Software Requirements Specification

for

Enhancement on pgAgent Features

(In collaboration with IITM Pravartak)

Version 1.0

Prepared by
Brian Christopher
Joel Daniel Pradeep

TKM COLLEGE OF ENGINEERING

30 January 2025

Table of Contents

1. Introduction

1.1 Purpose

This document outlines the functional and non-functional requirements for enhancing pgAgent with Job Dependency and Role-Based Access Control (RBAC) features. The primary objectives are:

- ***Job Dependency:*** *Enable the scheduling of jobs based on the completion status of other jobs.*
- ***RBAC:*** *Introduce user roles with controlled access to job creation, execution, and monitoring.*

These enhancements aim to improve job execution workflows, enforce security measures, and optimize job scheduling efficiency in PostgreSQL environments.

1.2 Project Scope

The system extends pgAgent functionalities by implementing:

- ***Job Dependency:*** *Supporting sequential, conditional, and parallel job dependencies.*
- ***RBAC:*** *Defining role-based permissions for different user levels, ensuring controlled access.*

This enhancement will facilitate efficient database task automation and improve system security and job management flexibility.

1.3 References

The following references provide guidelines, best practices, and solutions relevant to the project:

1. ***PostgreSQL Official Documentation***
   - *Comprehensive resource for understanding PostgreSQL's features, including job management and role-based access controls.*
   - *Covers details on extensions like pgAgent and built-in role management capabilities.*
2. ***NIST Access Control Guidelines***
   - *Provides industry-standard practices for implementing Role-Based Access Control (RBAC).*
   - *Focuses on secure and scalable role hierarchies and granular access controls.*
3. ***Apache Airflow Documentation***
   - *Airflow is a workflow orchestration platform offering robust support for defining job dependencies using Directed Acyclic Graphs (DAGs).*
   - *A relevant reference for implementing job dependency management effectively.*
4. ***SQL Server Agent Documentation***
   - *SQL Server Agent supports job scheduling and dependency management via configurable steps and triggers.*
   - *A practical example of dependency handling in a database context.*
5. ***PostgreSQL pgAgent Documentation***
   - *Documentation for pgAgent, the job scheduling agent for PostgreSQL.*
   - *Details existing job management features and provides a foundation for enhancements.*
6. ***AWS Step Functions***
   - *AWS Step Functions is a serverless workflow service that offers flexible dependency and state management.*
   - *Serves as an example for dynamic dependency resolution and error handling.*

2. System Overview

2.1 System Perspective

The enhancement integrates with existing pgAgent functionalities, extending its capability to support complex job dependencies and secure user access control.

2.2 Core Functionalities

- ***Job Dependency:***
  - *Define dependencies between jobs.*
  - *Set conditions such as "execute only if the parent job succeeds."*
  - *Support parallel execution of dependent jobs.*
- ***RBAC:***
  - *Create roles such as Admin, Developer, and Viewer.*
  - *Restrict job creation and execution based on roles.*
  - *Ensure secure access control via authentication mechanisms.*

2.3 User Classes and Characteristics

- ***Database Administrators:*** *Define job dependencies and manage user roles.*
- ***Developers:*** *Implement job workflows based on dependency conditions.*
- ***Operators:*** *Monitor job execution and manage job failures.*

2.4 Design and Implementation Constraints

- *Must adhere to PostgreSQL security guidelines.*
- *Ensure minimal performance impact due to dependency checking mechanisms.*
- *Provide backward compatibility with existing pgAgent deployments.*

3. External Interface Requirements

3.1 Hardware Interfaces

- *Supports multi-core CPU environments for parallel job execution.*
- *Requires sufficient memory for handling multiple dependent jobs efficiently.*

3.2 Software Interfaces

The system will interact with the following components:

- **PostgreSQL 14+** for job execution.
- **pgAdmin 4** for job monitoring and role management.
- **pgAgent** for job scheduling.
- **Authentication Systems** for enforcing RBAC.

