Software Requirements Specification

for

Enhancement on pgAgent Features
(in Collaboration with IITM Pravartak)

Version 1.0

Prepared by
Deepak S
Muhzin Mohammed

TKM COLLEGE OF ENGINEERING

30 January 2025

Table of Contents

Table of Contents .... **ii**
Revision History .... **ii**
1. Introduction **2**
1.1 Purpose 2
1.2 Project Scope 2
1.3 References 2
2. System Overview **4**
2.1 System Perspective 4
2.2 Core Functionalities 4
2.3 User Classes and Characteristics 3
2.4 Design Implementation Constraints 4
3. External Interface Requirements **5**
3.1 Hardware Interfaces 5
3.2 Software Interfaces 5
3.3 Communications Interfaces 5
4. System Features **6**
4.1 Real-Time job Status 6
4.2 Advanced Scheduling Options 7
5. Nonfunctional Requirements **8**
5.1 Performance 8
5.2 Scalability 8
5.3 Security 8
5.4 Availability 8
6. System Architecture and Design **9**
6.1 Architectural Overview 9
6.2 System Flow 9
7. Implementation Plan **10**
7.1 Phases of Development 10
7.2 Implementation Steps 10
8. Test Plan **11**
8.1 Testing Objective 11
8.2 Testing strategy 11
8.3 Test Cases 11
9. Conclusion and Future Scope **12**
9.1 Summary 12
9.2 Future Enhancement 12

1. Introduction

1.1 Purpose

This document outlines the functional and non-functional requirements for enhancing the real-time job status monitoring and advanced scheduling features in collaboration with IITM. The primary goal is to augment the capabilities of ***pgAdmin*** *and* ***pgAgent*** *by providing:*

- *Real-time job execution updates for better monitoring of jobs.*
- *Support for complex job scheduling patterns to handle diverse and intricate job execution schedules.*

This enhancement aims to streamline the management of PostgreSQL jobs, making it easier for database administrators to monitor, control, and schedule database tasks effectively.

1.2 Project Scope

The system aims to extend the capabilities of ***pgAdmin*** *and* ***pgAgent*** *by:*

- *Displaying real-time updates for job execution in the pgAdmin interface.*
- *Introducing advanced scheduling options, allowing complex, customizable job schedules like recurring jobs on specific days or intervals.*

By implementing these features, the system will significantly improve operational efficiency, helping to automate and optimize the scheduling of routine database maintenance tasks.

1.3 References

1. ***Prometheus Toolkit*** *– Open-source system monitoring and alerting toolkit that integrates with databases for real-time metric tracking.*
2. ***DBMS_SCHEDULER Documentation*** *– Oracle's job scheduling system, useful for implementing advanced scheduling patterns and real-time monitoring.*
3. ***PostgreSQL Official Documentation*** *– A comprehensive guide for managing job scheduling and monitoring using PostgreSQL, pgAdmin, and pgAgent.*
4. **Research Paper: "Efficient Monitoring and Scheduling in Distributed Databases"**

2. System Overview

2.1 System Perspective

This system builds upon the existing features of ***pgAdmin*** *and* ***pgAgent****, offering enhancements to improve the monitoring and scheduling of jobs in a PostgreSQL environment. The core improvements are:*

- ***Real-time Job Status****: Users can view the real-time status of job execution directly in the pgAdmin interface.*
- ***Advanced Scheduling****: The system allows users to define custom job schedules using an intuitive interface, which supports complex intervals like weekly, monthly, or yearly recurrences.*

2.2 Core Functionalities

The system is designed to provide the following capabilities:

- ***Real-Time Job Status****: Displays live updates on job execution, giving users immediate feedback on job progress, completion, or failure.*
- ***Advanced Scheduling****: The system allows users to define custom, complex job schedules through an intuitive interface, including advanced options like recurring jobs based on custom intervals.*

2.3 User Classes and Characteristics

The primary users of this system are:

- ***Database Administrators****: They will use the system to monitor and troubleshoot job executions and optimize scheduling to meet system needs.*
- ***Developers****: They will be responsible for implementing and customizing the new features and ensuring their integration with existing systems.*
- ***Project Managers****: They will oversee the project, ensuring timely delivery and meeting the system’s requirements*

  .

