interact with the primary system. Integration with these legacy systems is necessary to ensure smooth operations and data exchange.

In summary, the primary actors in a system include users who directly interact with the system, administrators who manage and maintain the system, and external systems that interact with the primary system through APIs or other means.

USE CASE DIAGRAMS

A use case diagram is a visual representation of the interactions between a system and its external entities or actors. It showcases the high-level functionalities of the system and helps in understanding how different actors interact with the system to achieve specific goals or tasks.

In a use case diagram, the system is represented as a box in the center, surrounded by actors and their associated use cases. Actors are external entities that interact with the system, such as users, other systems, or hardware devices. Use cases represent specific tasks or functionalities that the system provides to its actors.

The primary purpose of a use case diagram is to provide an overview of the system's functionality from an external perspective. It helps in identifying the different actors involved, their roles, and the tasks they perform within the system. By visualizing these interactions, stakeholders can gain a better understanding of how the system will be used and what features it needs to support.

Use case diagrams are commonly used in software development projects to capture requirements and define the scope of the system. They serve as a communication tool between stakeholders, including developers, designers, project managers, and end-users. Use case diagrams can also be used during system testing to ensure that all required functionalities are implemented correctly.

To create a use case diagram, several steps need to be followed:

1. Identify Actors: Start by identifying all the external entities that interact with the system. These can include users, other systems, or hardware devices.

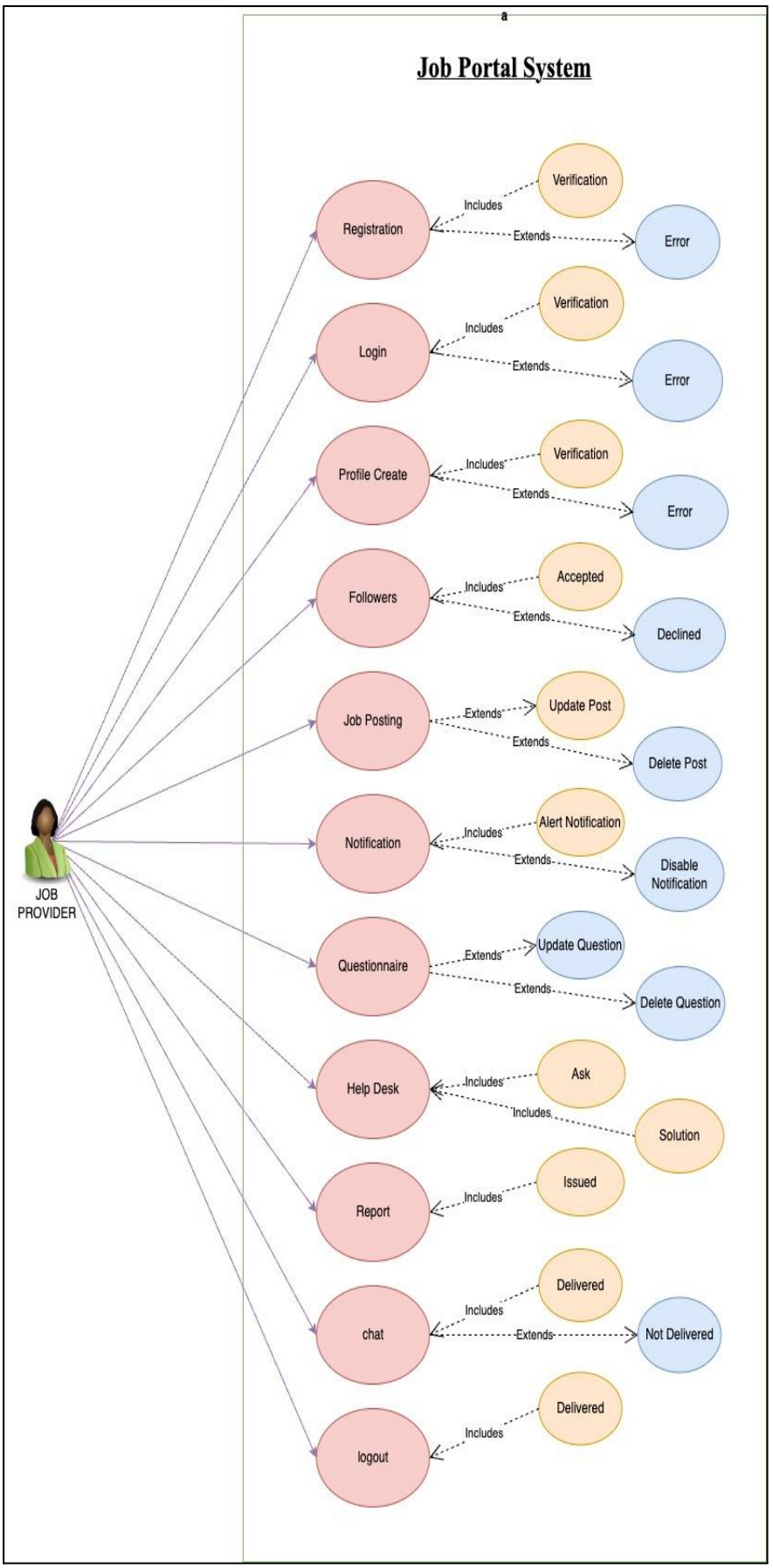
2. Define Use Cases: Identify all the tasks or functionalities that the system needs to provide to its actors. Each use case should represent a specific goal or task that an actor wants to achieve.

