Project Title: Multi-Tenant SaaS Billing & Usage Analytics Platform

**1.Project Description**

A scalable **SaaS platform** for subscription billing and usage analytics. The system supports **multi-tenant management**, including **plan upgrades,** **downgrades,** and **reactivation,** while enforcing **usage limits** across free and premium plans.

The backend is built with **FastAPI (async)** and **MongoDB** for efficient, secure data storage. **JWT authentication** ensures secure admin access, while tenants seamlessly manage their subscriptions and usage. **Docker containerization** makes deployment simple, portable, and scalable.

**2. Project Overview**

This project is a **scalable SaaS billing and analytics platform** designed to handle multiple tenants, manage subscriptions, and track usage in real-time.  
It leverages **FastAPI (async)**, **MongoDB**, and **Docker** for efficient deployment and performance.

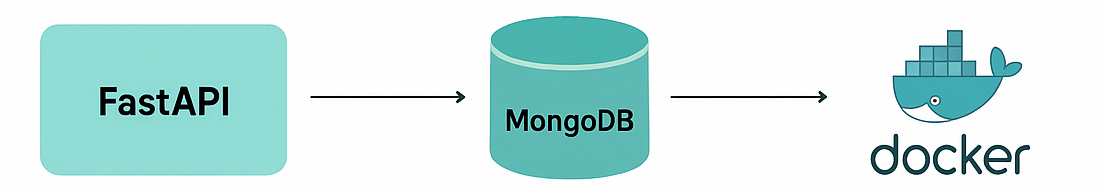
**Key Highlights:**

* Multi-tenant architecture for subscription management
* Plan upgrades, downgrades, and reactivation
* Usage-based billing with quotas and pricing
* JWT authentication for admin security
* Fully asynchronous backend with FastAPI

**3. Project Objectives**

* Build a **scalable SaaS platform** for managing tenants and subscriptions.
* Support **plan upgrades/downgrades** with automated billing.
* Track and enforce **usage limits** for free and premium tenants.
* Ensure **secure authentication & authorization** using JWT.
* Deploy application in **containerized environment** using Docker.

**4. System Architecture**



**Explanation:**

* FastAPI handles tenant & admin requests asynchronously.
* MongoDB stores tenant, subscription, and usage data.
* JWT ensures secure authentication for admin operations.
* Docker enables consistent deployment across environments.

