Project Report: Document Research & Theme Identification Chatbot

~ *Heerak kashyap*

1. Project Overview : The goal was to build a backend system that allows users to upload documents (PDFs, images), extract their text using OCR, store and index the content for semantic search, and provide an API for querying documents by meaning.

2. Folder Structure :

```
chatbot_theme_identifier/
├── backend/
│   ├── app/
│   │   ├── api/
│   │   │   ├── router.py
│   │   │   └── __init__.py
│   │   ├── core/
│   │   ├── models/
│   │   ├── services/
│   │   │   └── vector_store.py
│   ├── data/
│   │   └── uploads/
│   ├── main.py
│   ├── config.py
│   ├── Dockerfile
│   └── requirements.txt
├── docs/
├── tests/
├── demo/
└── README.md
```

3. Technology Stack :

- **Backend Framework:** FastAPI (Python)
- **OCR:** Tesseract (via pytesseract, Pillow)
- **PDF to Image:** pdf2image (requires Poppler)

- **Vector Database:** ChromaDB
- **Embeddings:** sentence-transformers (all-MiniLM-L6-v2)
- **Other:** Uvicorn (ASGI server)

4. Key Features Implemented :

A. Document Upload & OCR

- Users can upload PDFs or images via the /upload-document endpoint.
- Uploaded files are saved in backend/data/uploads/.
- OCR is performed:
- Images: Directly via Tesseract.
- PDFs: Each page is converted to an image (Poppler), then OCR is run.
- Extracted text is saved as a .txt file in the same folder.

B. Vector Store & Semantic Search

- Extracted text is embedded using a sentence transformer.
- Embeddings and metadata are stored in ChromaDB.
- /search endpoint allows users to query documents by semantic similarity, not just keywords.

C. API Endpoints

- GET /ping: Health check.
- POST /upload-document: Upload a document, extract text, store in vector DB.
- GET /search: Query documents by meaning.

5. Installation & Setup :

A. Python Dependencies

Install all dependencies:

pip install -r chatbot_theme_identifier/backend/requirements.txt

B. Tesseract OCR - Extract and add the bin folder to the PATH in env variables

C. Poppler (for PDF support) - Extract and add the bin folder to the PATH in env variables

D. Run the Server

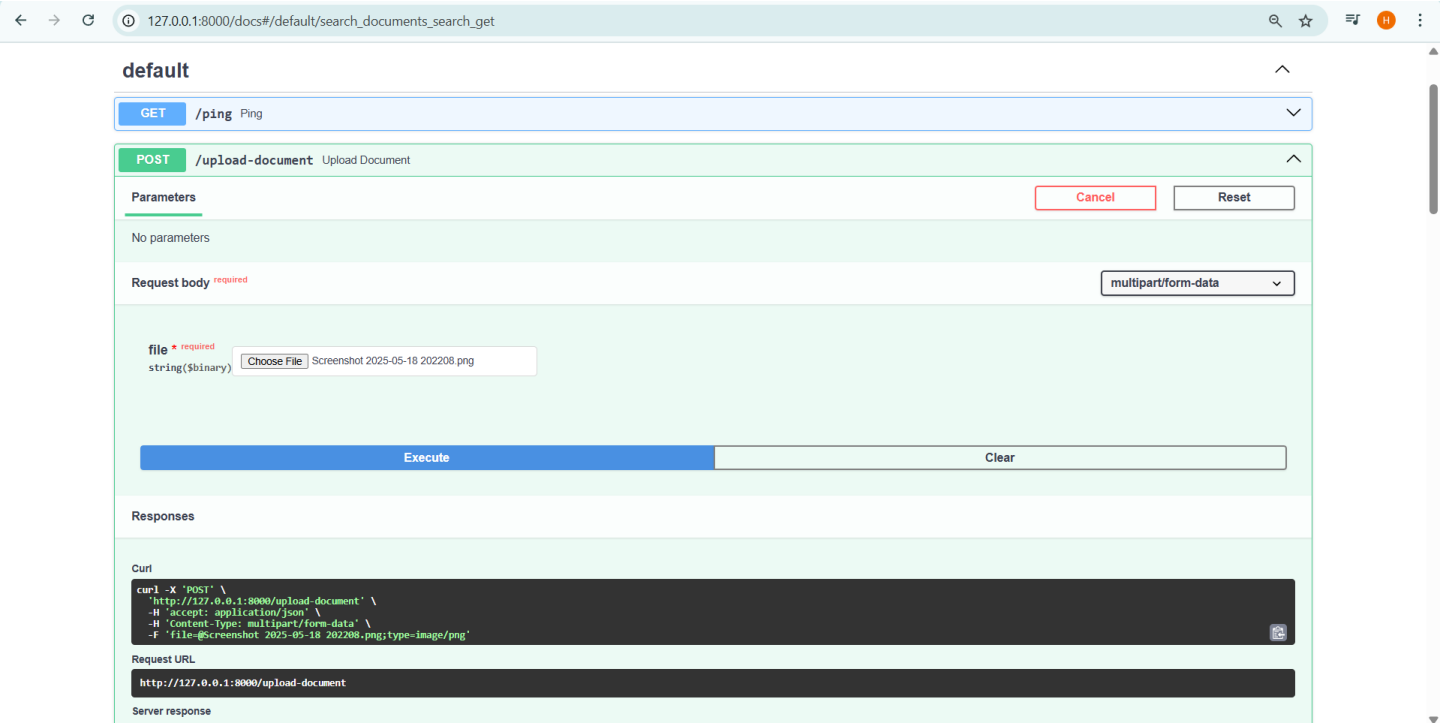
cd chatbot_theme_identifier/backend

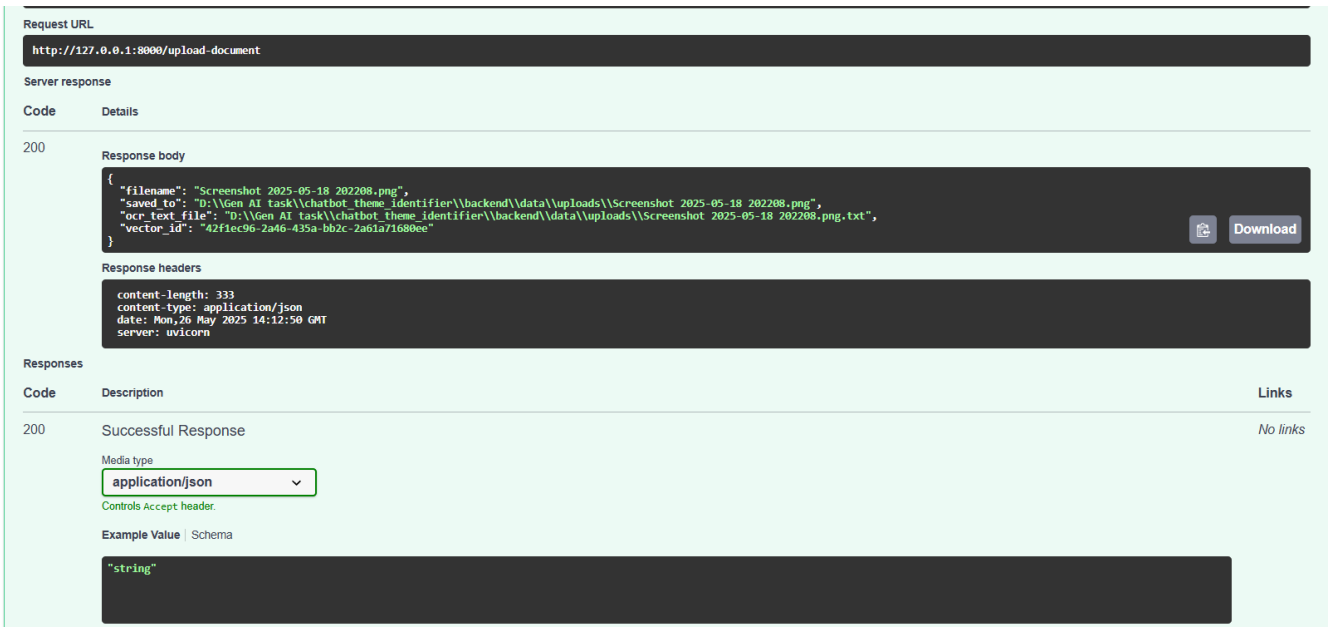
uvicorn main:app –reload - Access the API docs at http://127.0.0.1:8000/docs.