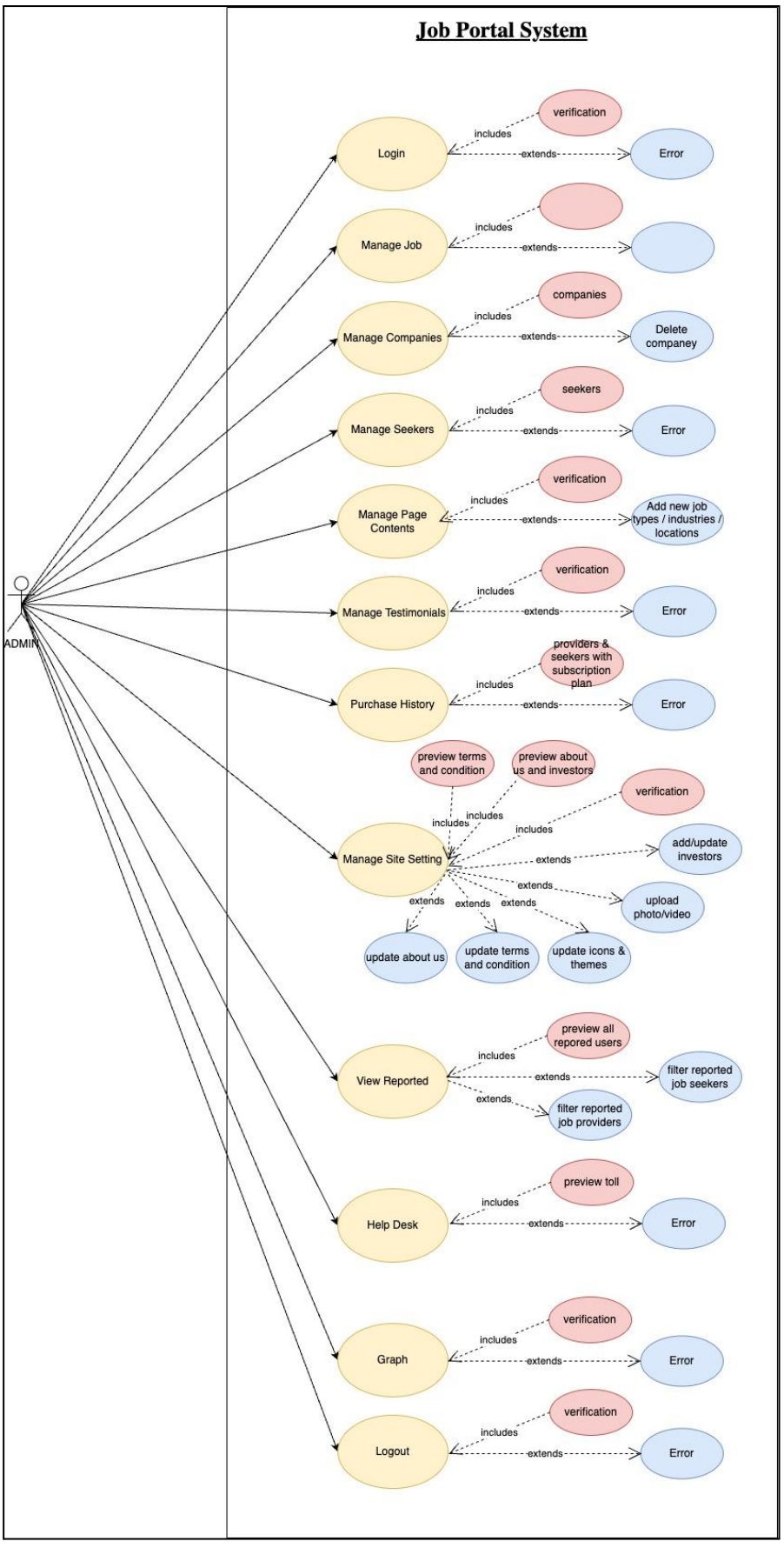
3. Establish Relationships: Connect each actor with their associated use cases using lines or arrows. This represents how each actor interacts with the system and which functionalities they have access to.

