Software Design Document (SDD)

Requirement: 3.2 - Subscription Management

---

1. Introduction

1.1 Purpose

This document outlines the design and technical implementation of the Subscription Management module in the SaaSManager project. This module facilitates adding SaaS tools from a catalog, tracking subscriptions and licenses, and monitoring usage activity to optimize costs and enhance utilization.

1.2 Scope

The Subscription Management module is a core part of the SaaSManager platform and includes the following functionalities:

- SaaS catalog and subscription tracking.
- License allocation and reassignment.
- Monitoring subscription usage and detecting inefficiencies.

---

2. Architecture and Design Approach

2.1 Overview

The Subscription Management module will use a microservices architecture, separating functionalities into scalable and independent services:

1. **Catalog Service:** Manage predefined SaaS tool data.
2. **Subscription Service:** Track subscription details and assigned users.
3. **Usage Monitoring Service:** Monitor user activity and report on underutilized subscriptions.

2.2 Technologies

- **Backend:** Java with Spring Boot for REST API development.
- **Database:** PostgreSQL for persistent data storage.
- **Frontend:** React for UI, using Material-UI for design consistency.
- **Monitoring Tools:** Integration with tools like Google Workspace or custom APIs for tracking user activity.

- **CI/CD:** Docker for containerization, Jenkins for CI/CD pipelines.

2.3 High-Level System Components

- **Catalog Service:** Stores and provides access to a predefined list of SaaS tools.
- **Subscription Service:** Handles subscription lifecycle, renewal dates, costs, and license allocation.
- **Monitoring Service:** Tracks user activity for underutilized or unused licenses.

---

3. Detailed Design

3.1 Database Design

1. **Catalog Table:** Stores predefined SaaS tools.

```
CREATE TABLE catalog (

  tool_id UUID PRIMARY KEY,

  name VARCHAR(255) NOT NULL,

  description TEXT,

  default_cost DECIMAL(10, 2),

  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP

);
```

2. **Subscriptions Table:** Tracks subscription details.

```
CREATE TABLE subscriptions (

  subscription_id UUID PRIMARY KEY,

  tool_id UUID REFERENCES catalog(tool_id),

  renewal_date DATE,

  cost DECIMAL(10, 2),

  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP

);
```

3. **Subscription Users Table:** Tracks users assigned to subscriptions.

```
CREATE TABLE subscription_users (
  id UUID PRIMARY KEY,
  subscription_id UUID REFERENCES subscriptions(subscription_id),
  user_id INT REFERENCES users(user_id) ON DELETE CASCADE,
  allocated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

4. **Licenses Table:** Tracks license allocation and usage.

```
CREATE TABLE licenses (
  license_id UUID PRIMARY KEY,
  subscription_id UUID REFERENCES subscriptions(subscription_id),
  user_id INT REFERENCES users(user_id) ON DELETE CASCADE,
  allocated_at TIMESTAMP,
  status VARCHAR(50) CHECK (status IN ('allocated', 'unallocated')),
  last_used_at TIMESTAMP
);
```

5. **Usage Logs Table:** Tracks user activity for monitoring purposes.

```
CREATE TABLE usage_logs (
  log_id UUID PRIMARY KEY,
  user_id INT REFERENCES users(user_id) ON DELETE CASCADE,
  tool_id UUID REFERENCES catalog(tool_id),
  activity_date TIMESTAMP,

  activity_type VARCHAR(255)

);
```

3.2 API Design

1. **Catalog Service APIs**
   - **GET /catalog:** Fetch all available SaaS tools.
   - **POST /catalog:** Add a new SaaS tool (Admin only).
2. **Subscription Service APIs**
   - **POST /subscriptions:** Create a new subscription.
     { "tool_id": "tool_id", "renewal_date": "2025-01-01", "cost": 99.99 }
   - **GET /subscriptions:** Get all subscriptions.
   - **GET /subscriptions/{subscription_id}:** Fetch details of a specific subscription.
   - **PUT /subscriptions/{subscription_id}:** Update subscription details.
3. **License Management APIs**
   - **POST /licenses:** Allocate a license to a user.
     { "subscription_id": "subscription_id", "user_id": "user_id" }
   - **DELETE /licenses/{license_id}:** Revoke a license.
   - **GET /licenses/{subscription_id}:** Get all licenses for a subscription.
4. **Usage Monitoring APIs**
   - **GET /usage/logs:** Fetch usage logs for analysis.
   - **GET /usage/underutilized:** Detect underutilized or unused subscriptions.

3.3 Subscription Tracking Workflow

1. Admin selects a SaaS tool from the catalog and creates a subscription via the Subscription Service.
2. Admin assigns licenses to users using the License Management API.
3. Users interact with the SaaS tool. Activity is logged by the Monitoring Service.
4. Admin monitors usage data and reassigns unused licenses as necessary.

3.4 Frontend Design

Key Pages:

1. **Catalog Page:**

    - Displays the predefined SaaS catalog.
    - Admins can add new tools.
2. **Subscription Management Page:**
    - List of subscriptions with renewal dates, costs, and assigned users.
    - Ability to add, update, or delete subscriptions.
3. **License Management Page:**
    - Displays allocated and unallocated licenses for each subscription.
    - Allows license allocation or reassignment.
4. **Usage Monitoring Page:**
    - Visualize usage metrics (e.g., active vs. inactive users).
    - Highlight underutilized subscriptions.

3.5 Monitoring Flow

1. Collect activity logs through integrations or API calls.
2. Analyze data to detect users with low or no activity.
3. Display insights and recommendations in the admin dashboard.

---

4. Security Considerations

1. **Access Control:**
    - Admin-only endpoints for subscription and license management.
    - Role-based access checks using Spring Security.
2. **Data Validation:**
    - Validate all inputs for subscription creation and license allocation.
3. **Audit Logging:**
    - Track changes to subscriptions and licenses for compliance.

---

5. Testing Strategy

1. **Unit Testing:**
    - Test API endpoints (e.g., adding subscriptions, allocating licenses).
2. **Integration Testing:**

    - Ensure seamless interaction between Catalog, Subscription, and Monitoring services.
3. **E2E Testing:**
    - Simulate the admin workflow (adding a tool, creating a subscription, allocating licenses).
4. **Load Testing:**
    - Test system behavior with high volumes of subscriptions and usage logs.

---

6. Deployment Considerations

1. **Database Migrations:**
    - Use Flyway or Liquibase to manage schema updates for new tables.
2. **Service Scaling:**
    - Scale the Monitoring Service independently to handle high activity logs.
3. **CI/CD Pipelines:**
    - Automate builds, tests, and deployments for microservices.

---

7. Future Enhancements

1. **Dynamic SaaS Catalog:**
    - Integrate with APIs to fetch live SaaS catalogs from external sources.
2. **Advanced Analytics:**
    - Add ML-based insights for predicting underutilized subscriptions.
3. **License Optimization Suggestions:**
    - Provide recommendations for optimizing license allocations.

This design document provides a comprehensive approach to implementing the Subscription Management module. Let me know if further refinements or additional details are needed!