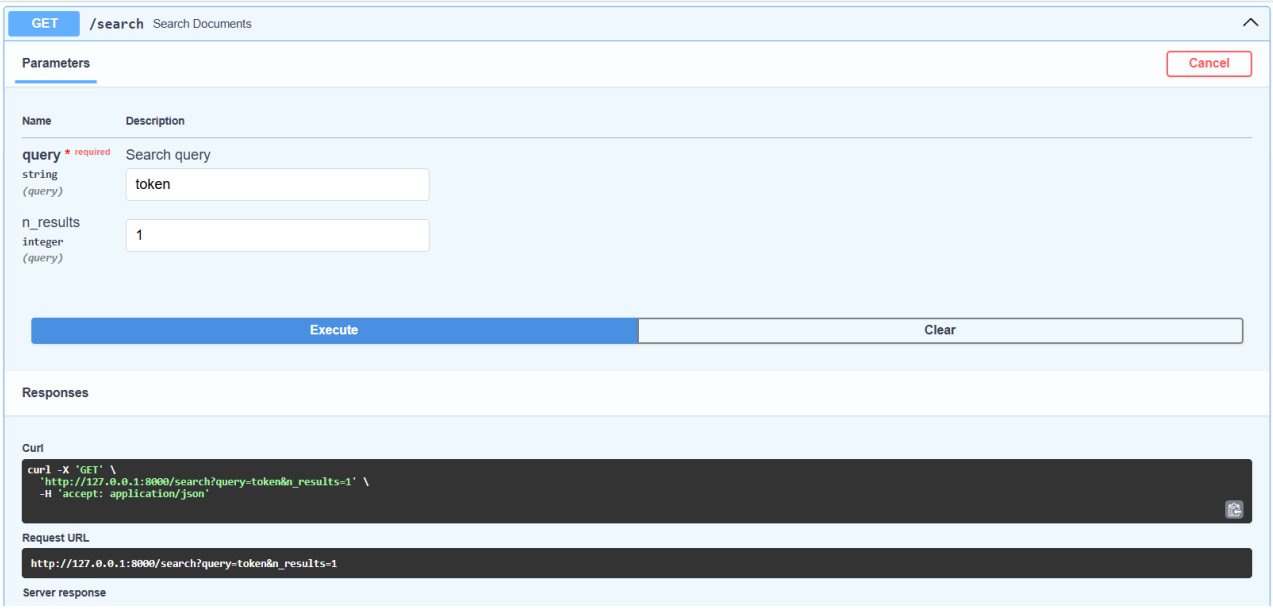
6. Testing & Results :

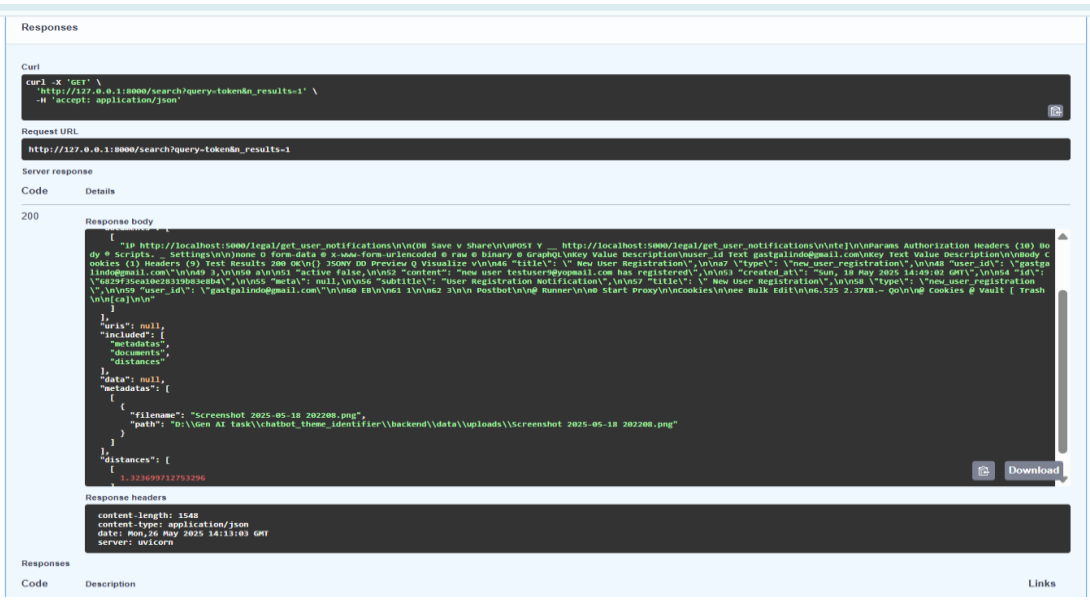
- **Upload:** Used /upload-document to upload PDFs and images.
- **OCR Output:** Verified .txt files were created with extracted text.
- **Semantic Search:** Used /search endpoint to query for relevant documents by meaning.
- **Poppler & Tesseract:** Verified installations using pdfinfo -v and tesseract --version.