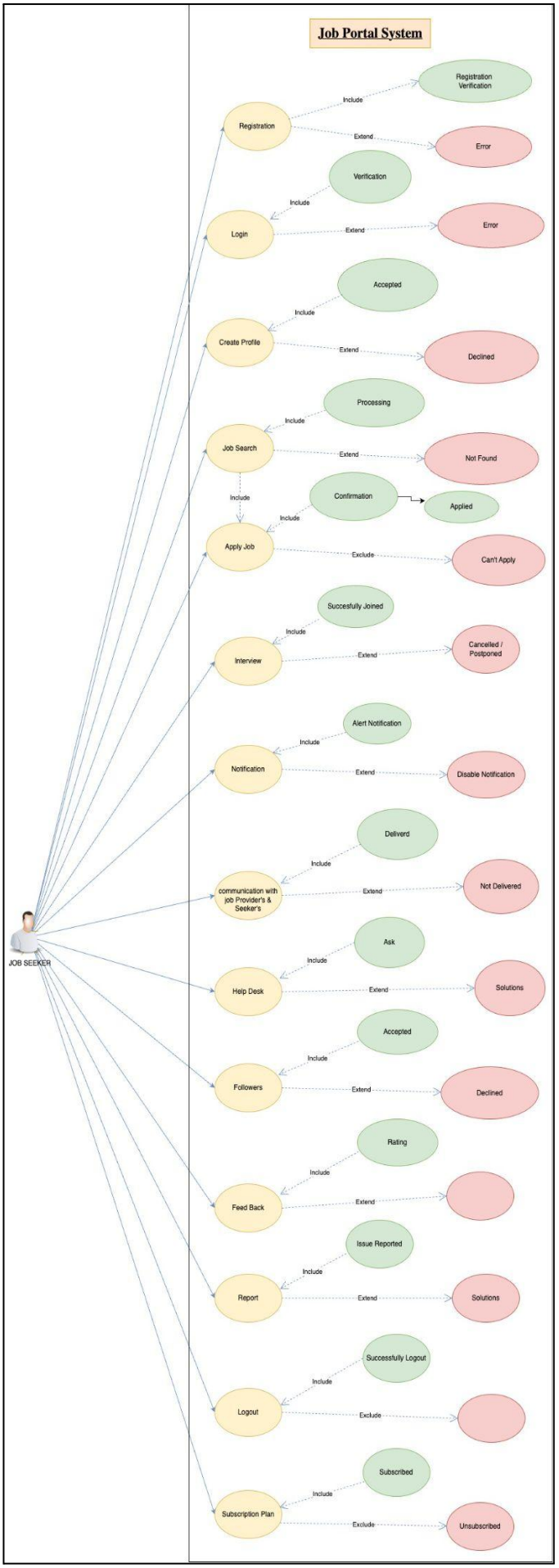
4. Refine and Validate: Review the use case diagram with stakeholders to ensure that all necessary actors and use cases are included. Make any necessary revisions based on feedback.