3.3 Communication Interfaces

- ***Secure Network Protocols:*** *TLS 1.2+ for encrypted communication.*
- ***REST APIs:*** *For external system integration with job dependency management and RBAC.*
- ***Alerts and Notifications:*** *Email and webhook notifications for job execution failures or security breaches.*

4. Functional Requirements

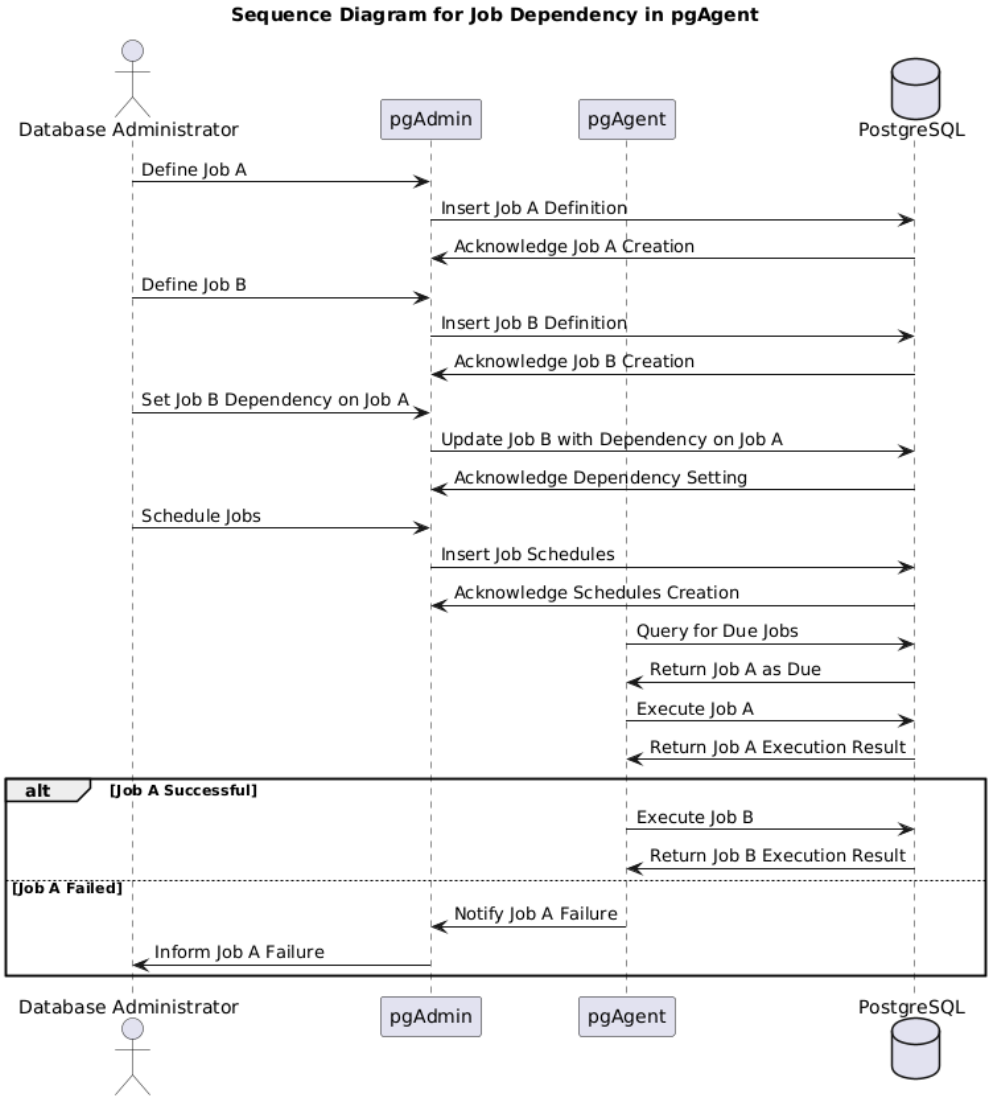
4.1 Job Dependency

Description:
Allows users to configure job execution based on dependencies.

