TECHNICAL DOCUMENTATION

Project Title: PDF-Based Question Answering System

Author: Preksha Rai

Date: 4th April 2025

Problem Statement

Building a PDF question-and-answer application with LangChain, Google LLM, and Chroma Vector Database.

Overview

This is a FastAPI-based application that allows users to upload PDF documents, extract and embed their content using HuggingFace embeddings, store it in ChromaDB (a vector database), and then answer user queries based on the document content using Google's Gemini API.

Project Structure

```
pdf_qa_app/
├── main.py                    # FastAPI entry point
├── config.py                  # API key, Gemini setup, and upload dir
├── logger_config.py           # Logging configuration
├── services/
│   ├── pdf_service.py         # Upload and processing logic
│   └── query_service.py       # Q&A logic
├── utils/
│   ├── embedding_utils.py  # HuggingFace embedding functions
│   └── chromadb_utils.py      # ChromaDB initialization
├── uploads/                   # Stores uploaded PDFs
├── chroma_db/                 # Vector DB directory
├── .env                       # Environment variables (e.g., API keys)
└── requirements.txt           # Python dependencies
```

Functional Modules

1. PDF Upload (/upload/)

- Method: POST
- Input: PDF file via form
- Process:
  Saves the PDF to the `uploads/` directory.
  Returns file path and success message.

2. PDF Processing (/process/)

- Method: POST
- Input: pdf_name (string)
- Process:
  Checks if the file exists.
  Checks if already processed in ChromaDB.
  Reads text from PDF using pypdf.
  Segments text into 500-character chunks.
  Generates embeddings using HuggingFace (all-MiniLM-L6-v2).
  Stores embeddings in ChromaDB.

3. Querying the Document (/query/)

- Method: GET
- Inputs:
  question (string)
  pdf_name (string)
- Process:
  Converts question into an embedding.
  Queries ChromaDB for top similar chunks.
  Constructs a prompt with the document context.
  Uses Gemini API (gemini-1.5-flash) to generate a JSON-structured answer.
  Returns the summary answer.

Technologies Used

Technology	Purpose
● FastAPI	Web API framework
● pypdf	PDF text extraction
● HuggingFace + LangChain	Sentence embeddings

● ChromaDB	Vector similarity search
● Gemini API	Answer generation
● dotenv	Load environment variables
● Logging	For event tracking and debugging

Environment Variables (.env)

```
GOOGLE_API_KEY=your_gemini_api_key_here
```

Installation & Setup

1. Clone the repository:

```
git clone https://github.com/<username>/<repository-name>.git
cd <repository-name>
```

2. Create virtual environment & activate:

```
python -m venv venv
source venv/bin/activate  # Windows: venv\Scripts\activate
```

3. Install dependencies:

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

4. Create .env file:

```
GOOGLE_API_KEY=your_gemini_api_key_here
```

5. Run the FastAPI app:

```
uvicorn main:app --reload
```

6. Access the API docs:

http://localhost:8000/docs

Example API Usage (via Swagger UI)

- Upload PDF

```
POST /upload/
Form-data: file = [upload.pdf]
```

- Process PDF

```
POST /process/
Body: { "pdf_name": "upload.pdf" }
```

- Ask Question
  ```
  GET /query/
  Params:
    question = "What is the main topic of this document?"
    pdf_name = "upload.pdf"
  ```

Response Format

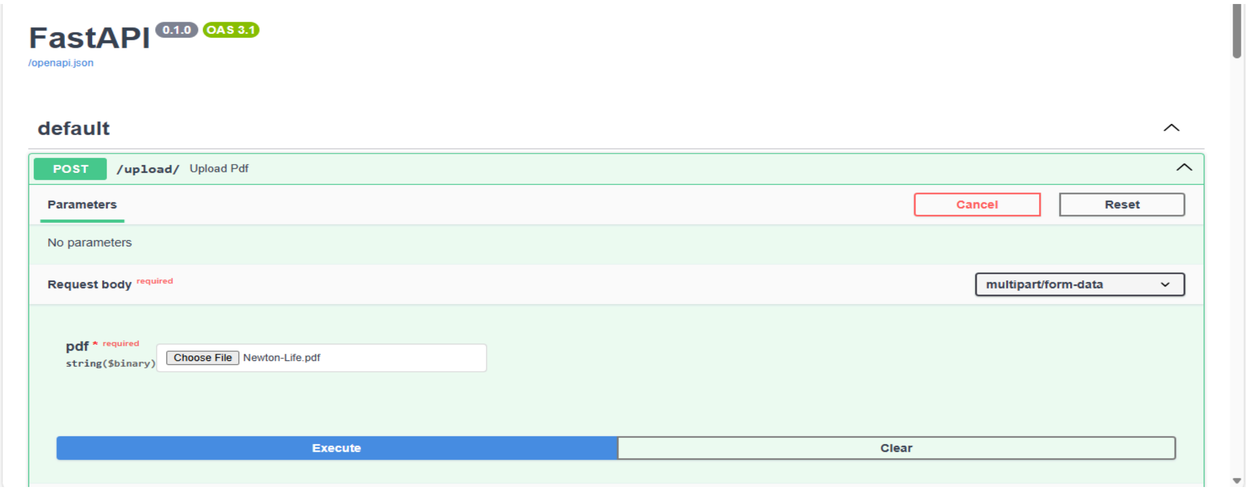
- On Successful Query:
  ```
  {
    "question": "What are the project goals?",

    "answer": "The project aims to build a recommendation system using AI..."
  }
  ```