5. Document: Once the use case diagram is finalized, it should be documented and shared with the development team and other stakeholders. This documentation serves as a reference for system design and implementation.

In summary, a use case diagram is a visual representation of the interactions between a system and its external entities or actors. It helps in understanding the high-level functionalities of the system and how different actors interact with it to achieve specific goals or tasks.


a
Job Portal System
JOB PROVIDER
Registration
Verification
Includes
Error
Extends
Login
Verification
Includes
Error
Extends
Profile Create
Verification
Includes
Error
Extends
Followers
Accepted
Includes
Declined
Extends
Job Posting
Update Post
Extends
Delete Post
Extends
Notification
Alert Notification
Includes
Disable Notification
Extends
Questionnaire
Update Question
Extends
Delete Question
Extends
Help Desk
Ask
Includes
Solution
Includes
Report
Issued
Includes
chat
Delivered
Includes
Not Delivered
Extends
logout
Delivered
Includes


Job Portal System
ADMIN
Login
verification
Error
includes
extends
Manage Job
Manage Companies
companies
Delete companey
Manage Seekers
seekers
Error
Manage Page Contents
verification
Add new job types / industries / locations
Manage Testimonials
verification
Error
Purchase History
providers & seekers with subscription plan
Error
Manage Site Setting
preview terms and condition
preview about us and investors
verification
add/update investors
upload photo/video
update about us
update terms and condition
update icons & themes
View Reported
preview all repored users
filter reported job seekers
filter reported job providers
Help Desk
preview toll
Error
Graph
verification
Error
Logout
verification
Error


Job Portal System
JOB SEEKER
Registration
Registration Verification
Error
Verification
Login
Error
Accepted
Create Profile
Declined
Processing
Job Search
Not Found
Confirmation
Applied
Apply Job
Can't Apply
Succesfully Joined
Interview
Cancelled / Postponed
Alert Notification
Notification
Disable Notification
Deliverd
communication with job Provider's & Seeker's
Not Delivered
Ask
Help Desk
Solutions
Accepted
Followers
Declined
Rating
Feed Back
Issue Reported
Report
Solutions
Successfully Logout
Logout
Subscribed
Subscription Plan
Unsubscribed
Include
Extend
Exclude

USER STORIES

Here are two examples of user stories

login
In the context of a job portal **login**, user stories can be used to define the login functionality from different user perspectives. Here are some examples:

1. As a job seeker, I want to be able to create an account and log in with my credentials so that I can access personalized job recommendations, save job searches, and apply for jobs easily.

2. As an employer, I want to have a secure login system that allows me to access my company's account, post job listings, manage applications, and communicate with potential candidates.

3. As an administrator, I want to have administrative privileges to manage user accounts, handle authentication and authorization processes, and ensure the overall security of the job portal.

Help desk
A help desk in a job portal plays a crucial role in assisting users with their queries, technical issues, and providing support throughout their job search process. User stories are a valuable tool used in software development to capture requirements from the perspective of end-users. In the context of a help desk in a job portal, user stories can be used to outline the various scenarios and interactions between users and the help desk team. Here are some user stories that highlight common situations and requests encountered by a help desk in a job portal:

1. As a job seeker, I want to reset my password:
- Description: As a user who has forgotten or wants to change their password, I need assistance from the help desk to reset it.
- Acceptance Criteria: The help desk should provide clear instructions on how to reset the password, including any necessary verification steps.
2. As an employer, I want to post a job listing:
- Description: As an employer looking to hire new talent, I need guidance from the help desk on how to create and publish a job listing on the portal.
- Acceptance Criteria: The help desk should provide step-by-step instructions on creating a job listing, including details such as required information, formatting guidelines, and payment options if applicable.

