

Environment Initialization & API Key Setup

Objective: Configure and secure external service credentials (Watsonx & Pinecone).

Generate API Keys

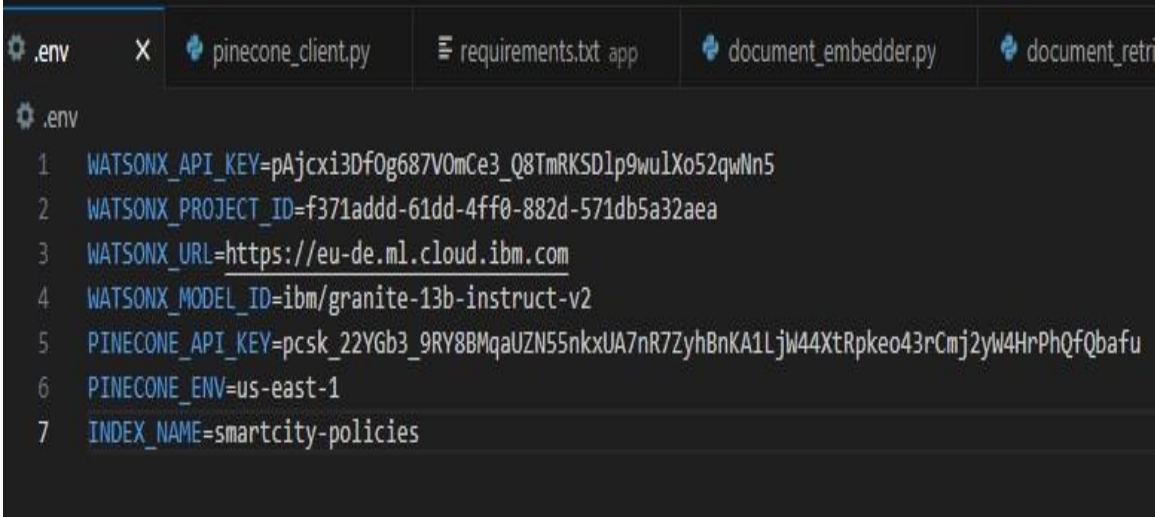
Watsonx Granite credentials from IBM Cloud dashboard

Pinecone API key and environment from

<https://app.pinecone.io>

Define .env File

Create a .env file to hold: **WATSONX_API_KEY=your_ibm_api_key**
WATSONX_PROJECT_ID=your_project_id
WATSONX_URL=https://your-region.ml.cloud.ibm.com
WATSONX_MODEL_ID=ibm/granite-13b-instruct-v2
PINECONE_API_KEY=your_pinecone_key
PINECONE_ENV=your_pinecone_env **INDEX_NAME=smartcity-policies**