- On Insufficient Info:
  ```
  {
    "question": "Explain quantum mechanics.",
    "answer": "Insufficient information available in the document."
  }
  ```

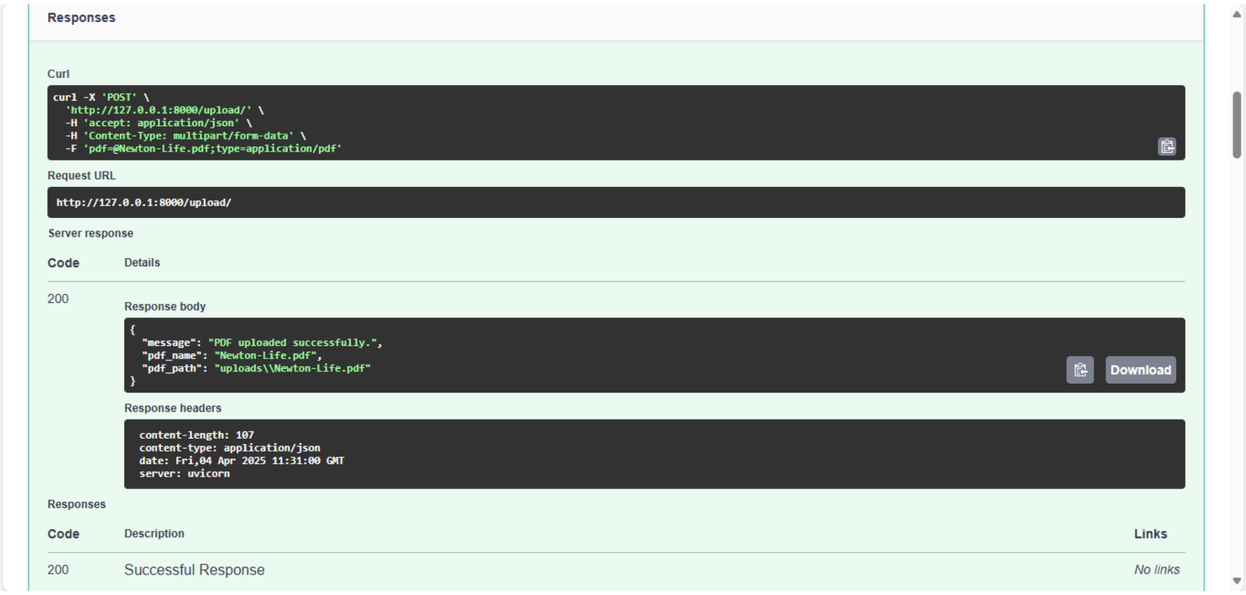
API Endpoint Demonstrations

1. Upload Endpoint – `/upload/`



No parameters

pdf * required
string($binary)
Choose File Newton-Life.pdf



Curl

```
curl -X 'POST' \
  'http://127.0.0.1:8000/upload/' \
  -H 'accept: application/json' \
  -H 'Content-Type: multipart/form-data' \
  -F 'pdf=@Newton-Life.pdf;type=application/pdf'
```

Request URL

```
http://127.0.0.1:8000/upload/
```

Server response

Code	Details
200	Response body

```
{
  "message": "PDF uploaded successfully.",
  "pdf_name": "Newton-Life.pdf",
  "pdf_path": "uploads\\Newton-Life.pdf"
}
```

