# Cloud Service Access Management System

## Description

This project involves developing a backend system for managing access to cloud services based on user subscriptions. The primary users of the system are Customers who access cloud services and Admins who manage subscription plans and permissions. The system will regulate access to various cloud APIs based on the subscription plan purchased by the customer. If a customer exceeds the maximum limit of a service as defined in their plan, access to that specific service will be temporarily restricted.

## Objectives

- To develop a backend system that dynamically manages access to cloud services based on user subscriptions.
- To implement a role-based access control (RBAC) system where the admin can modify user permissions and subscription plans.
- To simulate cloud service usage and enforce limits based on subscription plans.

## Key Features

- Create at least 6 random APIs similar to Cloud Services which will be used as the services that are being managed by this system. It does not matter what these APIs return or these APIs are. They are just here to be managed. Do not focus much on these APIs.

  **Management APIs**

  Subscription Plan Management
    - Admin Functions: **Create, modify, and delete** subscription plans. Each plan will include a set of permissions (API access) and usage limits.
    - Plan Attributes: Plan name, description, list of API permissions, usage limits (e.g., API call limits).

  Permission Management
    - Admin Functions: Add, modify, or delete permissions. Permissions are essentially names of APIs that customers can access.
    - Permission Attributes: Permission name, API endpoint, description.

User Subscription Handling

- Customer Functions: Subscribe to a plan, view current subscription details, and usage statistics.
- Admin Functions: Assign or modify a user's subscription plan.

Access Control

- Implement a mechanism to check if a customer’s API request is within the scope of their subscription plan. Deny access if the user’s plan does not have permission or the request exceeds the plan’s limit.

Usage Tracking and Limit Enforcement

- Track the number of API requests made by each user.
- Temporarily restrict access to an API once a user reaches the maximum limit defined in their subscription.

**Deliverables:**

- GitHub repository containing the source code for the project
- Documentation on how to run and use the API, names of all the team members. (README.txt)
- A brief video on the project, including the design choices and implementation details

**Grading Criteria:**

- Implementation of all required features and functionality
- Proper use of FastAPI
- Proper error handling
- Proper use of asynchronous programming
- Code readability and organization
- Documentation quality

**Good luck with the project!**

Note : You can choose any database that suits the backend design.

## Example - Sample API list

It is not required to follow this API list. This is an example.

Subscription Plan Management

- POST /plans (Create Plan)
- PUT /plans/{planId} (Modify Plan)
- DELETE /plans/{planId} (Delete Plan)

Permission Management

- POST /permissions (Add Permission)
- PUT /permissions/{permissionId} (Modify Permission)
- DELETE /permissions/{permissionId} (Delete Permission)

User Subscription Handling

- POST /subscriptions (Subscribe to Plan)
- GET /subscriptions/{userId} (View Subscription Details)
- GET /subscriptions/{userId}/usage (View Usage Statistics)
- PUT /subscriptions/{userId} (Assign/Modify User Plan)

Access Control

- GET /access/{userId}/{apiRequest} (Check Access Permission)

Usage Tracking and Limit Enforcement

- POST /usage/{userId} (Track API Request)
- GET /usage/{userId}/limit (Check Limit Status)