■ Product Search System (Text + Image) with ChromaDB & FastAPI

1. ■ Project Overview

```
This project allows searching products using:
- Text search (SentenceTransformer all-MiniLM-L6-v2)
- Image search (CLIP model clip-ViT-B-32)
- Products embedded & stored in ChromaDB
- Backend APIs with FastAPI for Insert, Update, Delete, Search
```

2. ■ Project Structure

```
AI_PRODUCT_SEARCH_MAIN/
■■■ backend.py             # FastAPI backend (CRUD + Search APIs)
■■■ embed_to_chroma.py     # Embedding script
■■■ chroma_db/             # ChromaDB storage
■■■ requirements.txt       # Dependencies
```

3. ■■ Setup Instructions

```
1. Install Python 3.9+
2. Create virtual environment:
   python -m venv venv
   source venv/bin/activate (Linux/Mac)
   venv\Scripts\activate (Windows)

3. Install dependencies:
   pip install -r requirements.txt
```

requirements.txt

```
fastapi
uvicorn
chromadb
psycopg2
sentence-transformers
pillow
requests
```

4. ■■ Data Embedding

```
Run: python embed_to_chroma.py
- Fetch products from PostgreSQL
- Embed text & images
- Store embeddings in ChromaDB collections
```

5. ■ Backend API (FastAPI)

```
Run server:
   uvicorn backend:app --reload

Docs available at:
   http://127.0.0.1:8000/docs
   http://127.0.0.1:8000/redoc
```

6. ■ Available APIs

```
Root: GET /
Fetch products: GET /products?offset=0&limit=50
Text search: GET /search?q=sample&top_k=25
Image search: POST /image-search (upload file)
Insert: POST /insert
Update: POST /update
Delete: POST /delete
```

Insert Payload Example

```
{ "id": "12345", "oem_id": "OEM-111", "name": "Test Product", "description": "This is a test insert",
```

Update Payload Example

```
{ "id": "12345", "name": "Updated Product", "description": "Updated desc", "images": "https://example
```

Delete Payload Example

```
{ "id": "12345" }
```

7. ■ Testing APIs

```
- Use Postman or curl
- Insert/Update: Body → raw → JSON
- Delete: Body → raw → JSON
- Image Search: Body → form-data → file
```

8. ■ Project Flow

```
1. PostgreSQL → embed_to_chroma.py → ChromaDB
2. backend.py → FastAPI → CRUD + search
3. Frontend (optional) → API calls → UI results
```