2.4 Design and Implementation Constraints

- *The system must comply with industry standards for* ***database management*** *to ensure that the implementation does not compromise the security, integrity, or performance of the database.*
- *The solution should minimize the impact on database* ***performance*** *during job execution, ensuring that real-time job updates do not degrade overall database operations.*

3. External Interface Requirements

3.1 Hardware Interfaces

The system will support standard database server configurations, including:

- *Multi-core CPUs for efficient parallel processing of tasks.*
- *High-memory servers to ensure optimal performance and scalability, especially in enterprise environments.*

3.2 Software Interfaces

The system will interact with the following components:

- ***PostgreSQL 14+**: The core relational database management system supporting job scheduling and execution.*
- ***pgAdmin 4**: The graphical user interface used for displaying real-time job execution statuses.*
- ***pgAgent**: The agent responsible for handling advanced job scheduling and execution in PostgreSQL.*
- ***External Monitoring Systems**: The system will be capable of integrating with **Prometheus** and **Grafana** for enhanced monitoring capabilities.*
- ***Database Extensions**: The system will support extensions like **pg_stat_statements** to help optimize query performance and improve database operations.*

3.3 Communication Interfaces

- ***Network Protocols**: The system will use secure communication channels, employing **TLS 1.2 or higher** to protect data during transmission.*
- ***Data Encryption**: Sensitive data transmitted and stored by the system will be encrypted using **AES-256** to ensure confidentiality and integrity.*
- ***Alerting and Notifications**: The system will support **email, SMS, and webhook-based notifications** for real-time alerts about job execution statuses (success, failure, etc.).*
- ***REST APIs**: The system will expose REST APIs to interact with external monitoring and alerting systems, enabling real-time data integration.*

