AI-based Applications
From RAGs to riches

July 2025

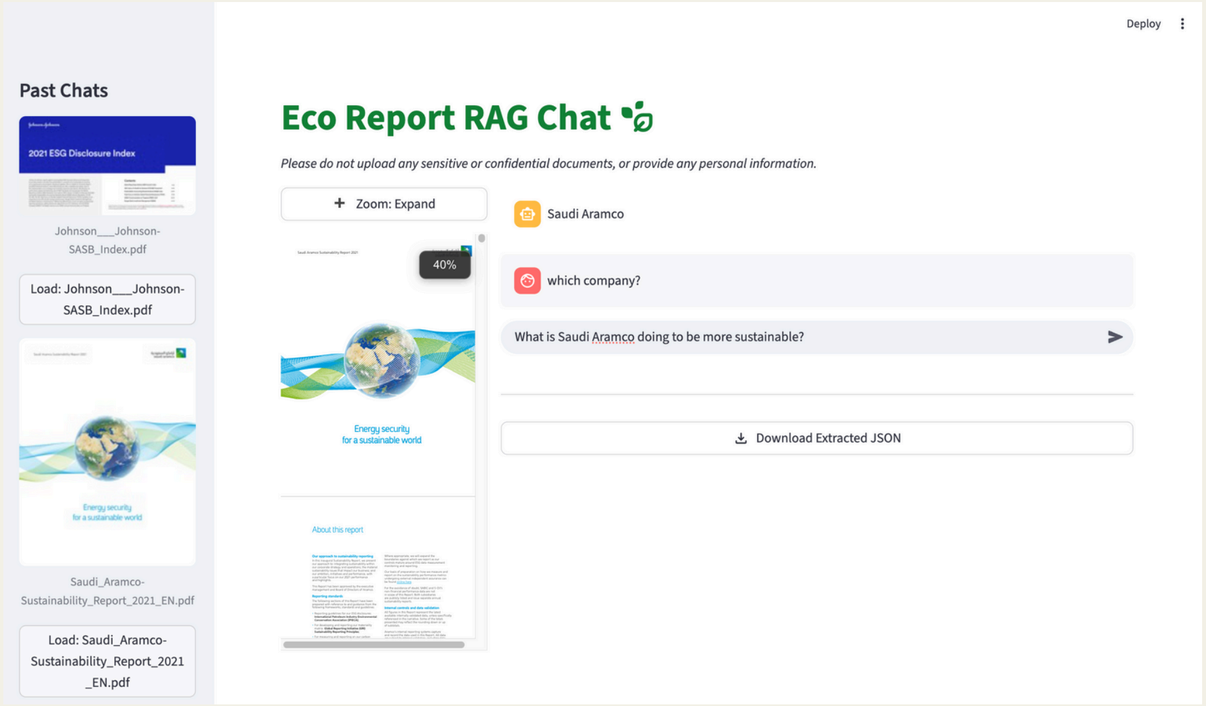
Presented by
Joschua Levi Prieß, Leo Strietzel, Margareta Marie Dittrich, Mahsa Alaem, Victor Caplan

Goal of Project

Application that allows users to chat with a pdf with following capabilities

- High accuracy and high speed
- JSON file download
- Chat function
- Conversational awareness
- PDF viewer
- Additional feature: chat history and user responsiveness

