Functional Requirements Document (FRD)

Project Title :DEMO-PROJECT
Date: July23, 2025

Prepared By : John

1. Purpose

The purpose of this document is to define the functional and non-functional requirements for the DEMO-PROJECT application. This system aims to unify operational visibility and support workflows by integrating various platforms like UC4, ServiceNow and Informatica under a single frontend. DEMO-PROJECT will enhance efficiency, improve turnaround times, and provide a centralized repository for operational data, logs, incidents, and change requests.

This FRD serves as the foundation for the development, testing, and deployment of DEMO-PROJECT, ensuring all stakeholders have a shared understanding of its functionalities and objectives.

2. Project Overview

2.1 Objective

To develop a Java-based web application that provides end-to-end operational visibility and job support functionality by integrating disparate systems (UC4, ServiceNow and Informatica) into a unified platform for DEMO-CLIENT.

2.2 Scope

- Development of a secure, role-based web application.
- Integration with external platforms to pull job data, incidents, change requests, and run statistics.
- User-friendly dashboards and reporting mechanisms using embedded Power BI.
- Centralized support documentation repository.

2.3 Out of Scope

- Bi-directional updates to external systems.
- Mobile app development.
- Infrastructure provisioning and CI/CD setup.
- Production deployments.

2.4 Assumptions

- External system integrations will provide necessary APIs and data access.
- DEMO-CLIENT will handle CI/CD pipeline setup and production deployments.
- Authentication will use DEMO-CLIENT's existing Okta integration.

3. Functional Modules

3.1 Menu Structure

Menu Item	Submenus/Features
Home	Search Job, Subject Area List, Recent Incidents, Change Requests, Rally Tickets
Dashboard	System health overview, alerts, and key metrics
Subject Area	List and manage subject areas for job categorization
Jobs List	View all jobs with filters, sorting, and search
Change Requests	List, view, and manage change requests
Service Now	View incidents and tickets from ServiceNow
Reports	Embedded Power BI reports
Settings	User Management, Role Management, Report Management, Support Templates

3.2 Design Patterns

- Collapsible sidebar with icons and labels.
- Highlight active menu items.
- Role-based visibility (e.g., only Admins see Settings).

3.3 Home Page Features

- **Search Job:** (autocomplete, has filters) Search by job name, ID, or subject area.
- **Subject Area List:** Quick links to jobs grouped by subject area.

- **Recent Incidents:** Display user or team-specific incidents from ServiceNow.
- **Change Requests:** Show pending or active CRs relevant to the user/team.
- **Rally Tickets:** Display recent Rally tickets assigned to the user or team.

3.4 Job Detail Page

Section	Description
Job Information	Job name, description, status, owner, schedule
Job Support Instructions	Documentation and troubleshooting steps
Error Logs	Error messages and logs from UC4 and Informatica
Associated Incidents	Linked ServiceNow incidents
Change Requests	Change requests and deployment details
Job Schedule	Next run, last run, frequency, timing
Job Run Stats	Metrics on success/failure rates, processing time

1. Non-Functional Requirements

Requirement	Details
Performance	System should support concurrent users with minimal response time.
Scalability	Support scaling to multiple users as needed.
Security	Role-based access control (RBAC); Okta integration.
Availability	99.9% uptime, hosted on DEMO-CLIENT's Tanzu platform.
Logging and Monitoring	Centralized logging with error tracking.
Data Refresh	Batch and near real-time data refresh from snowflake database.
Browser Compatibility	Chrome, Edge.

2. User Roles and Permissions

Role	Permissions
Administrator	Full access to all modules, user management, role management, and report configs.
Developer	Access to view/edit job details, add support docs, and view logs/incidents.

Role	Permissions
Read-Only User	View-only access to all modules and reports.
Support Engineer	Edit support documentation and log resolutions.

3. Dependencies

Dependency	Owner
Snowflake DB setup and data population	DEMO-CLIENT ETL Team
API access to UC4, ServiceNow	DEMO-CLIENT
CI/CD pipeline setup	DEMO-CLIENT
Okta SSO configuration	DEMO-CLIENT
VPN Access	DEMO-CLIENT
Access to GitHub, ServiceNow, Rally	DEMO-CLIENT

4. Pre-Requisites

- VPN access for Demo-Company resources.
- Access credentials for all external systems (ServiceNow, UC4, Rally, Snowflake).
- Architecture requirements and documentation from DEMO-CLIENT teams.

5. Reporting Requirements

- Power BI Embedded Reports

6. Governance & Risk Management

- Weekly governance calls with DEMO-CLIENT stakeholders.
- Calculating and discussing risks with mitigation plans.
- Demo-Company team to participate in DEMO-CLIENT agile ceremonies if any.

7. Okta Integration Analysis

Option - 1

1. **Front End (Vue.js) -**
    a. Set up the Client ID and Issuer URL on the front end.
    b. Add dependencies for Okta Vue and Okta SDK.
    c. Open the Okta login page (with client logo) using the Issuer URL.
    d. Send a login request to Okta with credentials and Client ID.
    e. Store the access token and user details extracted from the token in a Pinia store, and control page content based on the role stored in the Pinia store
    f. Include the access token in the headers of each request.

2. **Back End (Spring Boot) -**
    a. Secure all REST APIs.
    b. Create a security configuration class to customize security-related configurations.
    c. Validate the access token.
    d. Create an API to retrieve user information (if required)

3. **Okta**

a. Create a free demo account and obtain the Issuer URI along with the Client ID.
b. Set the roles and user info inside claims in okta account.

Option - 2

1. **Front End (Vue.js) -**
    a. Very minimal responsibility — it's just a UI that relies on session-based authentication handled by the backend.

2. **Back End (Spring Boot) -**
    a. Open the Okta login page from the backend.
    b. Set the successful login redirect URL and logout URL.
    c. Authenticate the user using login credentials and manage the session and token within the JSESSIONID.
    d. Protect routes using Spring Security.
    e. Provide a user info endpoint to fetch user details.
    f. Provide a token endpoint to retrieve token details (if needed).
    g. On logout, clear the session and remove cookies.

3. **Okta**

a. Create a free demo account and obtain the Issuer URI along with the Client ID.
b. Set the roles and user info inside claims in okta account.