Technical Assignment: AI-Powered Text Intelligence API

Objective

Build an end-to-end NLP-based intelligent API service using Python, FastAPI, and an ML/LLM model. The project should showcase expertise in NLP, FastAPI backend design, model integration, embeddings, and deployment.

Task Overview

1. Text Sentiment & Keyword Analysis:
Endpoint /analyze: Accepts text and returns sentiment (positive/negative/neutral) and top 5 keywords.

2. Text Summarization or Generation:
Endpoint /summarize: Uses a pre-trained Transformer (T5, BART, GPT) to summarize text.

3. Embeddings & Vector Search (Optional):
Endpoint /semantic-search: Stores embeddings using FAISS, Pinecone, or Chroma and returns similar texts.

4. Architecture & Deployment:
Containerize the project using Docker and include setup instructions.
Ensure Swagger UI documentation is available at /docs.

Technical Requirements

Language: Python 3.8+
Framework: FastAPI (mandatory), Pydantic
NLP/ML Tools: Hugging Face Transformers, SpaCy, Scikit-learn
LLMs: OpenAI API, LangChain (for RAG or prompt chaining)
Databases: FAISS, Pinecone, or Chroma
Deployment: Docker (mandatory)
Bonus: MLOps pipeline, model versioning, or CI/CD

Deliverables

Source code (GitHub link or zip file)

requirements.txt

Dockerfile (ready to run)

README.md with setup instructions and sample requests

(Optional) Live demo URL if deployed on cloud

Evaluation Criteria

Code Quality & Documentation

Correctness of API Functionality

Proper Integration of ML/NLP Models

Performance Optimization

Deployment Readiness (Docker working)

Bonus: Use of LangChain, RAG, or LLM fine-tuning

Creativity in Implementation

Duration

Intermediate Level: 6-8 hours

Senior Level: 10-12 hours

Example API Usage

POST /analyze

Input:

```
{
  "text": "I love working with AI! It makes everything efficient."
}
```

Response:

```
{
  "sentiment": "positive",
  "keywords": ["AI", "efficient", "love"]
}
```