Project Kushim: Software Requirements Specification (SRS)

Engineering Architecture Team

January 2026

Contents

1 Introduction

1.1 Purpose

This document provides a detailed description of the software requirements for Project Kushim. It is intended for the development team to ensure all technical constraints and functional behaviors are implemented for a production-ready environment.

2 Overall Description

2.1 Product Perspective

Kushim is a cloud-native platform utilizing a modular architecture. It interacts with external SaaS APIs, cloud storage providers, and user-defined webhooks.

2.2 Design Constraints

- **Language:** TypeScript (Full-stack).
- **Framework:** Next.js (Frontend), NestJS (Backend).
- **Database:** PostgreSQL 15+ for persistence; Redis 7+ for caching.

3 Functional Requirements

3.1 Identity and Access Management (IAM)

1. The system shall implement RBAC (Role-Based Access Control) with "Admin", "Manager", and "Viewer" roles.
2. The system shall support OAuth2 login providers (Google, GitHub, Microsoft).
3. Password storage must utilize Argon2 hashing with a unique salt per user.

3.2 Data Aggregation Service

1. The system shall provide a standardized API adapter interface for third-party integrations.
2. Incoming data must be validated against a JSON Schema before being persisted in the `unified_records` table.
3. Data ingestion must be asynchronous, utilizing a message broker (RabbitMQ/Redis) to prevent request timeouts.

4 External Interface Requirements

4.1 API Specification

- All APIs must adhere to REST principles or GraphQL standards.
- Response format must be `application/json`.
- Error responses must include a standardized error code and correlation ID for log tracing.

5 Non-Functional Requirements

5.1 Performance

- The system shall handle at least 500 requests per second (RPS) per instance.
- Database queries must be optimized with appropriate indexes to ensure 95th percentile latency < 100ms.

5.2 Security

- All data in transit must be encrypted using TLS 1.3.
- JSON Web Tokens (JWT) must be short-lived (15 minutes) with a secure rotation policy for Refresh Tokens.