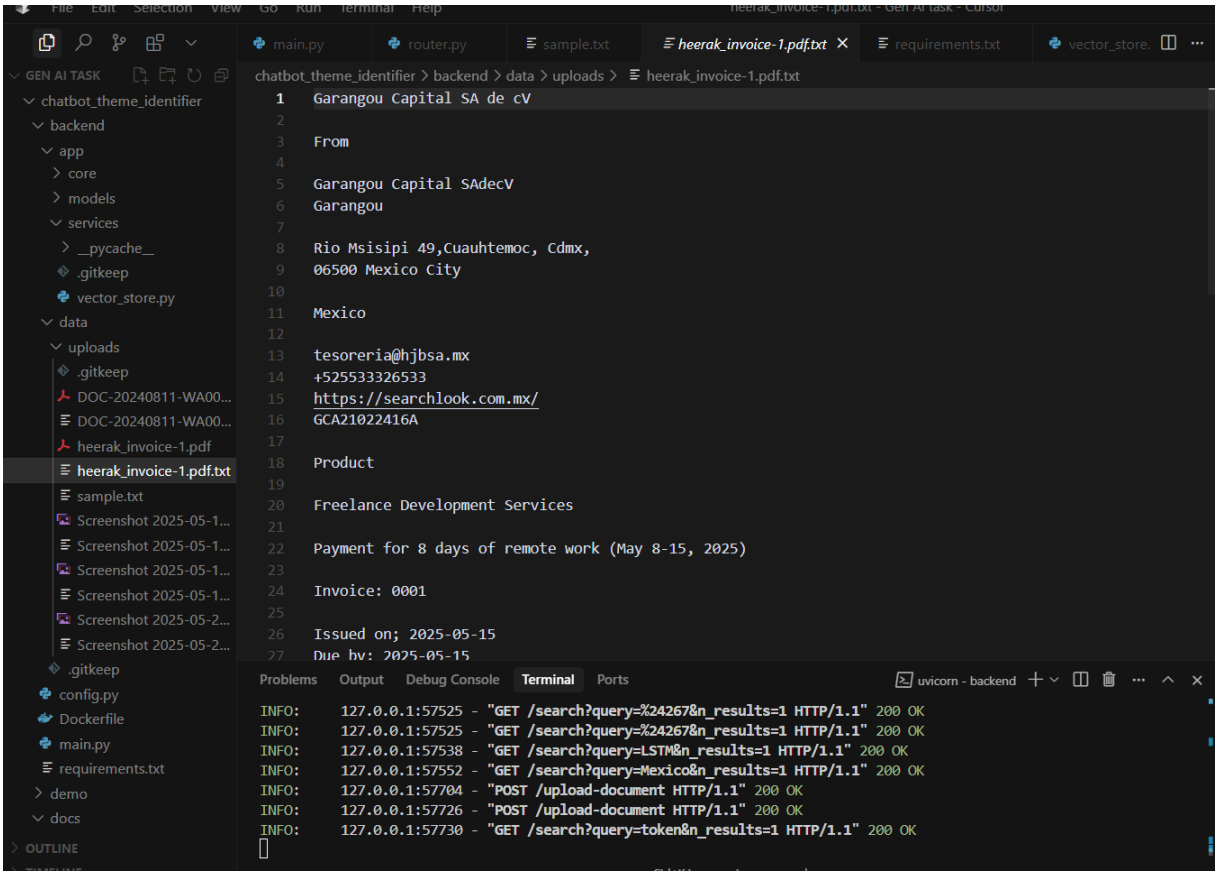
ON LOCAL SERVER : we can see the results as given below:


127.0.0.1:8000/docs#/default/search_documents_search_get
default
GET /ping Ping
POST /upload-document Upload Document
Parameters
Cancel
Reset
No parameters
Request body required
multipart/form-data
file * required
string($binary)
Choose File
Screenshot 2025-05-18 202208.png
Execute
Clear
Responses
Curl
curl -X 'POST' \
'http://127.0.0.1:8000/upload-document' \
-H 'accept: application/json' \
-H 'Content-Type: multipart/form-data' \
-F 'file=@Screenshot 2025-05-18 202208.png;type=image/png'
Request URL
http://127.0.0.1:8000/upload-document
Server response