→ Backend focus on performance, speed, and compute time



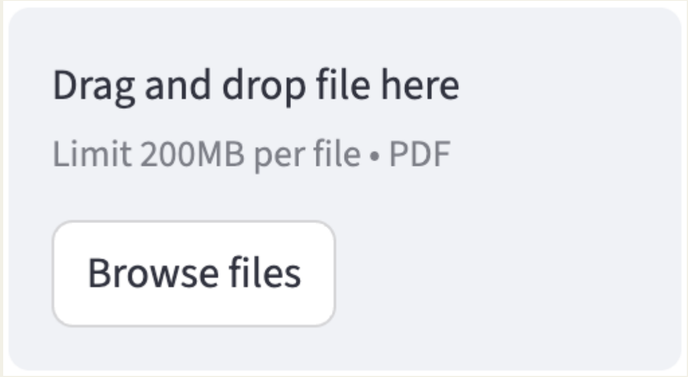
Live Demo

Frontend – Key Features

Key UI Features

PDF Upload & Chat Initialization

- **new_chat(): Handles new file uploads and starts a new backend session**

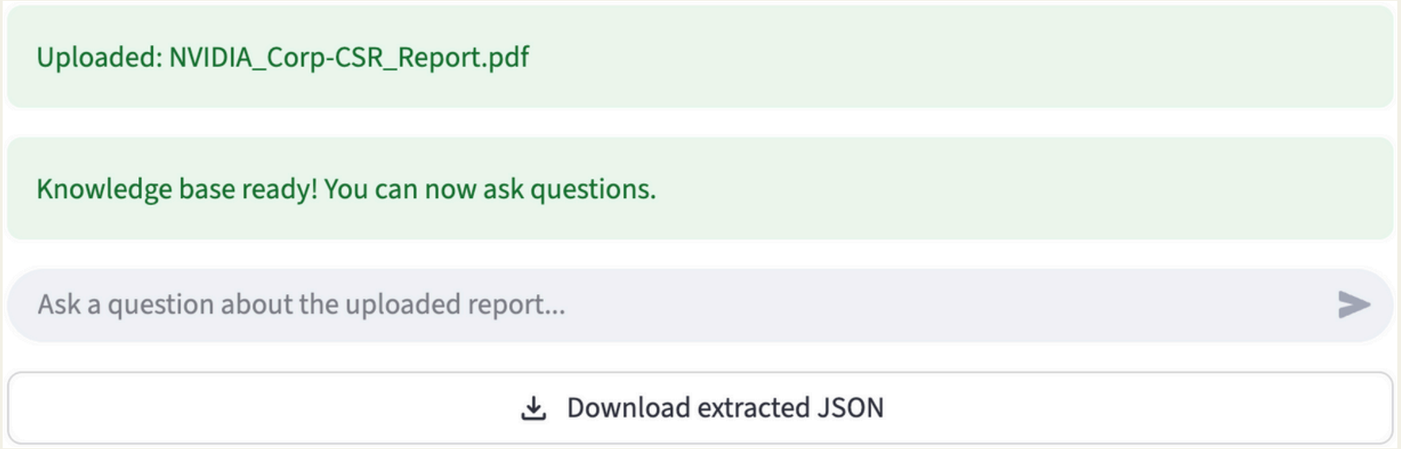


Chat Interface

- **Uses st.session_state.messages to persist conversation**
- **Displays assistant and user messages dynamically**

Downloadable Results

- **extract_json_for_download() & convert_json_for_download(): Export answers as structured JSON**



Frontend – Bonus Features

Key UI Features

PDF Preview (BONUS)

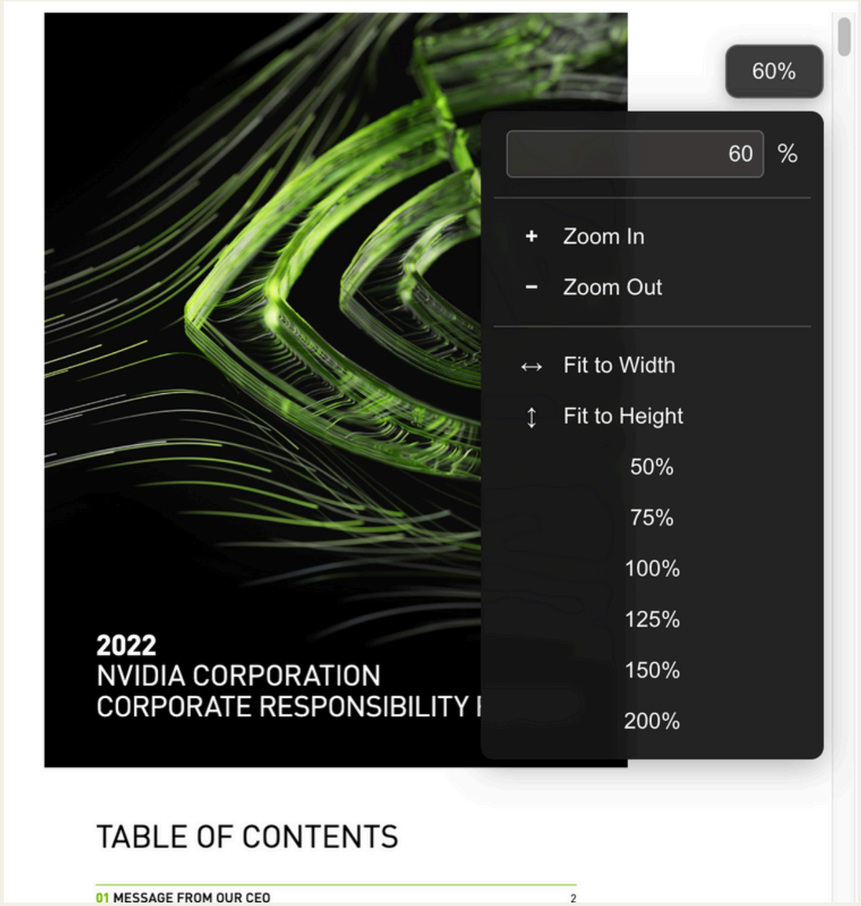
- Shows mini-previews using pdf2image and PIL in sidebar
- Sidebar built via display_past_chats() to list and reload previous interactions

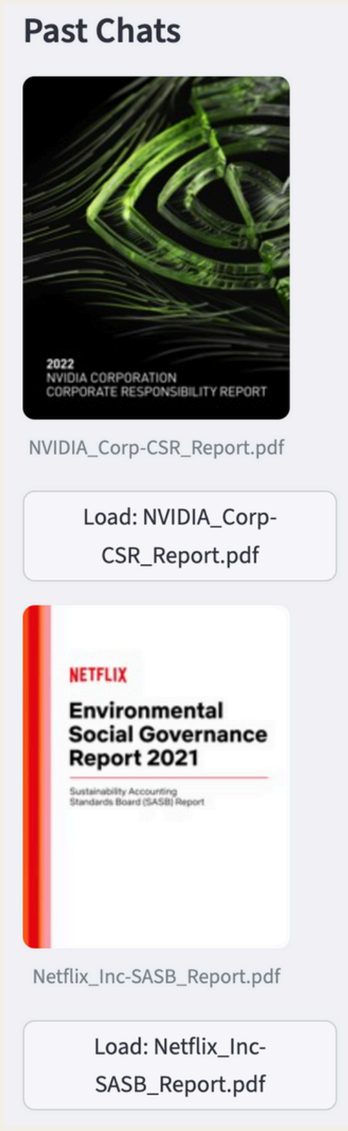
Extra UX (BONUS)

- toggle_pdf_wide() improves PDF view experience.

