Assignment: Crypto Data & Organization Management System

Objective
Develop a Django-based system that:

- Manages organizations (org_id)
- Fetches cryptocurrency prices from a public API
- Associates' prices with an organization (org_id)
- Provides scheduled updates using Celery
- Implements full CRUD APIs for organizations and crypto data

Requirements
1. Models
🔹 ***Organization Model***
Create a model Organization with:

- id (UUID, Primary Key)
- name (CharField, max_length=255, unique)
- created_at (DateTimeField, auto_now_add=True)

🔹 ***CryptoPrice Model***
Create a model CryptoPrice with:

- id (Primary Key, AutoField)
- org_id (ForeignKey to Organization)
- symbol (CharField, max_length=10, e.g., "BTC", "ETH")
- price (DecimalField, max_digits=20, decimal_places=10)
- timestamp (DateTimeField, auto_now_add=True)

2. Third-Party API
Use the **CoinGecko API** to fetch real-time crypto prices.
Example API:
https://api.coingecko.com/api/v3/simple/price?ids=bitcoin,ethereum&vs_currencies=usd

3. Scheduler (Celery & Redis)
Set up a Celery task that runs **every 5 minutes** to:

- Fetch the latest **Bitcoin (BTC)** and **Ethereum (ETH)** prices.
- Store or update the data in CryptoPrice for **each** org_id.

Use **Redis** as the Celery message broker.
4. API"s

🔹 Organization APIs
Create the organization api's (CRUD) separately.
🔹 Crypto Price APIs
Create the crypto data api's separately.

5. Authentication & Security

- **JWT Authentication** (Django Rest Framework JWT or Simple JWT)

- Users can **only** access org_id-specific data
- API endpoints should have **permissions** (only allow org owners to edit/delete)

6. Additional Challenges (For Extra Marks)

Implement **pagination** in GET /api/crypto-prices/ API
Store **historical price data** instead of overwriting previous prices
Implement **Django signals** to log when an org is created or deleted
Add **search** and **filtering** to the Organization API
Write **unit tests** for all APIs