**5. Tech Stack & Key Features**

**Technology Stack:**

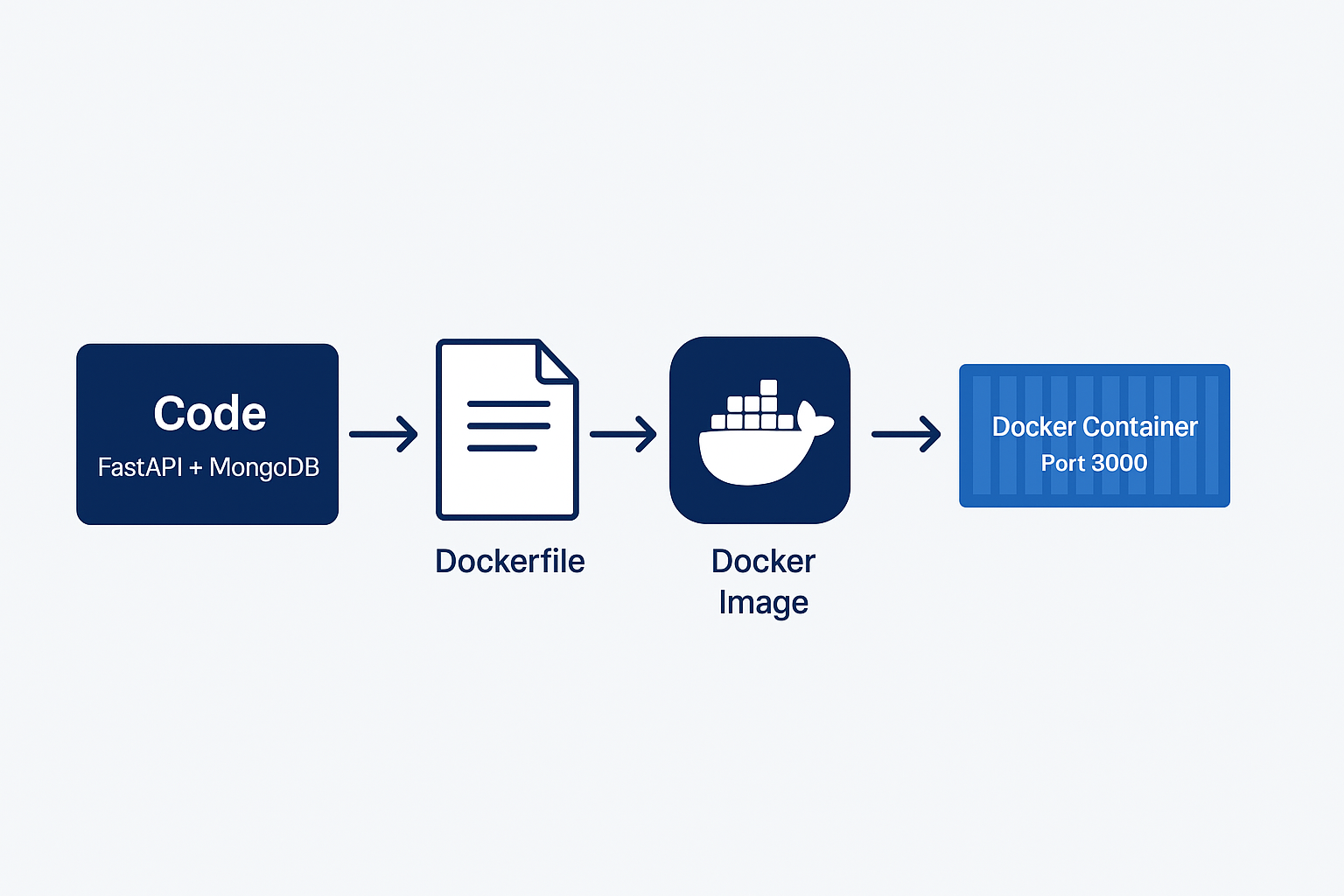
* **Backend:** FastAPI (async REST framework)
* **Database:** MongoDB (Motor for async queries)
* **Authentication:** JWT (for admins)
* **Deployment:** Docker (containerized environment)
* **Language:** Python 3.x

**Key Features:**

* Tenant registration & management
* Subscription upgrades/downgrades
* Usage tracking and enforcement
* Automated billing & invoice generation
* Admin dashboard APIs with JWT security