4. Functional Requirements

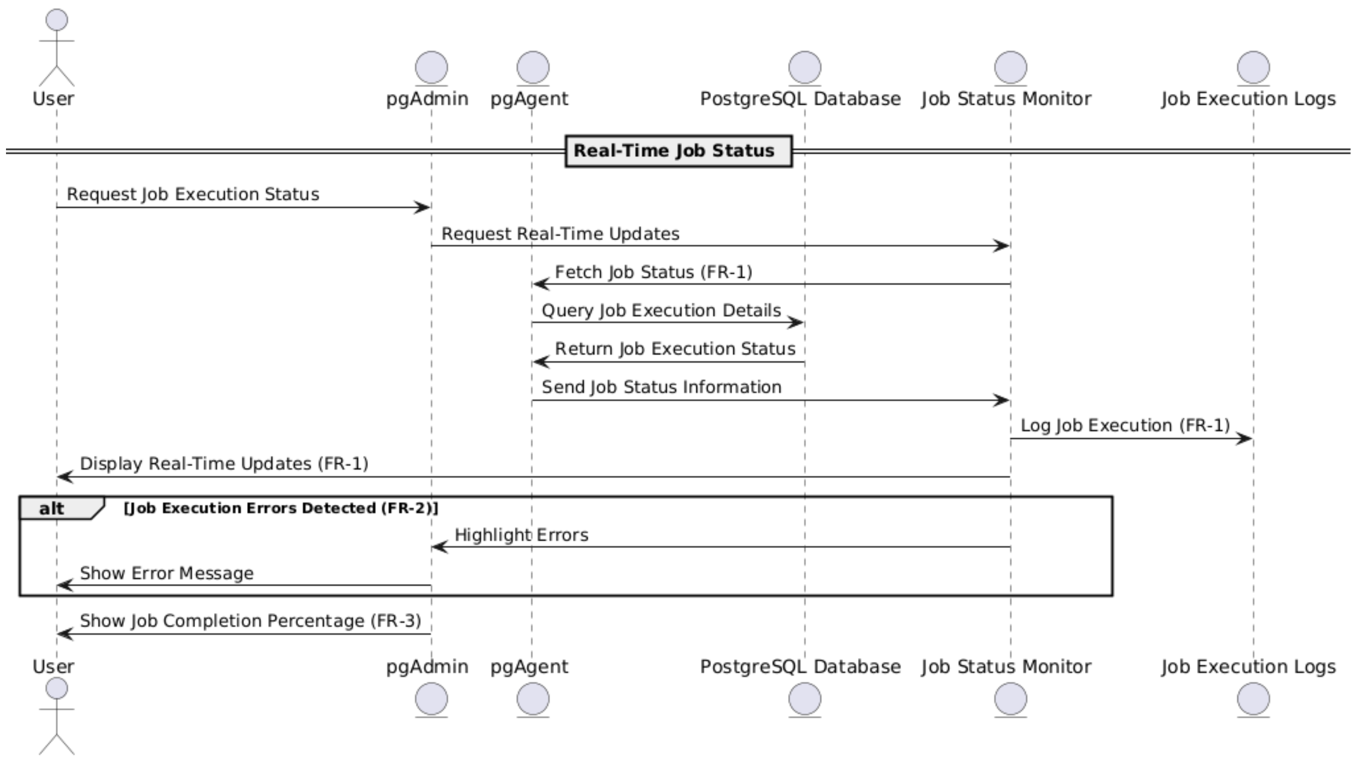
4.1 Real-Time Job Status

Description
Displays real-time job execution updates in the pgAdmin interface.

Priority
High

Functional Requirements
FR-1: The system must provide real-time updates for job execution status.
FR-2: The system must highlight errors during job execution in the interface.
FR-3: The system should display the percentage of job completion.

UML Diagram


User
pgAdmin
pgAgent
PostgreSQL Database
Job Status Monitor
Job Execution Logs
Real-Time Job Status
Request Job Execution Status
Request Real-Time Updates
Fetch Job Status (FR-1)
Query Job Execution Details
Return Job Execution Status
Send Job Status Information
Log Job Execution (FR-1)
Display Real-Time Updates (FR-1)
alt
[Job Execution Errors Detected (FR-2)]
Highlight Errors
Show Error Message
Show Job Completion Percentage (FR-3)