Past PDF (BONUS)

- Shows past PDFs
- Keeps them stored, so the user can go back



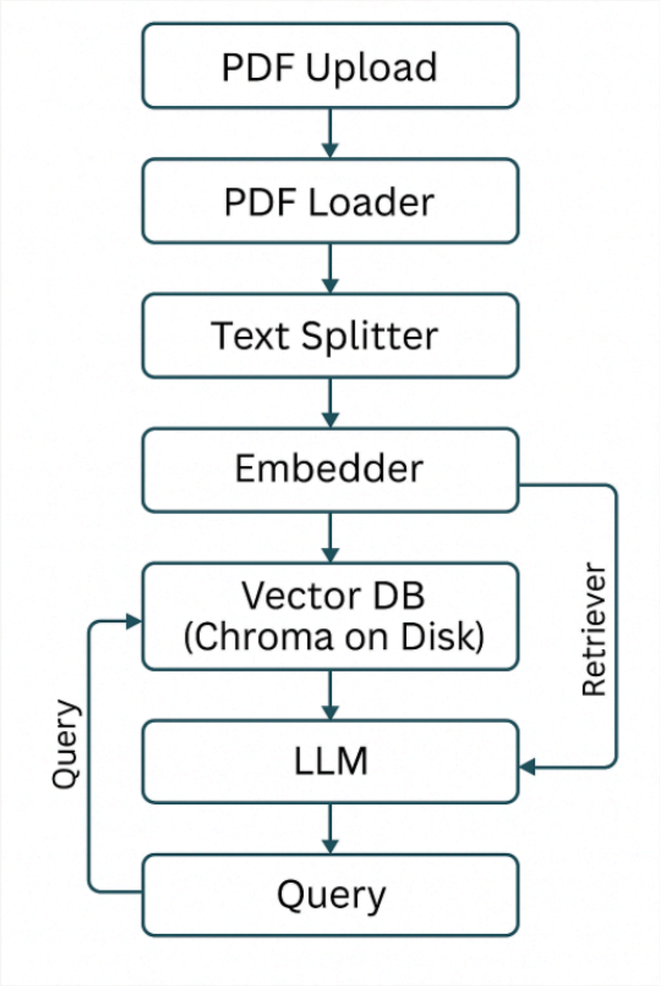


Backend

Why we chose specific components for the Architecture

PDF Upload (Streamlit)	Simple and fast file handling with built-in session state and file cache.
PDF Loader (pypdf)	Efficient and lightweight parsing compared to heavier alternatives.
Text Splitter	RecursiveCharacterTextSplitter ensures semantic chunking for better retrieval.
Vector DB (Chroma on Disk)	Disk storage drastically reduces RAM usage and improves scalability.
LLM	High-quality, natural-sounding answers and Free of charge with easy LangChain integration.
Query Box / Chat Interface	Streamlit-based chat keeps the interface intuitive and clean.

Architecture



Backend

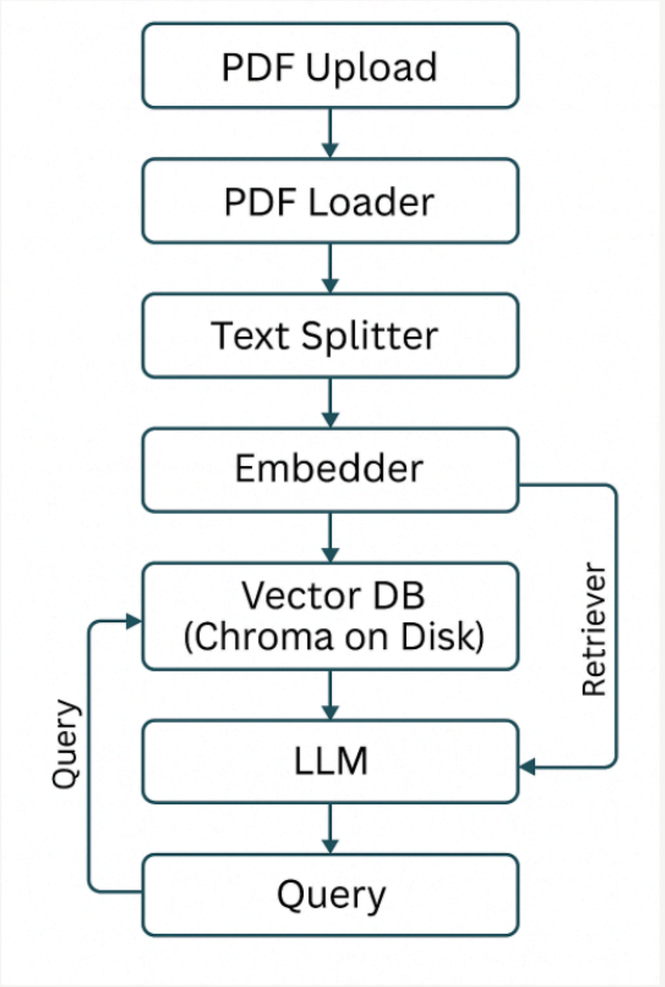
Our Codebase

```
# Import RAG and embedding libraries
from langchain_docling.loader import DoclingLoader…

# load pdf and split
def read_and_split_pdf(file_path, chunk_size=1000, chunk_overlap=200):

# Embeddings class for AcademicCloud API
class APIEmbeddings(Embeddings):…

# local Embeddings class for AcademicCloud API
# we can switch between local and cloud embeddings in case the compute
class LocalEmbeddings(Embeddings):…

# vectorize texts and store in chroma db
def embed_and_store(texts, index_path=None):…

def set_retriever(vector_store, k=10, use_mmr=True):…

# initialising llm
def set_llm():…


# initialising llm
def set_json_llm():…

# extracting info and creating json
def extract_info_as_json(llm, retriever, pdf_name):…
```