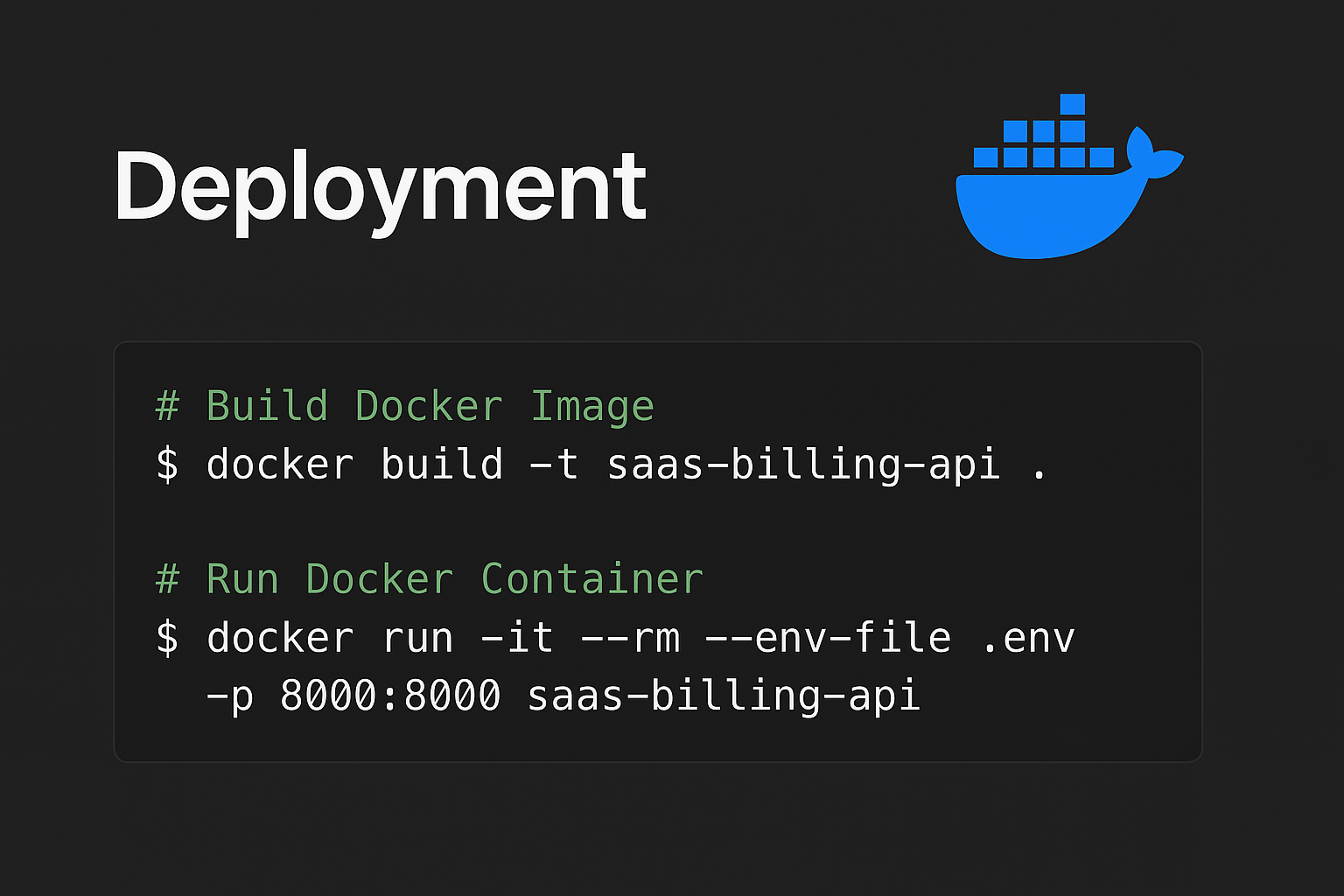
**6. Deployment – Dockerized FastAPI**

**Workflow:**

Code → Dockerfile → Docker Image → Docker Container (Port 8000)