Priority: *High*

Functional Requirements:

- *FR-1: Support sequential and conditional job execution.*
- *FR-2: Allow parallel execution of dependent jobs.*
- *FR-3: Provide visual representation of job dependencies in pgAdmin*



4.2 Role Based Access Control

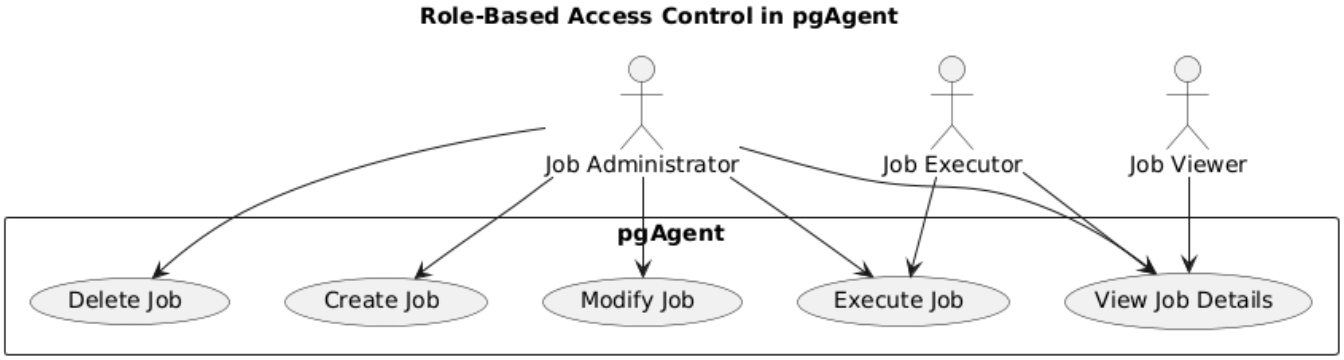
Description:
Implements role-based permissions for secure access control.

Priority: *Critical*

Functional Requirements:

- *FR-4: Define roles (Admin, Developer, Viewer) with specific permissions.*
- *FR-5: Enforce authentication for role assignment and job execution.*
- *FR-6: Restrict job editing and deletion based on roles.*

Usecase Diagram



5. Nonfunctional Requirements

5.1 ***Performance***

- *Ensure minimal latency in checking job dependencies and enforcing RBAC policies.*
- *Optimize queries and indexing for efficient job scheduling.*
- *Benchmark system performance under high job execution loads.*

5.2 ***Scalability***

- *Support an increasing number of job dependencies without significant performance degradation.*
- *Ensure RBAC policies scale for enterprise-level users.*

5.3 ***Security***

- *Use AES-256 encryption for stored job details.*
- *Enforce RBAC policies using secure authentication mechanisms.*
- *Implement logging and auditing of user activities.*

5.4 ***Availability***

- *Ensure a minimum uptime of 99.9% for job execution and monitoring services.*

6. System Architecture and Design

6.1 Architectural Overview

The system adopts a modular architecture that integrates seamlessly with PostgreSQL and pgAgent. The architectural components include:

- *Job Dependency Manager: Handles dependency rules and execution order.*
- *RBAC Module: Manages user roles and access control policies.*
- *Database Layer: Stores job details, dependency information, and role configurations.*
- *Interface Layer: Provides APIs and pgAdmin UI integration for job dependency visualization and*

6.2 System Flow

1. *Job Dependency:*
    - *Define job dependencies in pgAdmin.*
    - *Store dependency rules in the database.*
    - *Execute dependent jobs sequentially, conditionally, or in parallel.*
2. *RBAC:*
    - *Authenticate users via secure mechanisms.*
    - *Check roles before granting access to job creation, execution, or monitoring.*
    - *Log all user activities for auditing.*

7. Implementation Plan

7.1 Phases of Development

The project will be developed in the following phases:

Phase 1: Requirements Gathering and Analysis *(Duration: 2 weeks)*

Phase 2: Role-Based Access Control *(Duration: 3 weeks)*

Phase 3: Job-Dependency *(Duration: 2 weeks)*

Phase 4: System Testing and Optimization *(Duration: 2 weeks)*

7.2 Implementation Steps

- **Step 1: Requirement Analysis**

Analyze the scheduling and access control requirements by collaborating with stakeholders. Identify use cases for job dependencies and RBAC.

