Software Requirements Specification (SRS)

Multi-Tenant Subscription & Usage Metering Platform

1. Introduction

This Software Requirements Specification (SRS) defines the requirements for a Multi-Tenant Subscription and Usage Metering Platform. The system represents the backend core of a SaaS product and is designed as a realistic portfolio project suitable for a Master's-level software engineering graduate.

1.1 Problem Statement

SaaS platforms must enforce subscription limits fairly and deterministically. Incorrect enforcement leads to abuse, unpredictable costs, and poor user trust. This system addresses these challenges using explicit business rules and concurrency-safe usage tracking without reliance on AI-driven decision making.

1.2 Objectives

1. Design a scalable multi-tenant backend architecture.
2. Implement subscription plans with clear enforcement rules.
3. Track usage accurately under concurrent access.
4. Demonstrate backend engineering maturity.

2. Overall System Description

The system is implemented as a modular monolith backend exposing RESTful APIs. It manages organizations, users, subscription plans, and usage enforcement. Payment processing is explicitly excluded to focus on backend logic.

2.1 User Classes

1. Platform Administrator
2. Organization Administrator
3. Standard User

2.2 Assumptions and Constraints

1. Open-source technologies only.
2. Free-tier deployment targets.
3. No payment gateway integration.

3. Functional Requirements

3.1 Organization Management

1. Create, update, and deactivate organizations.
2. Ensure logical data isolation.

3.2 Subscription Management

1. Define subscription plans with quotas.
2. Assign plans to organizations.

3.3 Usage Metering

1. Track API requests per organization.
2. Evaluate usage within time windows.
3. Persist usage records reliably.

3.4 Enforcement

1. Warn when approaching limits.
2. Restrict access when limits exceeded.

4. Non-Functional Requirements

1. Performance: O(1) enforcement decisions.
2. Security: Strict tenant isolation.
3. Reliability: No data loss on partial failures.
4. Maintainability: Configurable business rules.

5. Core Algorithms and Logic

1. Sliding window counters.
2. Atomic increments.
3. Threshold-based decision trees.

6. Technology Stack

1. Backend: Python or Node.js
2. Database: PostgreSQL
3. Authentication: JWT
4. DevOps: Docker, GitHub Actions

7. Future Enhancements

1. External billing integration.
2. Advanced analytics.
3. Multi-region support.