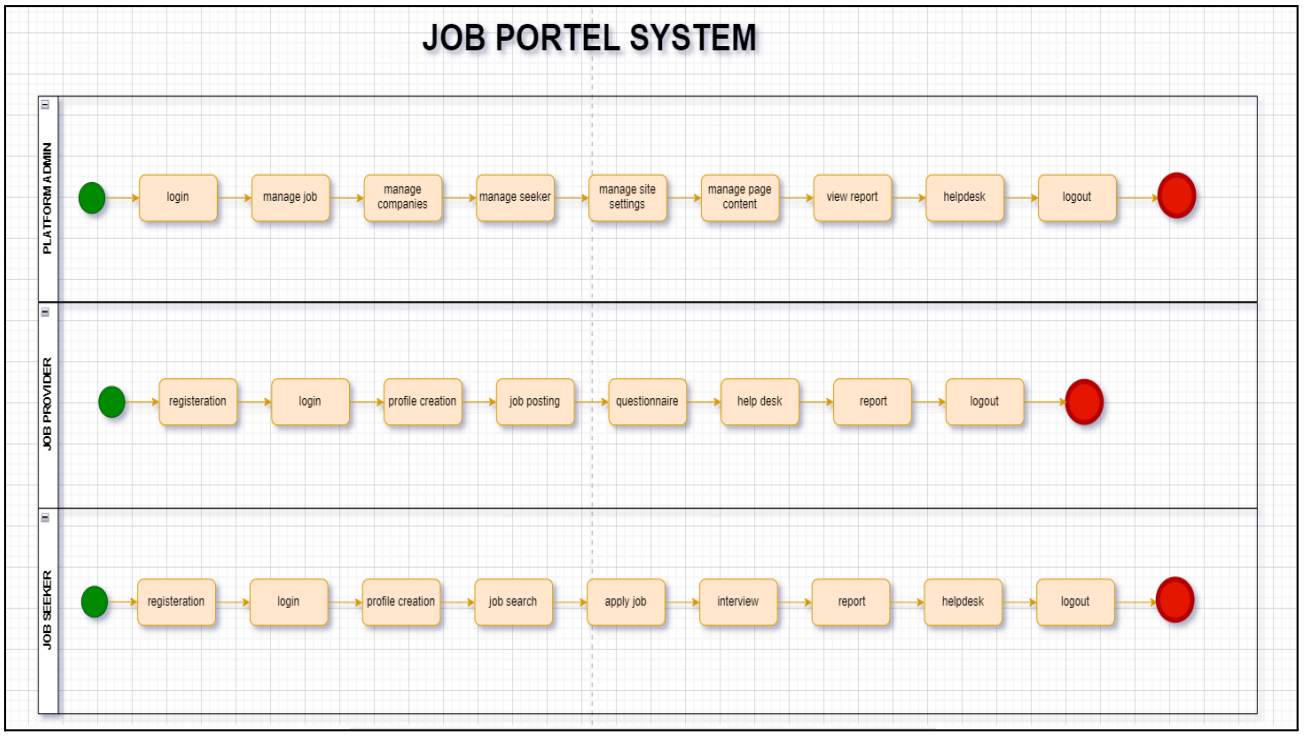
3. As a job seeker, I want to update my profile information:
- Description: As a user who wants to modify personal details or add new qualifications/experience to my profile, I require assistance from the help desk.
- Acceptance Criteria: The help desk should guide users on how to access and edit their profile information, ensuring that changes are accurately reflected on the portal.

4. As an employer, I want to search for resumes/candidates:
- Description: As an employer seeking suitable candidates for my job openings, I need support from the help desk to effectively search and filter resumes on the job portal.
- Acceptance Criteria: The help desk should provide guidance on utilizing search filters, keywords, and advanced search options to find relevant resumes/candidates.