Benefits of this structure:

- Easy to understand
- Modularity -> easy to edit
- Steps are closey linked to traditional RAG implementations
- Easily usable forother rag tasks
- Separate form front end
- Closely refelcts archetecture

Architecture



Speed and RAM usage

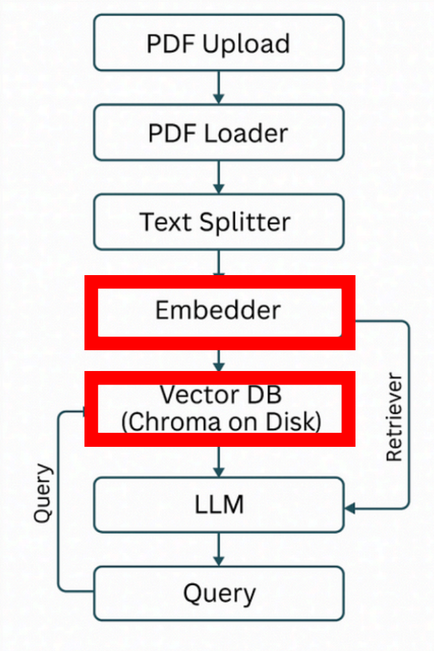
Problems

- High RAM usage from PDF loader
- Slow embedding (30s for small PDF)
- Vector DB in RAM caused memory pressure (more than 10 GB → got “OutOfMemory”-Error)
- Chunking too aggressive and unbalanced

Consequences

- At least 30 sec of loading for smallest document
- Using at least 9 GB of RAM until Memory Error

Architecture



Speed and RAM usage

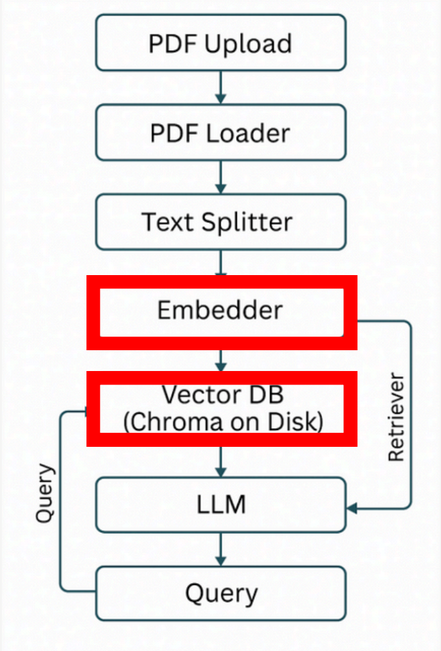
Solution

- Switched to pypdf loader for lightweight document loading
- Batched embeddings
- Moved to on-Disk ChromaDB (faster startup, lower RAM usage)
- Tuned chunck size and Overlap using “RecurseveCharacterTextSplitter”

Consequences

- Largest PDFs can be processed in 10–15 seconds
- Uses about 3GB of RAM

Architecture



Speed vs Quality

Model:

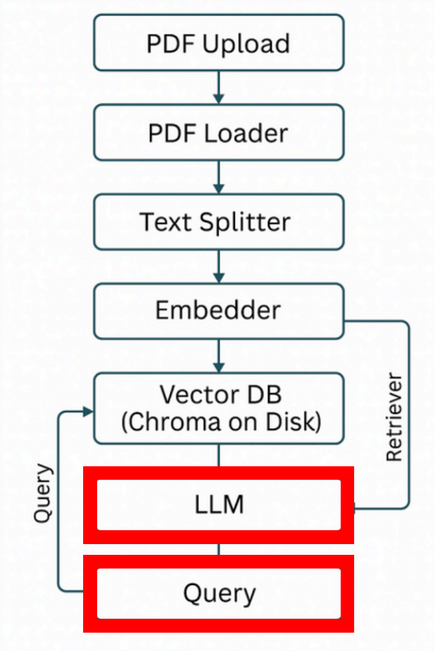
- Model hybrid of llama-3.1-sauerkrautlm-70b-instruct and meta-llama-3.1-8b-rag
- Best of both worlds

JSON and Query:

- Default system prompt is decent
- Added context for PDF usage
- Specified to save short responses in the JSON file

```
schema_questions = {
    "CO2": "What does the document say about CO2 emissions? please be as concise as possible",
    "NOX": "What information is provided about NOX emissions? please be as concise as possible",
    "Number of Electric Vehicles": "How many electric vehicles are mentioned or estimated? ",
    "Impact": "What impact is discussed in the document? please be as concise as possible",
    "Risks": "What risks are described? please be as concise as possible",
    "Opportunities": "What opportunities are mentioned? please be as concise as possible",
    "Strategy": "What strategy is outlined? please be as concise as possible",
    "Actions": "What actions are proposed or taken? please be as concise as possible",
    "Adopted_policies": "What policies have been adopted? please be as concise as possible",
    "Targets": "What targets are defined in the document? please be as concise as possible"
}
```

Architecture



Custom Prompt for Chat

Ensures that Model:

- Considers itself as an expert
- Gives context
- Cites page numbers from chucks
- Hallucinates less

```
from langchain.prompts import PromptTemplate

custom_prompt = PromptTemplate(
    input_variables=["context", "question"],
    template="""
You are an expert analyst for ESG reports.
Use the context below to answer the question.
Always cite the page number(s) in parentheses after each fact, e.g., "(Page 3)".
If you don't know, say "I don't know" and provide the most likely explanation.
Context:
{context}
Question:
{question}
Answer:
"""
)
```

Docker Deployability

Dockerfile

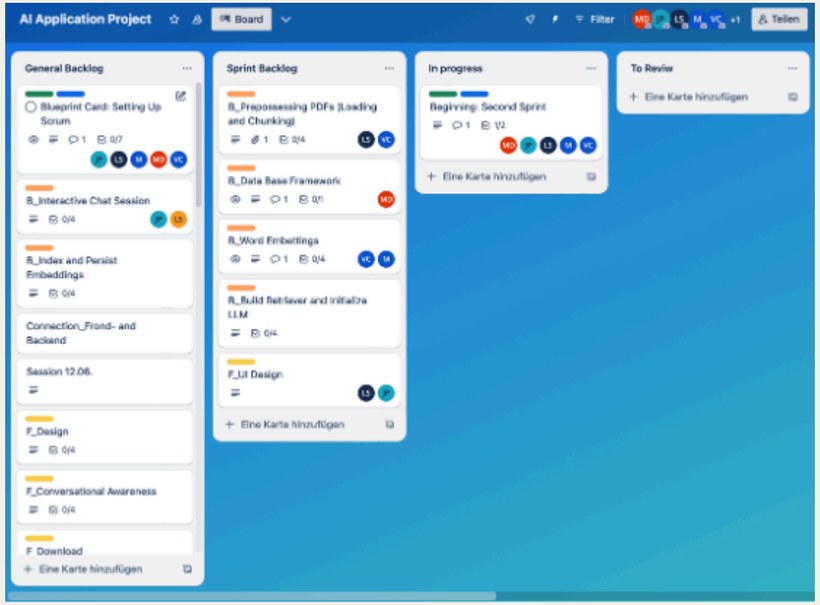
```dockerfile
# Use the official Python image as a base
FROM python:3.11-slim

# Set work directory
WORKDIR /app

# Install system dependencies
RUN apt-get update && apt-get install -y --no-install-recommends \
    build-essential \
    poppler-utils \
    && rm -rf /var/lib/apt/lists/*

# Copy requirements.txt
COPY requirements.txt .

# Install Python dependencies
RUN pip install --upgrade pip && pip install -r requirements.txt

# Copy project files
COPY . .

# Expose Streamlit port
EXPOSE 8080

# Run Streamlit app
CMD ["streamlit", "run", "app.py"]
```

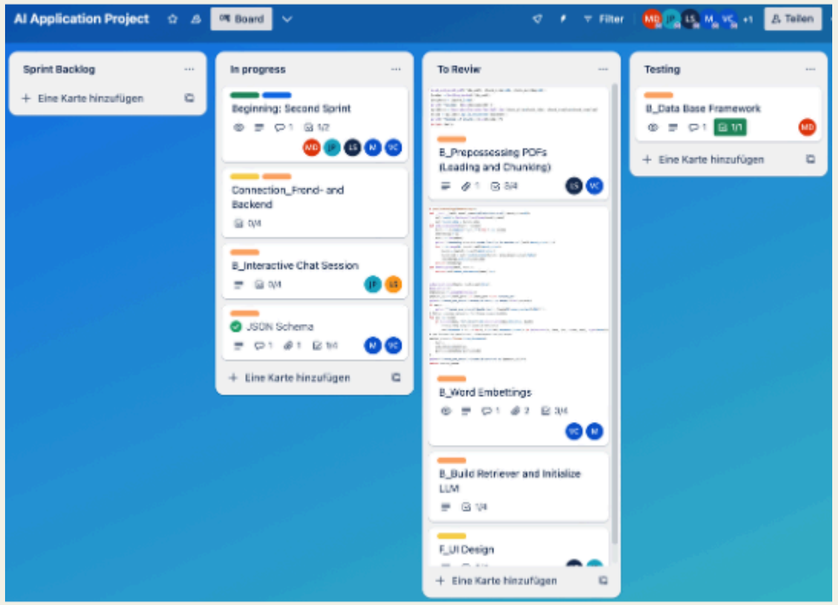
Compose File

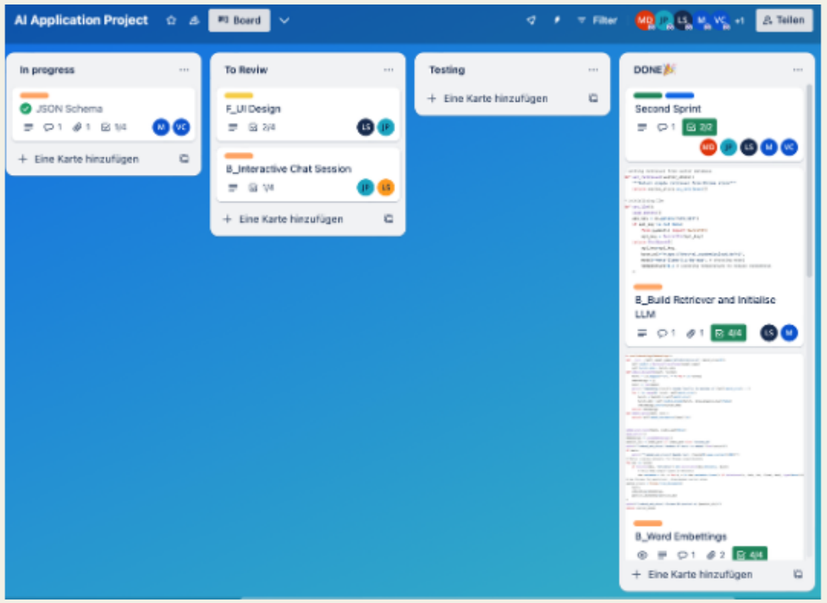
```yaml
services:
  ▷Run Service
  streamlit:
    container_name: "RAG_eco_reports"
    build:
      dockerfile: ./Dockerfile
      context: ./
    ports:
      - '8080:8080'
    command: streamlit run main.py --server.port=8080 --server.address=0.0.0.0
```

