🤖 RAG Flask Application

A production-ready **Retrieval-Augmented Generation (RAG)** system built with Flask, LangChain, ChromaDB, and Groq. Supports multiple file formats including documents and audio files with automatic transcription.

✨ Features

- 📄 **Multi-format Support**: PDF, TXT, DOCX, PPTX, CSV
- 🎤 **Audio Transcription**: MP3, WAV files (using Faster-Whisper)
- 💾 **Persistent Storage**: ChromaDB vector database survives restarts
- 🔍 **Semantic Search**: Advanced retrieval with sentence transformers
- 🎯 **Semantic Chunking**: Intelligent document splitting
- 🚀 **Production-Ready**: Proper error handling, logging, and validation
- 💅 **Modern UI**: Responsive design with loading states and feedback
- 🔄 **Real-time Processing**: No page reloads, smooth UX

📋 Prerequisites

- Python 3.8 or higher
- Groq API key (free tier available)

🚀 Quick Start

1. Clone and Setup

bash

```
# Create project directory
mkdir rag-flask-app
cd rag-flask-app

# Create virtual environment
python -m venv venv

# Activate virtual environment
# On Windows:
venv\Scripts\activate
# On macOS/Linux:
source venv/bin/activate
```

2. Install Dependencies

bash

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

3. Configure Environment

bash

```
# Copy environment template
cp .env.example .env

# Edit .env and add your Groq API key
# Get your key from: https://console.groq.com/keys
```

Your `.env` file should look like:

env

```
GROQ_API_KEY=gsk_your_actual_api_key_here
```

4. Create Project Structure

bash

```
mkdir -p templates static utils uploads chroma_db
```

Create the following files:

- `app.py` (main Flask application)
- `utils/transcription.py` (audio transcription)
- `utils/loaders.py` (document loaders)
- `templates/index.html` (query interface)
- `templates/upload.html` (upload interface)
- `static/style.css` (styling)

5. Run the Application

bash

```
python app.py
```

The application will start on `http://localhost:5000`

📖 Usage Guide

Uploading Documents

1. Navigate to `http://localhost:5000/upload`
2. Click "Choose Files" and select one or multiple files
3. Supported formats:
   - Documents: PDF, TXT, DOCX, PPTX
   - Data: CSV
   - Audio: MP3, WAV (will be transcribed automatically)
4. Click "Upload & Index"
5. Wait for processing to complete

Asking Questions

1. Go to `http://localhost:5000` (home page)
2. Type your question in the text area
3. Click "Ask Question"
4. View the AI-generated answer based on your documents
5. Use the "Copy" button to copy the answer

🔑 Getting a Groq API Key

1. Visit Groq Console
2. Sign up for a free account
3. Navigate to "API Keys" section
4. Click "Create API Key"
5. Copy the key and add it to your `.env` file

Note: Groq offers free tier access with generous limits, perfect for development and testing.

📐 Project Structure

```
rag-flask-app/
├── app.py                  # Main Flask application
├── requirements.txt        # Python dependencies
├── .env                    # Environment variables (create this)
├── .env.example            # Environment template
├── .gitignore              # Git ignore rules
├── README.md               # This file
├── chroma_db/              # Persistent vector database (auto-created)
├── uploads/                # Temporary file storage (auto-created)
├── templates/
│   ├── index.html          # Query interface
│   └── upload.html         # Upload interface
├── static/
│   └── style.css           # CSS styling
└── utils/
    ├── transcription.py    # Audio transcription module
    └── loaders.py          # Document loader utilities
```

🔧 Technical Details

Components

- **Flask**: Web framework
- **LangChain**: RAG orchestration
- **ChromaDB**: Vector database for embeddings
- **Sentence Transformers**: Text embeddings (all-MiniLM-L6-v2)
- **Groq**: LLM inference (Mixtral-8x7b)
- **Faster-Whisper**: Audio transcription

API Endpoints

- `GET /` - Home page (query interface)
- `GET /upload` - Upload page
- `GET /health` - Health check
- `POST /api/upload` - Upload and index files
- `POST /api/query` - Query the knowledge base

Processing Pipeline

1. **File Upload** → Temporary storage
2. **Audio Processing** → Transcription (if applicable)
3. **Document Loading** → Format-specific loaders
4. **Text Cleaning** → Remove extra whitespace, headers
5. **Semantic Chunking** → Intelligent splitting
6. **Embedding Generation** → Vector creation
7. **Storage** → ChromaDB persistence

Query Pipeline

1. **Question** → User input

2. **Embedding** → Question vectorization
3. **Retrieval** → Top 6 relevant chunks
4. **Context Building** → Combine chunks
5. **LLM Generation** → Answer synthesis
6. **Response** → Display to user

🐛 Troubleshooting

Common Issues

Issue: "No GROQ_API_KEY found"

- **Solution**: Ensure `.env` file exists with valid API key

Issue: Audio transcription fails

- **Solution**: Install FFmpeg: `sudo apt-get install ffmpeg` (Linux) or `brew install ffmpeg` (macOS)

Issue: ChromaDB errors

- **Solution**: Delete `chroma_db/` folder and restart

Issue: Import errors

- **Solution**: Ensure all dependencies installed: `pip install -r requirements.txt`

Issue: Port 5000 already in use

- **Solution**: Change port in `app.py`: `app.run(port=5001)`

📝 Configuration

Adjusting Retrieval

Edit `app.py`:

python

```
retriever = vectorstore.as_retriever(
    search_type="similarity",
    search_kwargs={"k": 6}  # Change number of chunks
)
```

Changing LLM Model

Edit `app.py`:

python

```
llm = ChatGroq(
    model="mixtral-8x7b-32768",  # Try: "llama2-70b-4096"
    temperature=0.3,             # Adjust creativity (0-1)
    groq_api_key=groq_api_key
)
```

File Size Limits

Edit `app.py`:

python

```
app.config['MAX_CONTENT_LENGTH'] = 100 * 1024 * 1024  # 100MB
```

🔒 Security Notes

- Never commit `.env` file to version control
- Keep API keys secure
- Validate all user inputs
- Implement rate limiting for production
- Use HTTPS in production

📚 Additional Resources

- LangChain Documentation
- ChromaDB Documentation
- Groq API Documentation
- Faster-Whisper

🤝 Contributing

1. Fork the repository
2. Create a feature branch
3. Make your changes
4. Test thoroughly
5. Submit a pull request

📄 License

MIT License - feel free to use for personal or commercial projects

🙏 Acknowledgments

- LangChain for RAG framework
- Anthropic for Claude API guidance
- Groq for fast LLM inference

- ChromaDB for vector storage
- Faster-Whisper for transcription

Built with ❤️ using Flask, LangChain, and ChromaDB