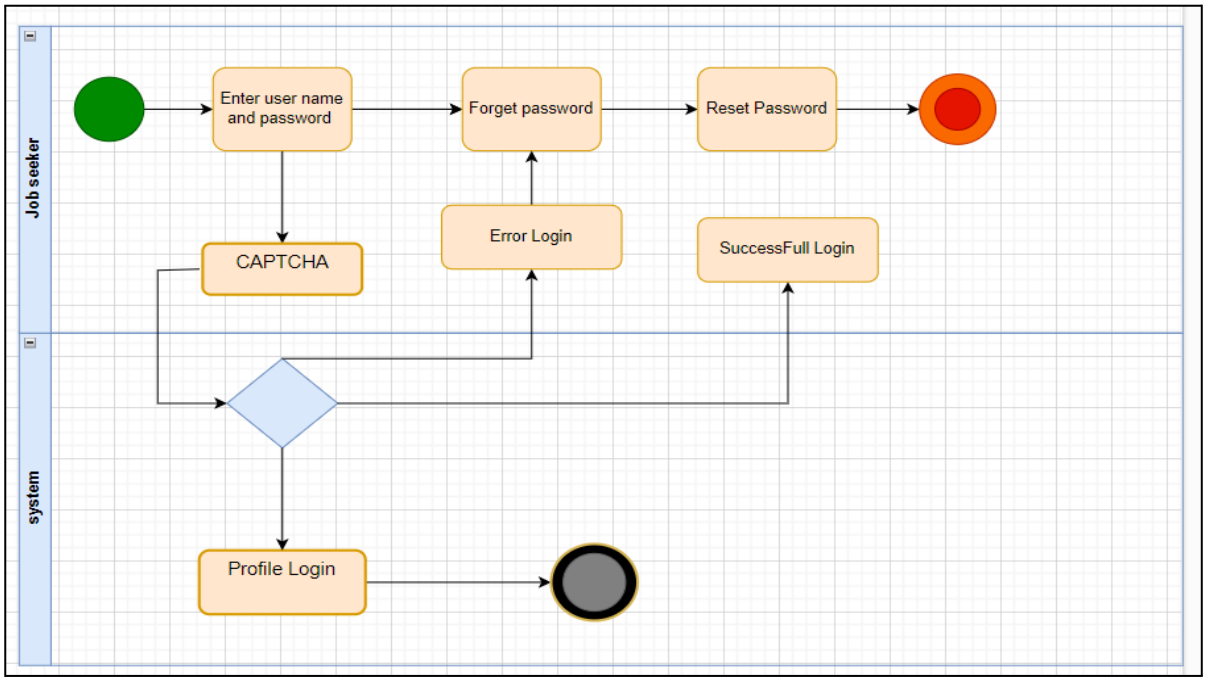
These user stories represent common scenarios encountered by users of a help desk in a job portal. By capturing these requirements in user stories, development teams can better understand user needs and prioritize their efforts accordingly.