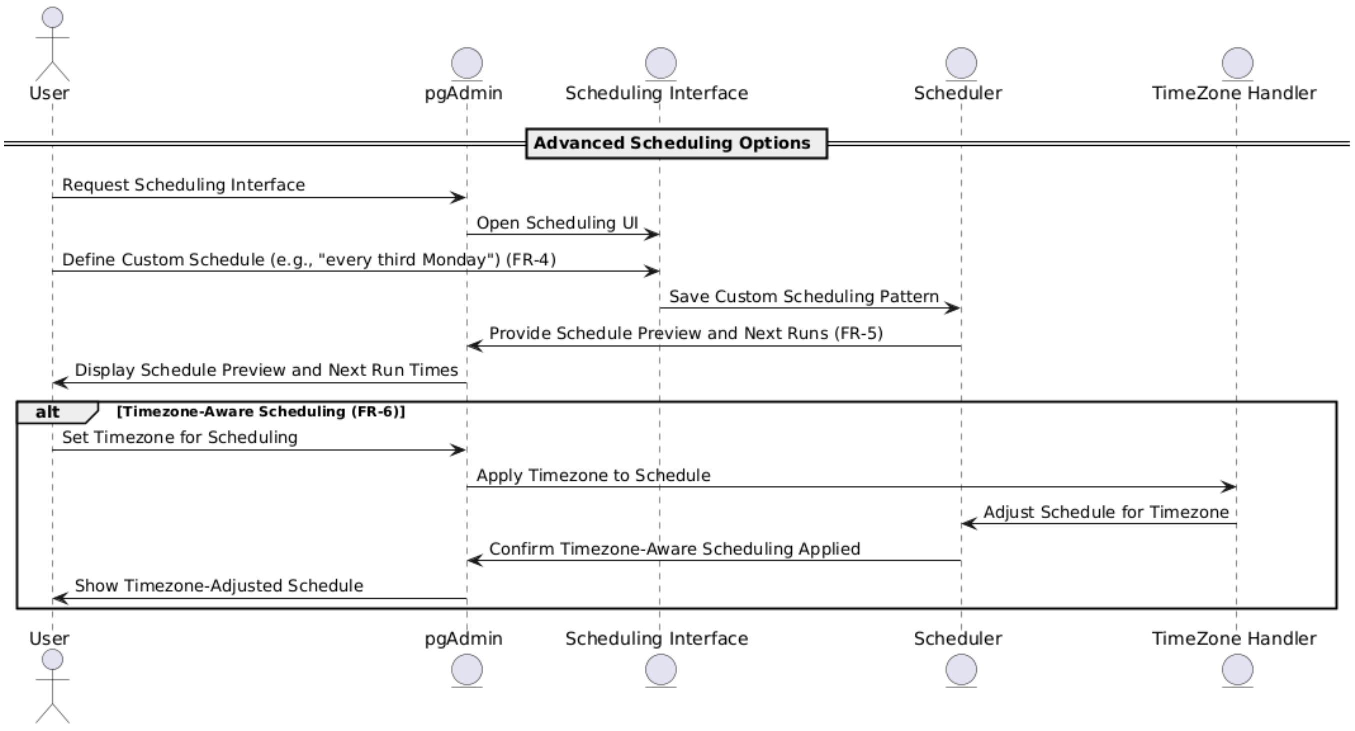
4.2 Advanced Scheduling options

Description
Enables complex job scheduling patterns such as "every third Monday of the month."

Priority
Medium

Functional Requirements
FR-4: The system must support custom scheduling patterns.
FR-5: The system should provide a preview of the next scheduled run times.
FR-6: The system must support timezone-aware scheduling.

UML Diagram


User
pgAdmin
Scheduling Interface
Scheduler
TimeZone Handler
Advanced Scheduling Options
Request Scheduling Interface
Open Scheduling UI
Define Custom Schedule (e.g., "every third Monday") (FR-4)
Save Custom Scheduling Pattern
Provide Schedule Preview and Next Runs (FR-5)
Display Schedule Preview and Next Run Times
alt
[Timezone-Aware Scheduling (FR-6)]
Set Timezone for Scheduling
Apply Timezone to Schedule
Adjust Schedule for Timezone
Confirm Timezone-Aware Scheduling Applied
Show Timezone-Adjusted Schedule

5. Nonfunctional Requirements

5.1 Performance

The system must minimize the impact on database performance during real-time updates. The additional load introduced by the job monitoring feature should not noticeably degrade the database's overall responsiveness.

5.2 Scalability

The system must be able to handle enterprise-level workloads, including a large number of jobs scheduled concurrently. It should be capable of scaling to accommodate growing numbers of job schedules and execution statuses.

5.3 Security

*The system must ensure the **security of communications** by using encrypted channels (AES-256 encryption for data at rest and TLS 1.2+ for data in transit). All sensitive information, including job details, must be kept confidential.*

5.4 Availability

*The system must maintain a minimum **uptime of 99.9%** to ensure continuous operation of job monitoring and scheduling functions, especially for critical tasks in production environments.*

6. System Architecture and Design

6.1 Architectural Overview

The system architecture consists of three primary components:

- ***User Interface (UI)**: The **pgAdmin** dashboard, which provides a visual representation of job statuses and allows the user to configure and view advanced scheduling options.*
- ***Job Execution Engine**: The **pgAgent**, which manages job scheduling and execution in PostgreSQL.*
- ***Monitoring and Alerting**: Integration with external monitoring systems (such as Prometheus and Grafana) for extended visibility and alerting features.*

6.2 System Flow

1. *A user schedules a job via the **pgAdmin** interface, which interacts with the **pgAgent** for execution.*
2. *During job execution, the **pgAdmin** interface displays real-time updates on the job's progress.*
3. *If a job fails or completes, the system will trigger notifications based on predefined alerting mechanisms, such as email or webhook alerts*

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: Real-Time Monitoring Implementation *(Duration: 2 weeks)*

Phase 3: Advanced Scheduling Feature Development *(Duration: 3 weeks)*

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

7.2 Implementation Steps

Step 1: Requirement Analysis – *Analyze the job scheduling and monitoring needs in collaboration with stakeholders.*

Step 2: System Design – *Design the architecture, including user interface and system components.*

Step 3: Development – *Implement real-time job status monitoring and advanced scheduling features.*

Step 4: Integration – *Integrate with monitoring systems like Prometheus and Grafana.*

Step 5: Testing – *Perform functional, performance, and security testing to ensure the system meets all requirements.*

Step 6: Deployment – *Deploy the solution for live testing and usage.*

---

8. Test Plan

8.1 Testing Objectives

- ***Verify real-time job status updates.***
- ***Validate scheduling accuracy.***
- ***Ensure system stability under load.***

8.2 Testing Strategy

- ***Functional Testing****: Ensure that real-time updates and advanced scheduling features function correctly.*
- ***Performance Testing****: Test the system under load to ensure it performs optimally even with a high number of concurrent jobs.*
- ***Security Testing****: Validate encryption, secure communications, and access control mechanisms.*
- ***Usability Testing****: Ensure that the user interface is intuitive and easy to use for scheduling jobs and monitoring statuses.*

8.3 Test Cases

- ***TC-1:*** *Check if a scheduled job appears in the UI.*
- ***TC-2:*** *Verify if job execution status updates in real time.*
- ***TC-3:*** *Validate custom scheduling options.*
- ***TC-4:*** *Test system response when scheduling conflicts occur.*
- ***TC-5:*** *Check time zone compatibility for scheduled jobs.*

9. Conclusion and Future Scope

9.1 Summary

This Software Requirements Specification (SRS) outlines enhancements to pgAdmin and pgAgent, focusing on real-time job status monitoring and advanced scheduling features. The system aims to provide:

- ***Real-Time Job Monitoring**: Enables users to track job execution in real-time within pgAdmin.*
- ***Advanced Scheduling**: Supports complex, recurring job schedules using an intuitive graphical interface.*
- ***Error Detection**: Highlights job failures for quick resolution.*
- ***Performance Optimization**: Designed for minimal impact on database performance while maintaining scalability.*

The solution addresses the need for better job execution monitoring and flexible scheduling, thus improving the operational efficiency of PostgreSQL-based systems. The improvements aim to reduce manual intervention and provide database administrators with better control over the scheduling and monitoring processes.

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. ***AI-Driven Job Execution Predictions**: Future versions could integrate machine learning to predict job success or failure based on historical data. This would allow proactive management of scheduled jobs and reduce downtime by providing early alerts on potential failures.*
2. ***Job Execution History & Visualization**: Enhancing the system to store detailed job execution history and offer interactive visual reports would allow administrators to analyze trends, identify inefficiencies, and optimize scheduling strategies.*
3. ***Cloud Integration**: Future versions could support cloud-based PostgreSQL deployments, enabling real-time monitoring and job scheduling for distributed systems, and leveraging the scalability and flexibility of cloud environments.*
4. ***Automated Job Recovery**: Building on real-time monitoring, future iterations could automatically reschedule or retry failed jobs, based on predefined rules or system load conditions, enhancing system robustness and reducing manual intervention*