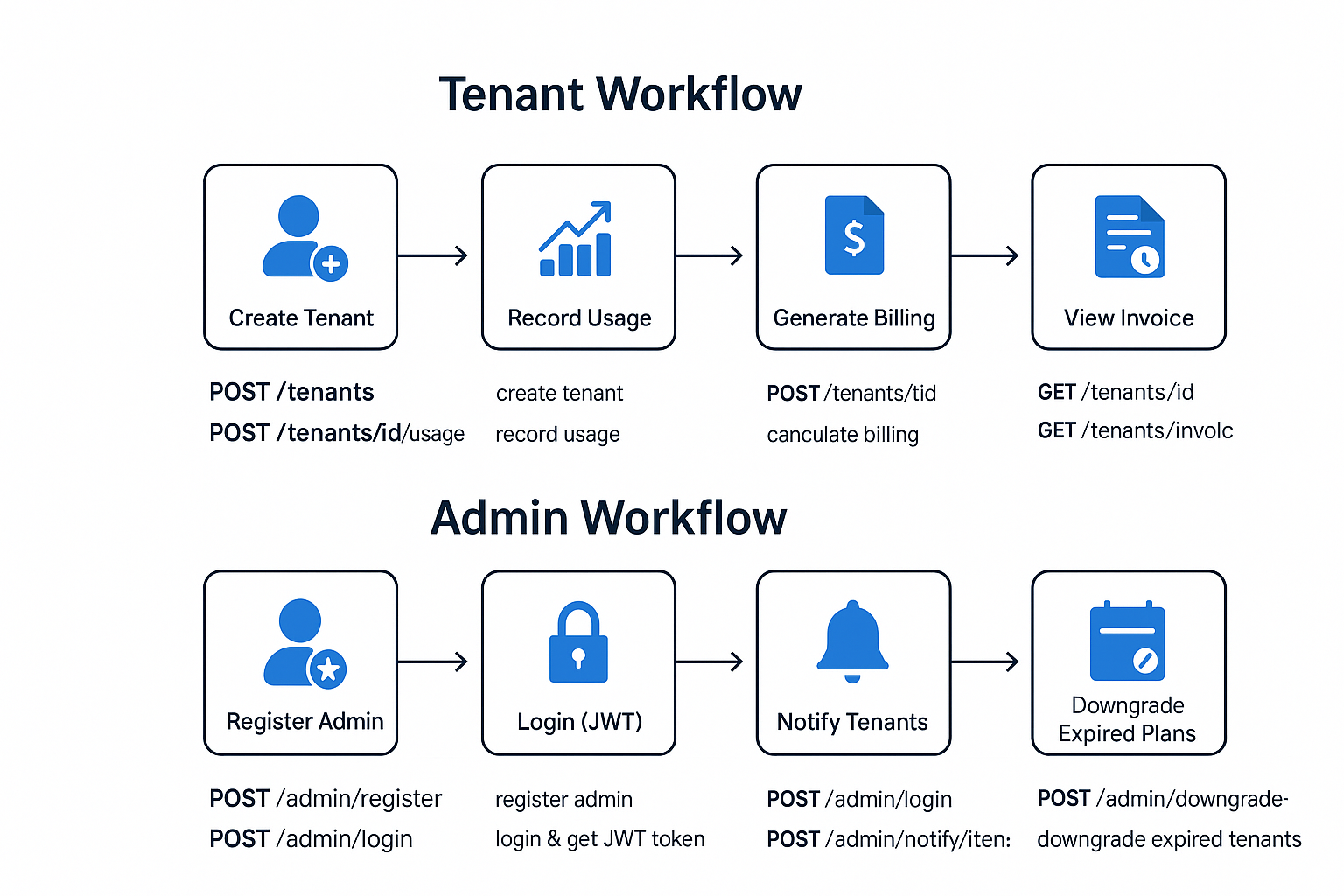
**Deployment Commands:**

* Ensures **portability & consistency** across environments
* MongoDB URI provided via **environment variable** (MONGO\_URI)
* Easy to run anywhere using:



**7. System Workflows**

**Tenant Workflow:**  
Register Tenant → Record Usage → Track Quota → Generate Invoice

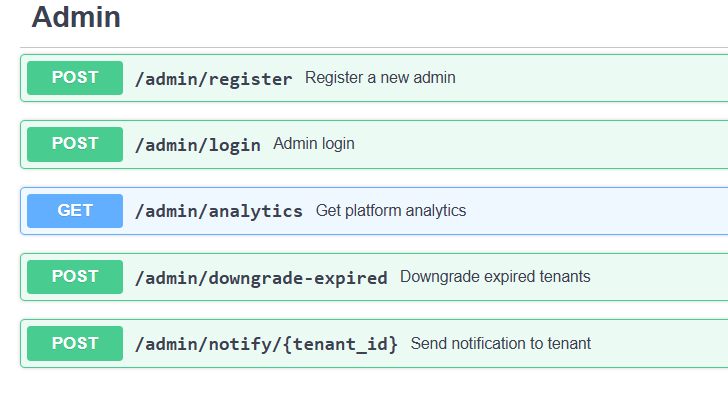


**Admin Workflow:**  
Register/Login → Issue JWT → Notify Tenant → Downgrade Expired Tenant

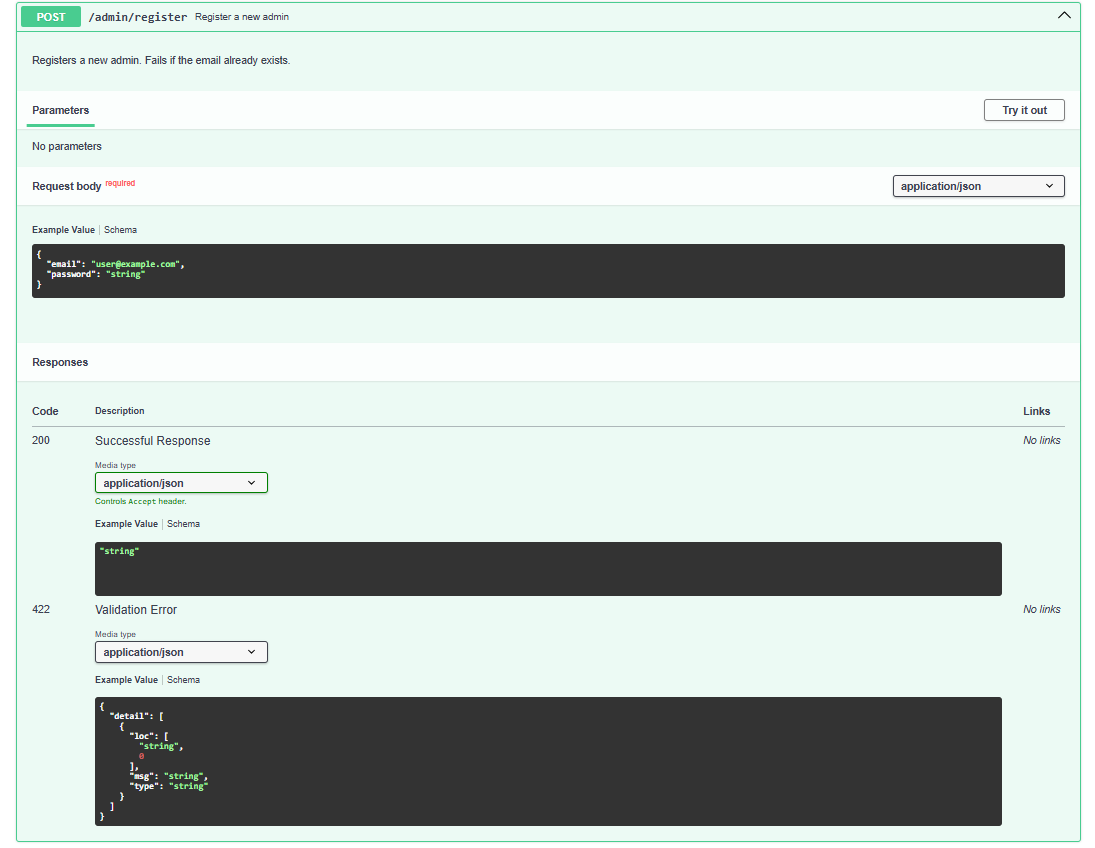
**8. Swagger UI – API Demonstration**

**Admin Endpoints (JWT Protected):**

* POST /admin/register → Register new admin
* POST /admin/login → Login & get JWT
* GET /admin/analytics → Get platform analytics
* POST /admin/notify/{tenant\_id} → Send notification
* POST /admin/downgrade-expired → Auto-downgrade tenants



**Sample Request/Response:**



**Swagger UI - Tenant Endpoints (No JWT):**

**Tenant Management**

* **POST** / → Add new tenant
* **GET** / → List all tenants
* **GET** /search → Search tenant by name or ID
* **GET** /{tenant\_id} → Fetch tenant details
* **DELETE** /{tenant\_id} → Deactivate tenant
* **POST** /{tenant\_id}/reactivate → Reactivate tenant