BUSINESS PROCESS DIAGRAM

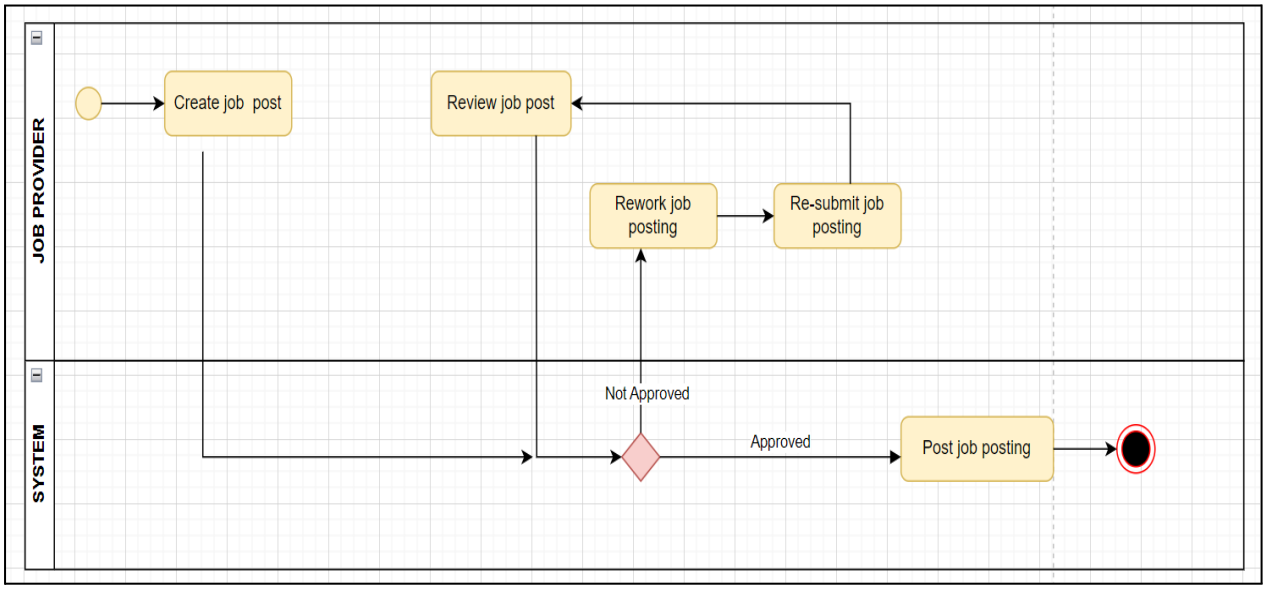
Here are some examples



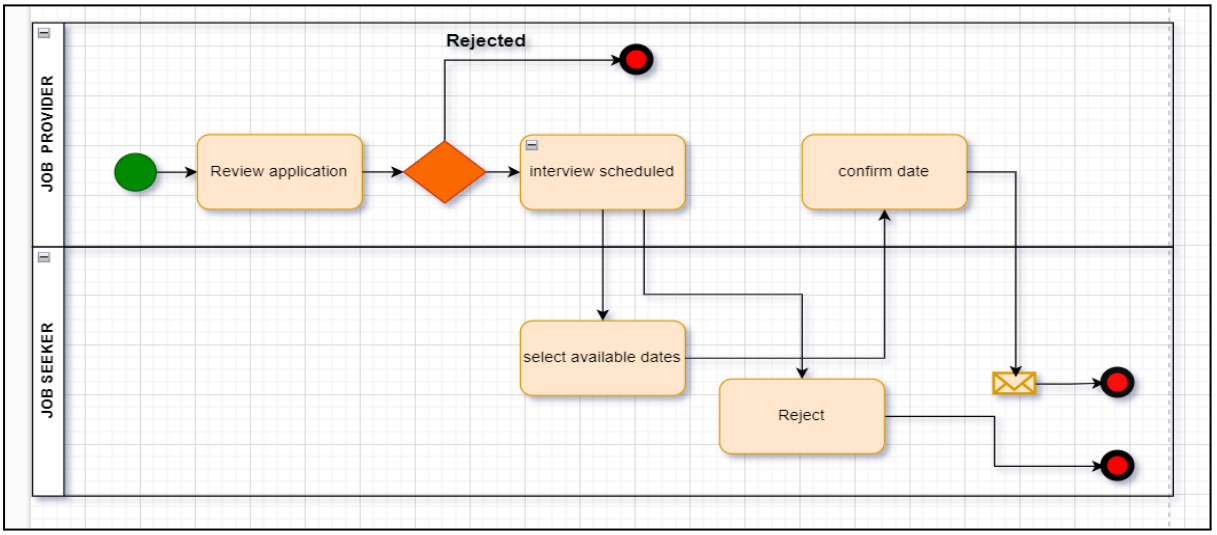
LOGIN



JOB POSTING



INTERVIEW



NON FUNCTIONAL REQUIRMENTS

When designing a system, it is crucial to consider not only the functional requirements but also the non-functional requirements. Non-functional requirements define the qualities and attributes of a system that are essential for its successful operation. These requirements encompass various aspects such as performance, reliability, security, usability, and more. In this comprehensive response, we will identify and specify some of the key non-functional requirements commonly considered in system development.

