

RAG Engine Instructions & Testing Guide

1. Procedure to Run the Backend

Step 1 — Install Dependencies

Run: `pip install -r requirements.txt`

Step 2 — Ensure .env is Configured

Your `.env` must include:

```
GOOGLE_API_KEY=...
PINECONE_API_KEY=...
PINECONE_INDEX_NAME=legal-docs
PINECONE_ENV=us-east-1
```

Step 3 — Start the FastAPI Server

Run: `uvicorn main:app --reload --host 0.0.0.0 --port 8000`

You should see: `Uvicorn running on http://127.0.0.1:8000`

2. Testing Endpoints Step-by-Step

A. Test Server Status (GET /)

Open: `http://localhost:8000/`

Expected Output: `{"message": "Backend running"}`

B. Test Document Processing (POST /process)

Go to: <http://localhost:8000/docs>

Use POST /process: Upload a .txt or .pdf file.

Expected Output:

```
{
  "content_groups": ["chunk1...", "chunk2..."],
  "images": []
}
```

C. Test Ingestion to Pinecone (POST /ingest)

Upload the same file to: POST /ingest

Expected:

```
{  
  "doc_id": "generated-uuid",  
  "stored_chunks": N  
}
```

This confirms:

- Chunking works
 - Gemini embeddings work
 - Pinecone upsert works
-

D. Test Semantic Search (POST /query)

Send:

```
{  
  "query": "What is this document about?"  
}
```

Expected:

```
{  
  "top_chunks": [  
    {"text": "...", "score": 0.89},  
    {"text": "...", "score": 0.84}  
  ]  
}
```

This confirms RAG retrieval works.

3. Next Steps

1. Implement `llm_engine.py` (LLM reasoning layer) ←— chatbot
 - Summaries
 - Simplified explanations

- Legal risk detection
 - Q&A;
2. Add `/ask` endpoint integrating:
 - RAG retrieval
 - Gemini reasoning model
 3. Add chat memory for multi-turn conversation.
 4. Add frontend:
 - File uploader
 - Chat interface
 - Document viewer
 - Risk panel
 - Voice support
 5. Add TTS (text-to-speech) & multilingual support.