MilitaryDocs-RAG: Document Chatbot with Vector Search

Overview:

MilitaryDocs-RAG is a Retrieval-Augmented Generation (RAG) application designed to interact with over 2500 military PDF documents. It combines vector databases and language models to answer document-specific queries accurately and efficiently.

Features:

- Semantic search across 2500+ PDFs using vector embeddings
- RAG-based chatbot powered by OpenAI (GPT-4)
- Metadata-aware document querying (sections, scopes, etc.)
- FastAPI backend with modular API
- Scalable ingestion of documents with chunking
- Secure configuration via .env file

Project Structure:

```
miltarydocs_rag/
??? backend/
?   ??? api/              # FastAPI endpoints
?   ??? services/         # Embedding and QA logic
?   ??? utils/            # Helper functions
?   ??? vector_store/     # Pinecone client setup
?   ??? main.py           # FastAPI application
??? scripts/
?   ??? ingest_documents.py  # PDF processing and ingestion
??? data/
?   ??? pdfs/             # Raw PDF files
```

```
??? .env                # API credentials (ignored in git)
??? requirements.txt       # Python dependencies
??? README.md              # Project info
```

Setup Instructions:

1. Clone the repository:

```
git clone https://github.com/yourusername/militarydocs_rag.git
```

2. Set up the virtual environment:

```
python3 -m venv venv
source venv/bin/activate
```

3. Install dependencies:

```
pip install -r requirements.txt
```

4. Configure environment variables in .env:

```
OPENAI_API_KEY=your_openai_key
PINECONE_API_KEY=your_pinecone_key
PINECONE_ENV=your_pinecone_env
PINECONE_INDEX_NAME=your_index_name
```

5. Ingest documents:

   Place PDFs in data/pdfs/ and run:

```
python scripts/ingest_documents.py
```

6. Start the backend:

```
uvicorn backend.main:app --reload
```

Example Questions:

- What is the scope of this document
- Tell me about section 1.2.2
- Write an inspection plan for requirement 2.3.3

Future Enhancements:

- Document chunking optimization
- Question type classifier (general vs specific)
- Feedback loop integration
- Frontend chat interface
- Local LLM fallback
- Caching and load optimization
- Security and compliance validation

Security Best Practices:

- API key management via .env
- Encrypt data at rest and in transit
- Role-based access controls
- Limit access to sensitive files

Contributors:

Yashwanth Sai Tirukkovalluru

You (Add more names if collaborating)

License: