Billing Coding Exercise

Overview

The goal of this exercise is to have you replicate how we integrate **Metronome** (our billing engine) on the Hugging Face Hub. Metronome handles usage-based billing, subscriptions, credits, and invoice generation, while **Stripe** serves as our payment provider.

Billing System Domain

Overview

Our billing system uses a **dual-provider model**:

1. **Metronome** (Billing Engine)

    - Manages subscriptions and billing
    - Applies credits and generates invoices
    - Handles tier transitions

2. **Stripe** (Payment Provider)

    - Stores customer payment methods
    - Processes payments
    - Handles payment failures and retries

Domain Concepts

Billing Entities

We bill two types of entities:

- **Users**: Individual users
- **Organizations**: Teams and organizations

Billing Tiers

We support multiple billing tiers (free, pro, team, enterprise, etc.). Each tier includes:

- A subscription product (recurring monthly/yearly)
- Monthly credits for usage-based products - see:
    - https://huggingface.co/pricing
    - https://huggingface.co/docs/hub/storage-limits

- https://huggingface.co/docs/inference-providers/pricing

Exercise: Implement Metronome Integration

Setup

Mock Metronome Server (Optional): Since Metronome (contrary to Stripe) doesn't allow developers to create accounts directly, we provide a mock Metronome server that emulates Metronome. You can use it to test your implementation, or you can mock/fake Metronome in any other way you prefer.

Your Task

Implement a Metronome integration that manages the full customer lifecycle of a simplified billing system. This includes:

- **Customer onboarding**: Setting up new customers
- **Tier management**: Managing subscription/billing lifecycle and billing tiers upgrades/downgrades

Design your own solution architecture and implementation approach. Document any tradeoff you decide on.

Deliverables

1. **Code**: Your implementation (TypeScript/JavaScript preferred)
2. **Documentation**: Brief explanation of your approach and design decisions
3. **Testing**: Test cases or instructions on how to test your implementation