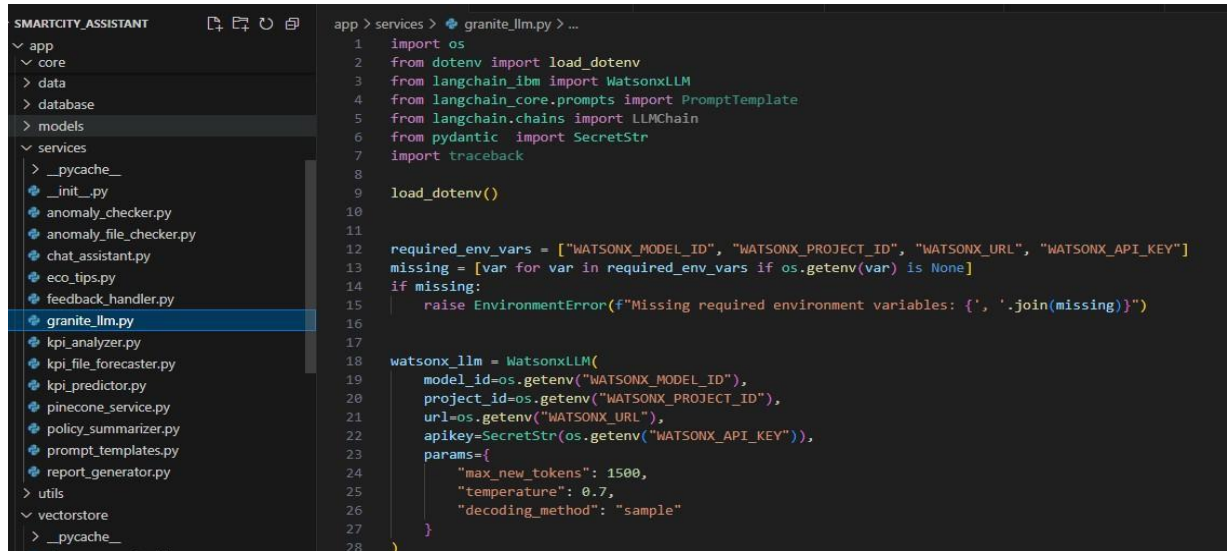
```
.env
1 WATSONX_API_KEY=pAjcxi3Df0g687V0mCe3_Q8TmRKSD1p9wu1Xo52qwNn5
2 WATSONX_PROJECT_ID=f371addd-61dd-4ff0-882d-571db5a32aea
3 WATSONX_URL=https://eu-de.ml.cloud.ibm.com
4 WATSONX_MODEL_ID=ibm/granite-13b-instruct-v2
5 PINECONE_API_KEY=pcsk_22YGb3_9RY8BMqaUZN55nKxUA7nR7ZyhBnKA1LjW44XtRpkeo43rCmj2yW4HrPhQfQbafu
6 PINECONE_ENV=us-east-1
7 INDEX_NAME=smartcity-policies
```

Watsonx Integration

Load env variables using python-dotenv

Set up granite_llm.py to handle summarization, chat, eco tips, and sustainability reports

Test LLM endpoints using dummy prompts



```
SMARTCITY_ASSISTANT
├── app
│   ├── core
│   ├── data
│   ├── database
│   ├── models
│   └── services
│       ├── __pycache__
│       ├── __init__.py
│       ├── anomaly_checker.py
│       ├── anomaly_file_checker.py
│       ├── chat_assistant.py
│       ├── eco_tips.py
│       ├── feedback_handler.py
│       └── granite_llm.py
│           ├── kpi_analyzer.py
│           ├── kpi_file_forecaster.py
│           ├── kpi_predictor.py
│           ├── pinecone_service.py
│           ├── policy_summarizer.py
│           ├── prompt_templates.py
│           └── report_generator.py
├── utils
├── vectorstore
├── __pycache__
└── ...

app > services > granite_llm.py > ...
1  import os
2  from dotenv import load_dotenv
3  from langchain_ibm import WatsonxLLM
4  from langchain_core.prompts import PromptTemplate
5  from langchain.chains import LLMChain
6  from pydantic import SecretStr
7  import traceback
8
9  load_dotenv()
10
11
12  required_env_vars = ["WATSONX_MODEL_ID", "WATSONX_PROJECT_ID", "WATSONX_URL", "WATSONX_API_KEY"]
13  missing = [var for var in required_env_vars if os.getenv(var) is None]
14  if missing:
15      raise EnvironmentError(f"Missing required environment variables: {' '.join(missing)}")
16
17
18  watsonx_llm = WatsonxLLM(
19      model_id=os.getenv("WATSONX_MODEL_ID"),
20      project_id=os.getenv("WATSONX_PROJECT_ID"),
21      url=os.getenv("WATSONX_URL"),
22      apikey=SecretStr(os.getenv("WATSONX_API_KEY")),
23      params={
24          "max_new_tokens": 1500,
25          "temperature": 0.7,
26          "decoding_method": "sample"
27      }
28  )
```

Implement LLM Service Functions

- > ask_granite(prompt) for chat
- > generate_summary(text) for policy summarization
- > generate_eco_tip(topic) for environmental suggestions
- > generate_city_report(kpi_data) for sustainability reports