# introduction

Overview

The self-service facility is comprehensive platform designed to manage systematic distribution or sharing of binary file to various vendor and internal teams. This documentation aims to guide users and administrators through the setup, usage, and maintenance of the software.

Target Audience

This documentation is intended for system administrators, developers, and end-users who will interact with the Self-Service Facility.

# getting started

System

AWS Server Details….

# Architecture overview

Components

* Frontend: Angular
* Backend: Spring Boot (Monolithic Architecture)
* File Storage: AWS S3
* Notification Service: AWS SQS/SNS
* Authentication & Authorization: Azure AD
* Security: DMZ
* Load Balancer: AWS ELB
* Database: Relational database (PostgreSQL, etc.)
* Additional Component: Nginx for additional functionality.

# system Components

### 3.1 Frontend (Angular)

#### Description:

The Angular-based frontend provides the user interface for interacting with the application.

#### Components:

* **User Interface (UI)**:
  + Allows users to upload various file types, including documents and binaries.
  + Provides a seamless experience for file sharing and collaboration.
* **Version Control**:
  + Incorporates versioning functionality for uploaded documents, ensuring users can manage different iterations.

#### Interaction with Backend:

* Communicates with the backend via RESTful APIs to perform file uploads, share documents, and manage versioning.

### 3.2 Backend (Spring Boot)

#### Description:

The Spring Boot backend manages the application logic, data storage, and serves as the communication hub for various components.

#### Components:

* **RESTful API Endpoints**:
  + Provides endpoints for file uploads, file sharing, and version management.
* **File Storage Integration**:
  + Integrates with AWS S3 to facilitate file storage and retrieval.
* **Database Integration**:
  + Utilizes a relational database to store metadata related to uploaded files, enabling efficient data management.

#### Interaction with Other Components:

* Communicates with AWS S3 for file storage.
* Integrates with the relational database for metadata management.
* Triggers notifications through AWS SQS/SNS based on specific events (e.g., file uploads, shares, updates).

### 3.3 Authentication & Authorization (Azure AD)

#### Description:

Azure AD manages user authentication and controls access to the application's functionalities.

#### Components:

* **User Authentication**:
  + Authenticates users accessing the application.
* **Role-Based Access Control (RBAC)**:
  + Controls user permissions based on predefined roles and access levels.

#### Interaction with the System:

* Validates user credentials during login and access requests.
* Implements role-based access control to determine user privileges within the application.

### 3.4 AWS Services Integration

#### AWS S3 (File Storage)

#### Description:

AWS S3 provides scalable and secure storage for uploaded files within the application.

#### Components:

* **File Storage and Management**:
  + Stores uploaded documents and binary files securely.
* **Versioning and Security**:
  + Implements version control for documents and ensures secure storage with encryption and access policies.

#### Interaction with Other Components:

* Receives file uploads from the backend for storage.
* Integrates with backend services for version control and security implementations.

#### AWS SQS/SNS (Notification Service)

#### Description:

AWS SQS/SNS enables the application to send notifications based on specific events.

#### Components:

* **Event-Driven Notifications**:
  + Triggers notifications for events like file uploads, shares, and updates.
* **Integration with Backend**:
  + Communicates with the backend to process events and send notifications accordingly.

#### Interaction with Other Components:

* Listens to events generated within the system (e.g., file uploads) and triggers notifications.
* Interacts with backend services to process notification-triggering events.

### 3.5 Security (DMZ)

#### Description:

The DMZ establishes a secure boundary between the internal network and external users, ensuring data protection and access control.

#### Components:

* **Network Segregation**:
  + Separates internal and external traffic, enhancing security.
* **Firewall Configuration**:
  + Implements rules and policies to filter incoming and outgoing traffic.

#### Interaction with the System:

* Controls and filters incoming and outgoing traffic to protect against unauthorized access or malicious activities.

### 3.6 Relational Database

#### Description:

The relational database stores metadata related to uploaded files for efficient data management.

#### Components:

* **Data Storage**:
  + Stores metadata associated with uploaded documents and their versions.
* **Backend Integration**:
  + Integrates with the backend to provide data access and manipulation.

#### Interaction with Other Components:

* Provides a storage repository for metadata required for document management within the application.
* Communicates with the backend to retrieve or update metadata as necessary.

### 3.7 Nginx

#### Description:

Nginx acts as a reverse proxy server and provides additional functionalities to the application.

#### Components:

* **Reverse Proxy Server**:
  + Acts as an intermediary between the frontend and backend, optimizing traffic flow.
* **Load Balancing and Caching**:
  + Optimizes performance by distributing traffic and caching frequently accessed resources.

#### Interaction with the System:

* Routes and manages incoming requests between the frontend and backend components.
* Improves application performance through load balancing and caching mechanisms.

# user management

Authentication Methods

We are going to user Azure AD for authetication

# functionality

# integration

# security

# troubleshooting

# support and resources