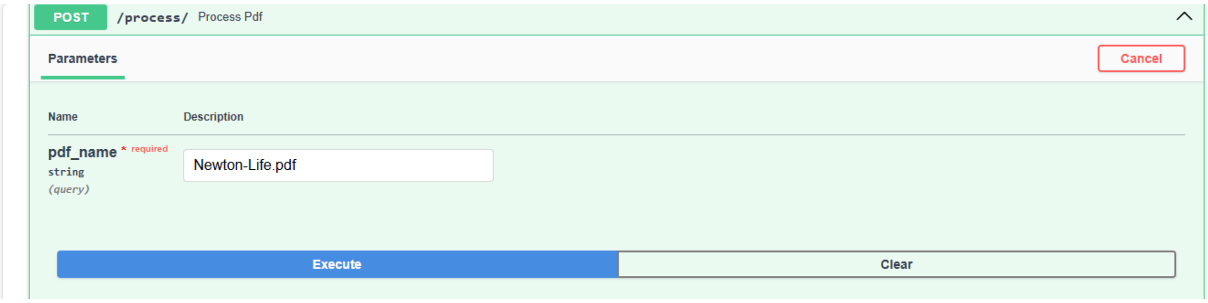
Response headers

```
content-length: 107
content-type: application/json
date: Fri,04 Apr 2025 11:31:00 GMT
server: uvicorn
```

Responses

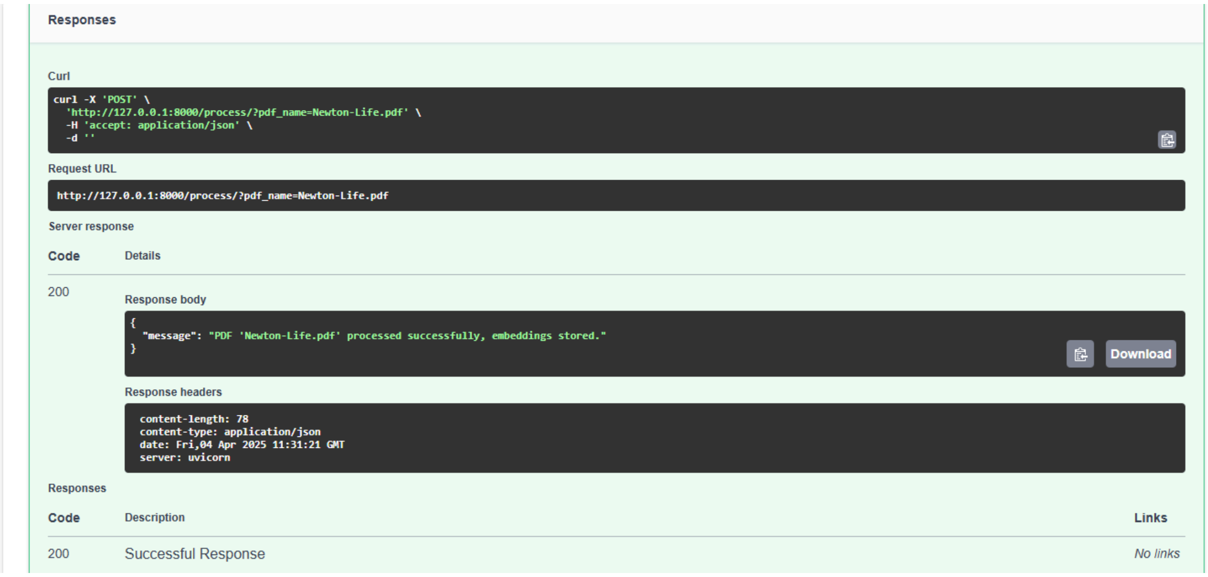
Code	Description	Links
200	Successful Response	*No links*

2. Process Endpoint – /process/



pdf_name * required
string
(query)

Newton-Life.pdf



Curl

```
curl -X 'POST' \
  'http://127.0.0.1:8000/process/?pdf_name=Newton-Life.pdf' \
  -H 'accept: application/json' \
  -d ''
```

Request URL

```
http://127.0.0.1:8000/process/?pdf_name=Newton-Life.pdf
```

Server response

Code Details

200

Response body

```
{
  "message": "PDF 'Newton-Life.pdf' processed successfully, embeddings stored."
}
```

