**Milestone 6: Pinecone Semantic Search Integration**

**🎯 Objective:**

Use the sentence-transformers model to **embed policy documents** and **store them in Pinecone** for later semantic search.

**📁 Directory Structure:**

arduino

CopyEdit

services/

├── document\_embedder.py ← Embeds & stores documents

├── document\_retriever.py ← Searches Pinecone

├── granite\_llm.py

core/

├── config.py

**✅ Step 1: Load Pinecone & Sentence-Transformers**

**📄 document\_embedder.py**

python

CopyEdit

import os

import pinecone

from sentence\_transformers import SentenceTransformer

from core.config import settings

# Init Pinecone

pinecone.init(api\_key=settings.pinecone\_api\_key, environment=settings.pinecone\_env)

index\_name = settings.index\_name

if index\_name not in pinecone.list\_indexes():

pinecone.create\_index(index\_name, dimension=384)

index = pinecone.Index(index\_name)

# Load embedding model

model = SentenceTransformer('all-MiniLM-L6-v2')

def embed\_and\_store\_document(doc\_text: str, doc\_id: str):

# Split doc into chunks if needed

chunks = [doc\_text[i:i+500] for i in range(0, len(doc\_text), 500)]

vectors = []

for i, chunk in enumerate(chunks):

embedding = model.encode(chunk).tolist()

vectors.append({

"id": f"{doc\_id}\_{i}",

"values": embedding,

"metadata": {"text": chunk}

})

# Upsert to Pinecone

index.upsert(vectors=vectors)

return f"Document {doc\_id} embedded and stored successfully."

**✅ Step 2: Add .env Configurations**

**📄 .env**

ini

CopyEdit

PINECONE\_API\_KEY=your\_pinecone\_key

PINECONE\_ENV=your\_pinecone\_env

INDEX\_NAME=smartcity-policies

**📄 core/config.py**

python

CopyEdit

from pydantic\_settings import BaseSettings

class Settings(BaseSettings):

pinecone\_api\_key: str

pinecone\_env: str

index\_name: str

class Config:

env\_file = ".env"

settings = Settings()

**✅ Step 3: Upload and Embed Route (FastAPI)**

**📄 routers/vector\_router.py**

python

CopyEdit

from fastapi import APIRouter

from services.document\_embedder import embed\_and\_store\_document

router = APIRouter()

@router.post("/upload-doc")

def upload\_and\_embed(text: str, doc\_id: str = "doc"):

message = embed\_and\_store\_document(text, doc\_id)

return {"message": message}