1. Performance:

Performance refers to how well a system performs its intended functions within specified constraints. It involves factors such as response time, throughput, scalability, and resource utilization. To ensure optimal performance, the system should be designed to handle expected workloads efficiently and effectively. This includes minimizing response times, maximizing throughput, and optimizing resource usage to meet user expectations.

2. Reliability:

Reliability is the ability of a system to perform its functions consistently and accurately over time. It involves factors such as availability, fault tolerance, error handling, and recoverability. A reliable system should minimize downtime, prevent data loss or corruption, and provide mechanisms for error detection and recovery. It should also have appropriate backup and restore procedures in place to ensure data integrity.

3. Security:
Security is of utmost importance in any system to protect sensitive information from unauthorized access, modification, or destruction. It encompasses aspects such as confidentiality, integrity, authentication, authorization, and auditability. The system should implement robust security measures like encryption, access controls, secure communication protocols, and secure storage mechanisms to safeguard data and prevent security breaches.

4. Usability:
Usability refers to how easily users can interact with the system and accomplish their tasks efficiently. It includes factors such as user interface design, ease of learning, navigation simplicity, and error prevention. A usable system should have an intuitive interface that requires minimal training for users to understand and operate effectively. It should also provide clear feedback and guidance to users throughout their interactions.

5. Maintainability:
Maintainability is the ease with which a system can be modified, enhanced, or repaired over its lifecycle. It involves factors such as modularity, extensibility, testability, and documentation. A maintainable system should have well-structured and modular code that allows for easy modifications without affecting other components. It should also provide comprehensive documentation to aid in understanding and maintaining the system.

6. Scalability:
Scalability refers to the ability of a system to handle increasing workloads or accommodate growth without significant performance degradation. It involves factors such as load balancing, horizontal or vertical scaling, and resource allocation. A scalable system should be designed to handle increased user demand by efficiently distributing the workload across multiple resources or by adding additional resources as needed.

7. Interoperability:
Interoperability is the ability of a system to interact and exchange data with other systems or components seamlessly. It involves factors such as adherence to standards, compatibility with different platforms or technologies, and support for data exchange formats. An interoperable system should have well-defined interfaces and protocols that enable seamless integration with other systems or components.

8. Compliance:
Compliance refers to the adherence of a system to relevant laws, regulations, industry standards, and organizational policies. It involves factors such as data privacy, data protection, accessibility, and ethical considerations. A compliant system should ensure that it meets all legal and regulatory requirements while also considering ethical implications related to user privacy and data protection.

In conclusion, non-functional requirements play a crucial role in the design and development of a system. By considering aspects such as performance, reliability, security, usability, maintainability, scalability, interoperability, and compliance, developers can create systems that meet user expectations while ensuring optimal operation and protection of sensitive information.

TECHNICAL REQUIREMENTS

Softwares

HTML
CSS
BOOTSTRAP
ANGULAR
ASP.NET RAZOR
ASP.NET BLAZER
ASP.NET ANGULAR
SWAGGER

Hardwares

Ram: 8gb
Display: 2560 x 1600
Hard disk: 100gb
Processor: i5 or i7
Os: Windows 11

CONCLUSION

The conclusion of a Software Requirements Specification (SRS) is a crucial section that summarizes the key findings and recommendations of the document. It serves as a final statement that outlines the main objectives, scope, and deliverables of the software project. The conclusion of an SRS typically includes a summary of the requirements, an evaluation of the feasibility and risks associated with the project, and any additional recommendations or suggestions.

In the conclusion section of an SRS, it is important to reiterate the main goals and objectives of the software project. This helps to ensure that all stakeholders have a clear understanding of what needs to be achieved. The conclusion should also summarize the functional and non-functional requirements identified throughout the document. This includes outlining the specific features, functionalities, and performance expectations that the software should meet.

Additionally, the conclusion should address the feasibility and risks associated with implementing the software project. This involves evaluating whether the proposed solution is technically feasible within the given constraints such as time, budget, and resources. It also involves identifying any potential risks or challenges that may arise during development or implementation and suggesting mitigation strategies to address them.