- **Step 2: System Design**

Design the system architecture, database schema for job dependencies and roles, and workflows for job execution and access control.

- **Step 3: Job Dependency Development**

Implement functionalities to define job dependencies, set conditions, and allow parallel or sequential execution of jobs.

- **Step 4: RBAC Development**

Develop the RBAC module, including role creation, role-based permissions, and secure user authentication mechanisms.

- **Step 5: Integration**

Integrate the enhancements with pgAdmin for visualization and external monitoring systems like Prometheus and Grafana.

- **Step 6: Testing**

Perform functional testing for job dependency workflows and RBAC policies, performance testing under high job loads, and security testing for access control mechanisms.

- **Step 7: Deployment**

Deploy the solution in a production-like environment, conduct live testing, and ensure the system meets all functional and nonfunctional requirements.

---

8. Test Plan

8.1 Testing Objectives

- ***Verify job dependency execution workflows.***
- ***Validate role-based access control for different user roles.***
- ***Ensure system stability and performance under high job loads.***

8.2 Testing Strategy

- ***Functional Testing: Ensure job dependencies (sequential, conditional, and parallel) and RBAC features work as intended.***
- ***Performance Testing: Test the system's efficiency under heavy job scheduling and execution loads.***
- ***Security Testing: Validate secure authentication mechanisms, role-based permissions, and data encryption for job details.***
- ***Usability Testing: Ensure the pgAdmin UI for dependency visualization and role management is user-friendly.***

8.3 Test Cases

- ***TC-1: Verify that jobs are executed in the correct order based on defined dependencies.***
- ***TC-2: Check if a dependent job executes only when the parent job succeeds.***
- ***TC-3: Validate parallel execution of dependent jobs without conflicts.***
- ***TC-4: Test access restrictions for different roles (Admin, Developer, Viewer).***
- ***TC-5: Verify job editing and deletion permissions based on user roles.***
- ***TC-6: Check the system's response when a parent job in a dependency chain fails.***
- ***TC-7: Test system behavior under a high number of concurrent job executions.***
- ***TC-8: Validate audit logs for tracking user activities and RBAC policy violations.***
- ***TC-9: Test proper notifications for job failures or access violations.***
- ***TC-10: Check the encryption of job details in the database and secure communication channels.***

9. Conclusion and Future Scope

9.1 Summary

The proposed enhancements to pgAgent, specifically the implementation of Job Dependency and Role-Based Access Control (RBAC), aim to significantly improve the usability, efficiency, and security of PostgreSQL job scheduling. By introducing advanced dependency management, users can schedule jobs based on sequential, conditional, or parallel execution requirements, optimizing workflows for complex database tasks. RBAC ensures that only authorized users can access specific functionalities, enhancing security and reducing the risk of accidental or malicious actions. These features, combined with seamless integration into existing pgAgent systems, provide a robust solution for database administrators, developers, and operators to manage and monitor jobs effectively.

9.2 Future Enhancements

While the current implementation addresses the core needs of job monitoring and scheduling, several areas for future development could further enhance the system's functionality:

1. ***Enhanced Visualization: Incorporate advanced graphical representations for job dependencies, such as Gantt charts or dependency trees, for better monitoring and debugging.***
2. ***Dynamic Role Management: Expand RBAC to allow customizable roles and permissions tailored to organizational needs.***
3. ***Real-Time Alerts: Introduce more detailed notification systems, including SMS and push notifications, for job status updates and security alerts.***
4. ***Improved Scalability: Optimize the system further to handle thousands of concurrent jobs and users in large-scale enterprise environments.***