**Plan & Subscription**

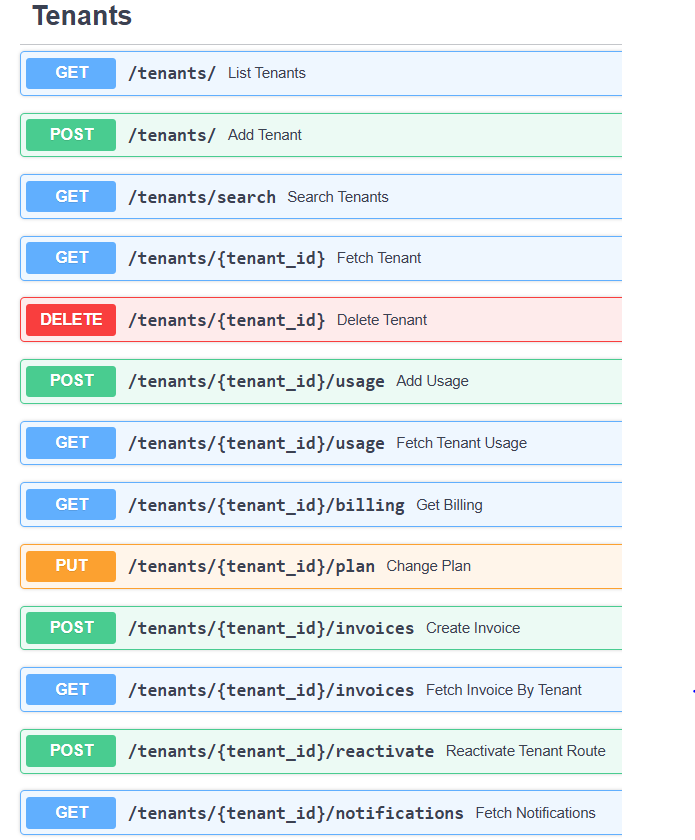
* **PUT** /{tenant\_id}/plan → Change tenant’s plan

**Usage & Billing**

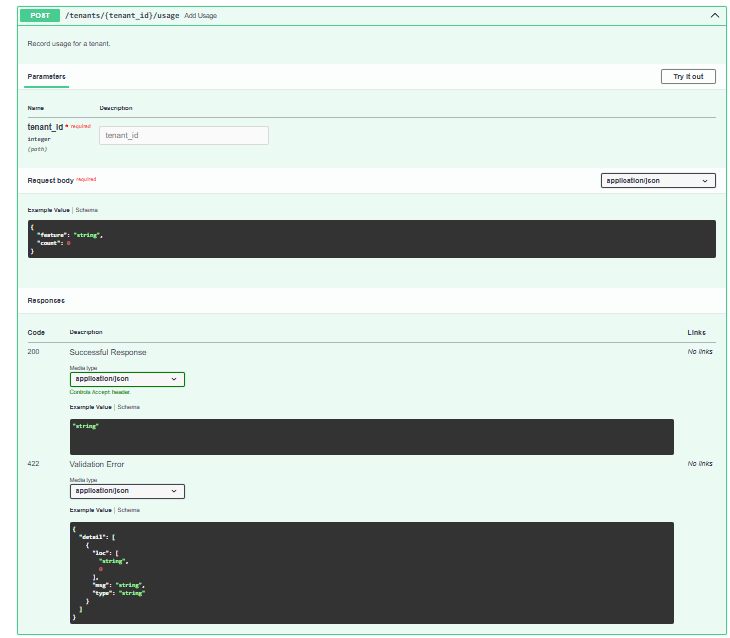
* **POST** /{tenant\_id}/usage → Record usage
* **GET** /{tenant\_id}/usage → Fetch usage summary
* **GET** /{tenant\_id}/billing → Billing invoice
* **POST** /{tenant\_id}/invoices → Create new invoice
* **GET** /{tenant\_id}/invoices → Fetch latest invoice

**Notifications**

* **GET** /{tenant\_id}/notifications → Fetch notifications



**Sample Request/Response:**



**9. Challenges & Solutions**

| **Challenge** | **Solution** |
| --- | --- |
| Multi-tenant data handling | Designed **tenant\_id-based segregation** in MongoDB |
| Async database operations | Used **Motor** for async MongoDB queries |
| Secure authentication | Implemented **JWT-based admin auth** |
| Deployment issues | Containerized with **Docker** for consistency |
| Usage-based billing | Implemented **quotas + pricing model** |

**10. Key Learnings**

* Built a **fully asynchronous backend** with FastAPI & MongoDB
* Implemented **JWT-based authentication** for secure endpoints
* Gained hands-on experience with **Docker containerization**
* Learned **decorators & custom logging** for execution time tracking
* Designed **scalable SaaS architecture** with real-world features

**11. Conclusion**

This project demonstrates a **production-grade SaaS billing system** with:

* Multi-tenant subscription management
* Usage tracking & billing automation
* Secure deployment using Docker