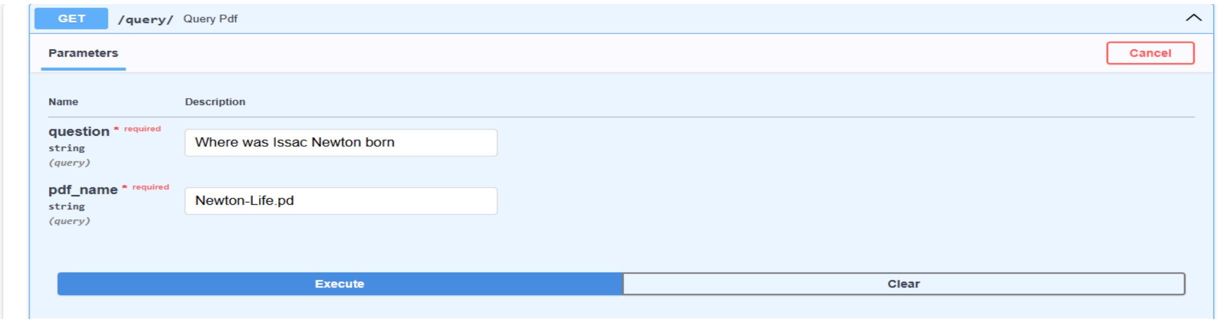
Response headers

```
 content-length: 78
 content-type: application/json
 date: Fri,04 Apr 2025 11:31:21 GMT
 server: uvicorn
```

Responses

Code	Description	Links
200	Successful Response	*No links*

3. Query Endpoint – /query/

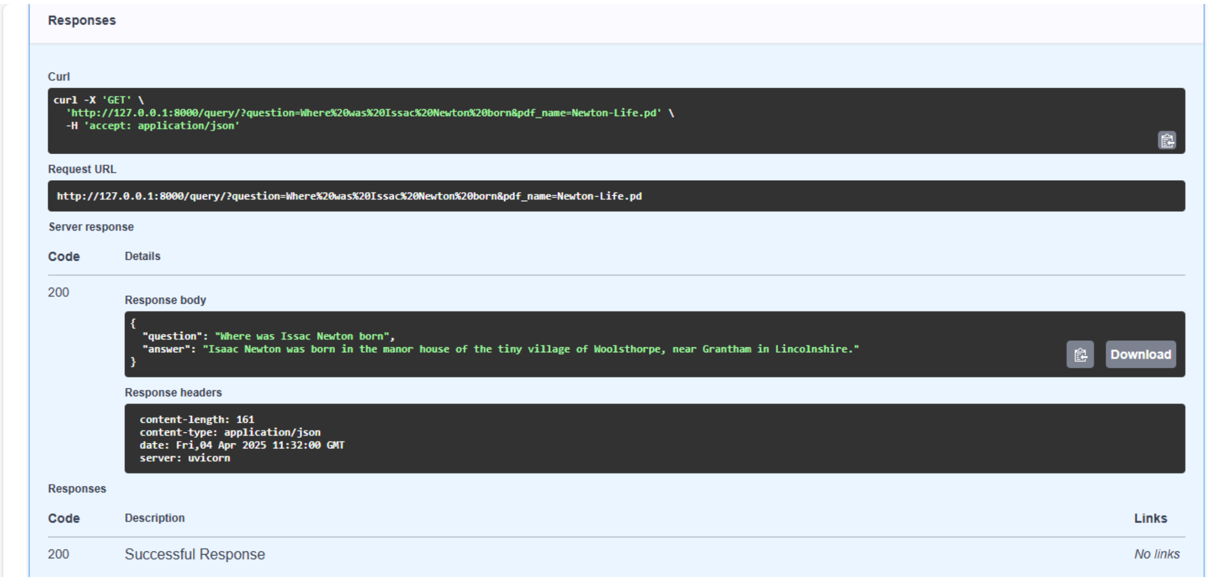


question * required
string
(query)

Where was Issac Newton born

pdf_name * required
string
(query)

Newton-Life.pd



Curl

```
curl -X 'GET' \
  'http://127.0.0.1:8000/query/?question=Where%20was%20Issac%20Newton%20born&pdf_name=Newton-Life.pd' \
  -H 'accept: application/json'
```

Request URL

```
http://127.0.0.1:8000/query/?question=Where%20was%20Issac%20Newton%20born&pdf_name=Newton-Life.pd
```

Server response

Code	Details
200	Response body

```
{
  "question": "Where was Issac Newton born",
  "answer": "Isaac Newton was born in the manor house of the tiny village of Woolsthorpe, near Grantham in Lincolnshire."
}
```

Response headers

```
 content-length: 161
 content-type: application/json
 date: Fri,04 Apr 2025 11:32:00 GMT
 server: uvicorn
```

Responses

Code	Description	Links
200	Successful Response	*No links*

ChromaDB Info

Storage Path: ./chroma_db/

Collection Name: pdf_embeddings

Each document chunk is stored with:
- id: e.g., report_0, report_1, ...
- embedding: 384-dimensional vector
- document: corresponding text

Error Handling

Status Code	Cause
400	Empty or non-text PDF
404	PDF not found
500	Upload, embedding, DB, or Gemini errors