Request URL
http://127.0.0.1:8000/upload-document
Server response
Code
Details
200
Response body
{
"filename": "Screenshot 2025-05-18 202208.png",
"saved_to": "D:\\Gen AI task\\chatbot_theme_identifier\\backend\\data\\uploads\\Screenshot 2025-05-18 202208.png",
"ocr_text_file": "D:\\Gen AI task\\chatbot_theme_identifier\\backend\\data\\uploads\\Screenshot 2025-05-18 202208.png.txt",
"vector_id": "42f1ec96-2a46-435a-bb2c-2a61a71680ee"
}
Download
Response headers
content-length: 333
content-type: application/json
date: Mon,26 May 2025 14:12:50 GMT
server: uvicorn
Responses
Code
Description
Links
200
Successful Response
No links
Media type
application/json
Controls Accept header.
Example Value
Schema
"string"

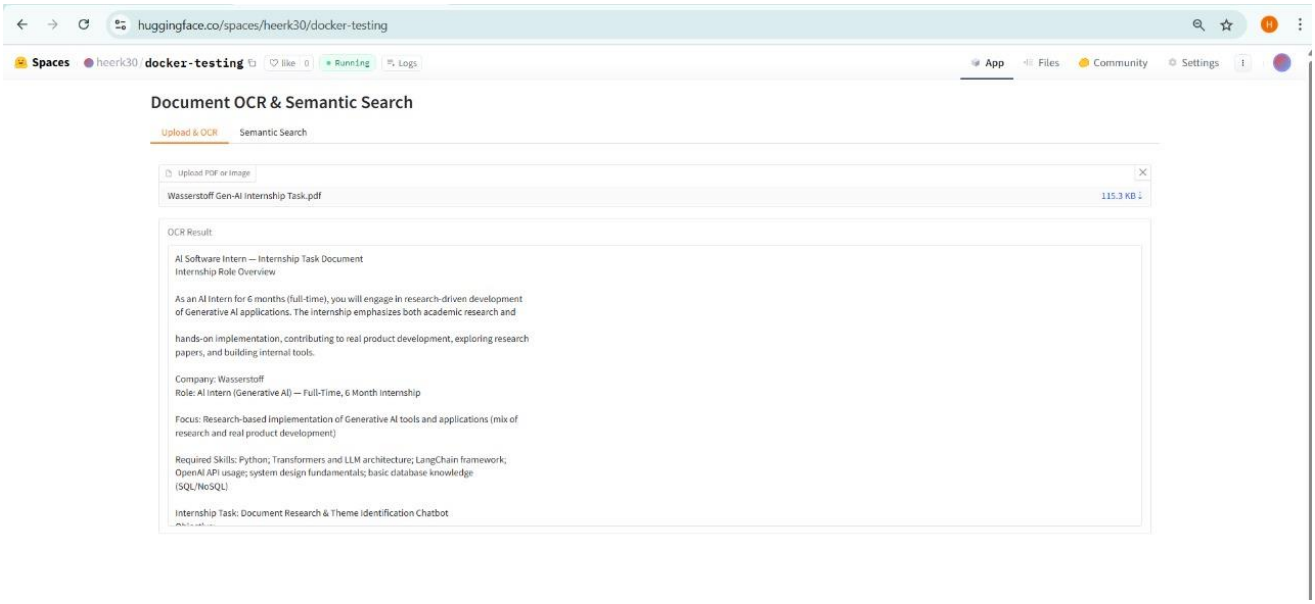

GET
/search
Search Documents
Parameters
Cancel
Name
Description
query * required
string
(query)
Search query
token
n_results
integer
(query)
1
Execute
Clear
Responses
Curl
curl -X 'GET' \
'http://127.0.0.1:8000/search?query=token&n_results=1' \
-H 'accept: application/json'
Request URL
http://127.0.0.1:8000/search?query=token&n_results=1
Server response


Responses
Curl
curl -X 'GET' \
'http://127.0.0.1:8000/search?query=token&n_results=1' \
-H 'accept: application/json'
Request URL
http://127.0.0.1:8000/search?query=token&n_results=1
Server response
Code
Details
200
Response body
Download
Response headers
content-length: 1548
content-type: application/json
date: Mon,26 May 2025 14:13:03 GMT
server: uvicorn
Responses
Code
Description
Links



- FastAPI docs UI with endpoints.
- .txt file with extracted text.

ON HuggingFace :



10. Conclusion :

The backend for the Document Research &Theme Identification Chatbot is fully functional, supporting document upload, OCR, semantic indexing, and search. The system is modular, extensible, and ready for further development or deployment.