Group Work – Trello Sprints

Example Sprint 2 – Start – Middle - End

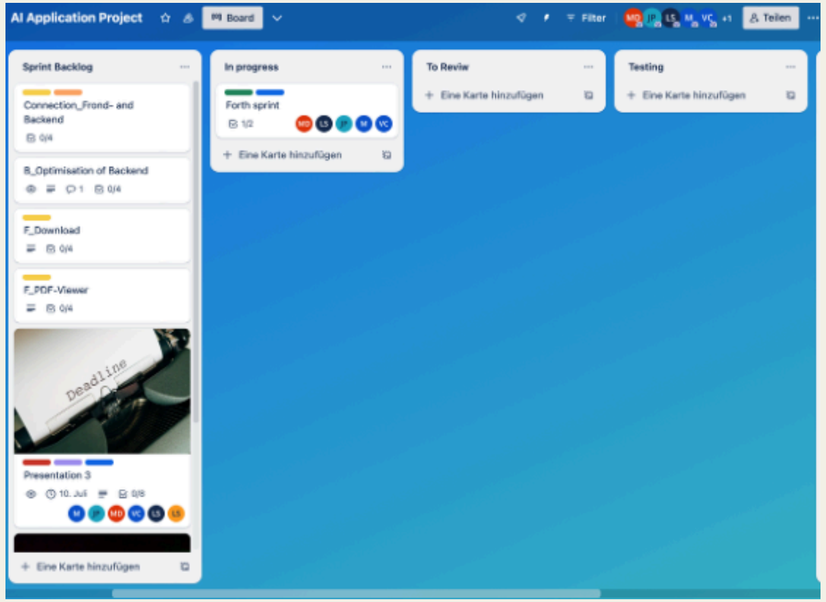


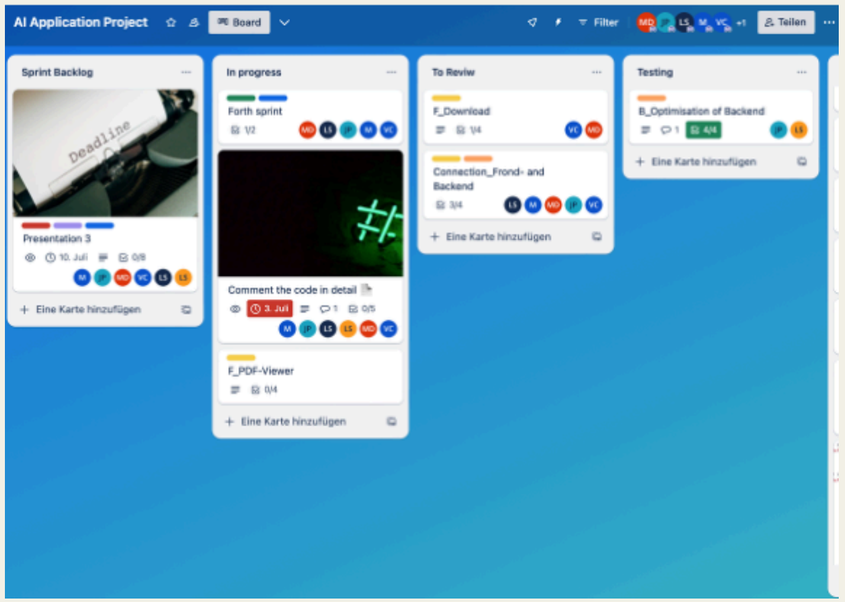


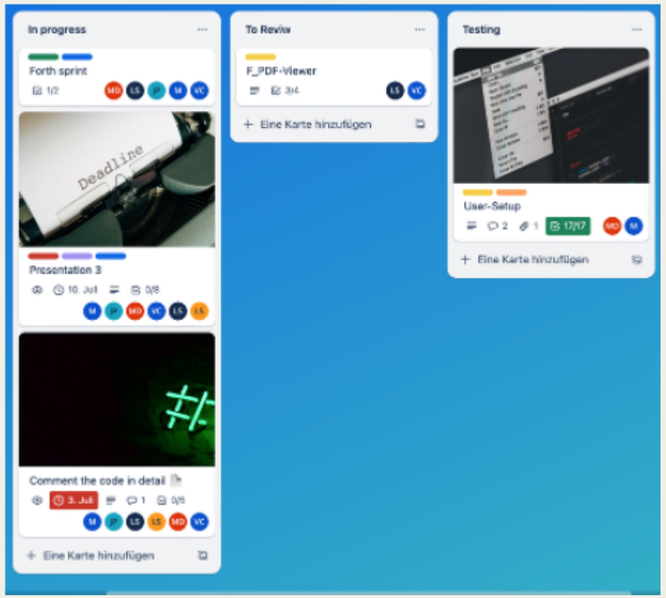


Group Work – Trello Sprints

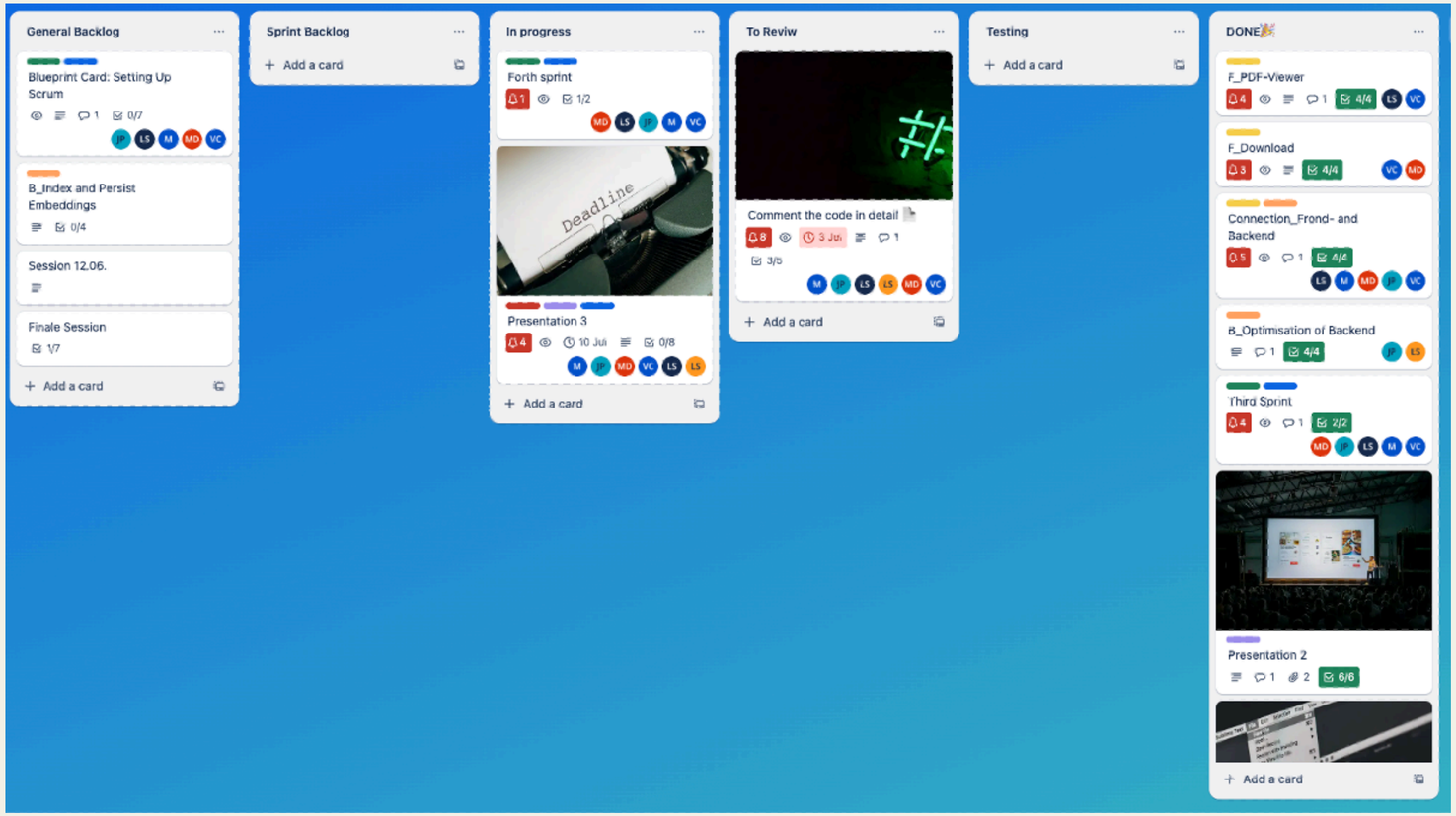
Example Sprint 4 – Start – Middle - End





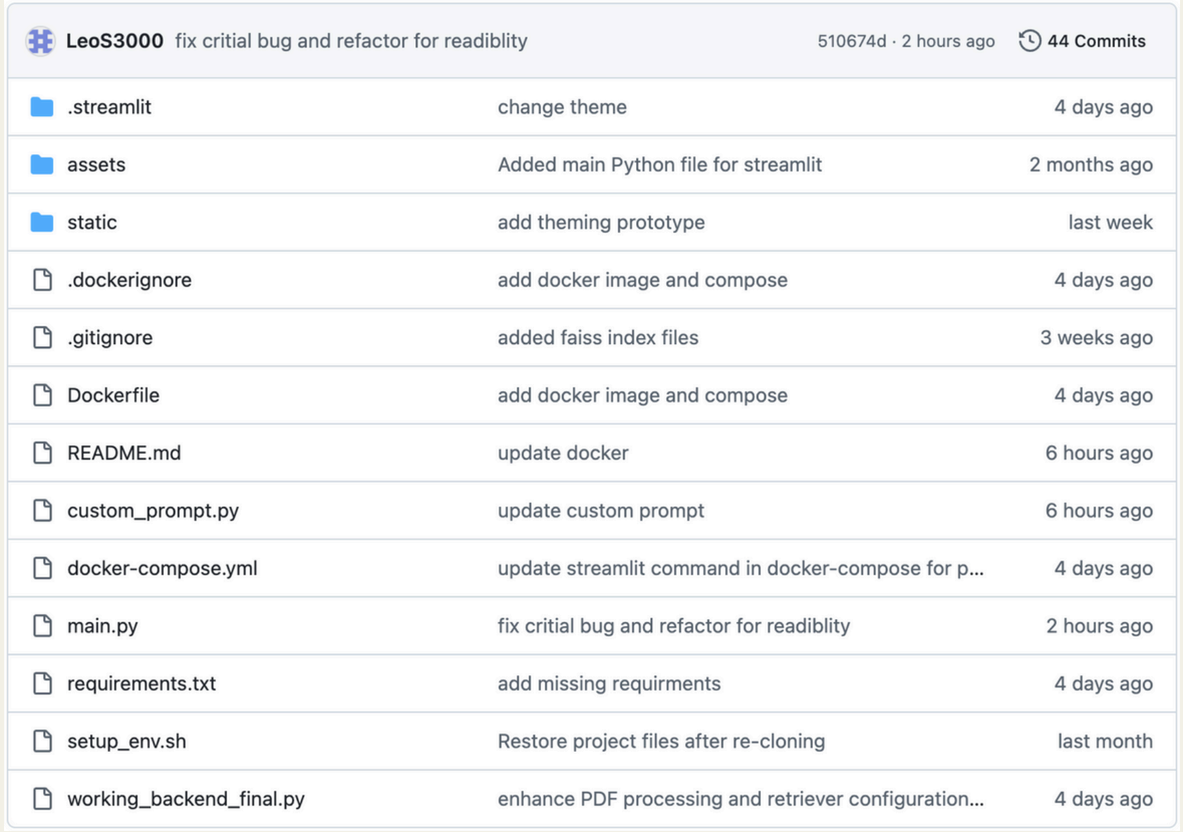


Group Work – Trello

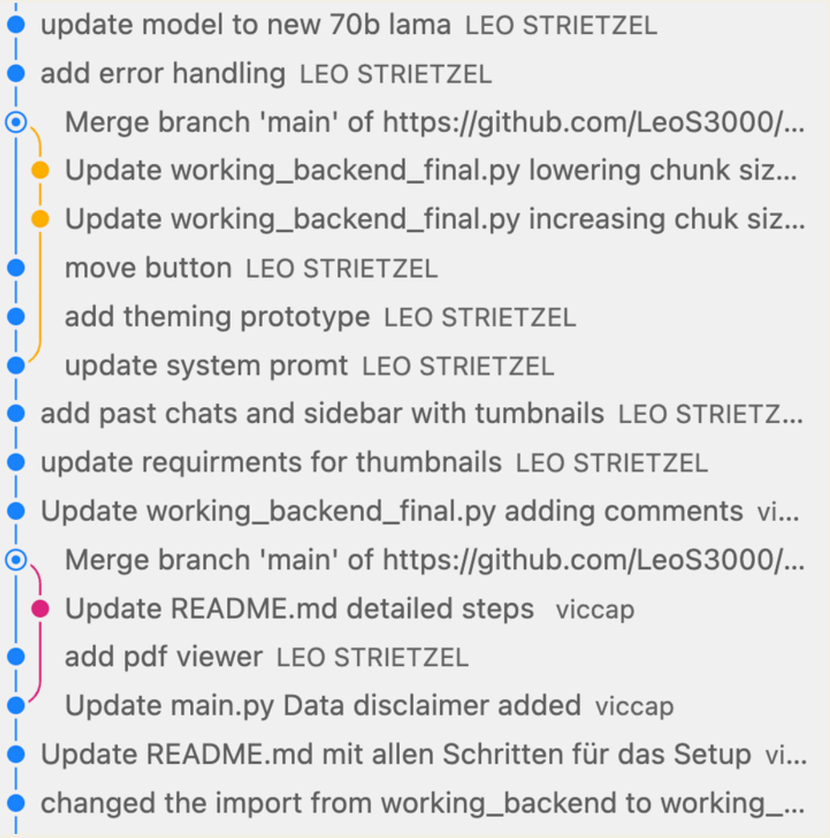


Link to Trelloboard: https://trello.com/b/8bUm8R9D/ai-application-project

Group Work – Git Hub



LeoS3000 fix critial bug and refactor for readiblity	510674d · 2 hours ago	44 Commits
.streamlit	change theme	4 days ago
assets	Added main Python file for streamlit	2 months ago
static	add theming prototype	last week
.dockerignore	add docker image and compose	4 days ago
.gitignore	added faiss index files	3 weeks ago
Dockerfile	add docker image and compose	4 days ago
README.md	update docker	6 hours ago
custom_prompt.py	update custom prompt	6 hours ago
docker-compose.yml	update streamlit command in docker-compose for p...	4 days ago
main.py	fix critial bug and refactor for readiblity	2 hours ago
requirements.txt	add missing requirments	4 days ago
setup_env.sh	Restore project files after re-cloning	last month
working_backend_final.py	enhance PDF processing and retriever configuration...	4 days ago



update model to new 70b lama LEO STRIETZEL
add error handling LEO STRIETZEL
Merge branch 'main' of https://github.com/LeoS3000/...
Update working_backend_final.py lowering chunk siz...
Update working_backend_final.py increasing chuk siz...
move button LEO STRIETZEL
add theming prototype LEO STRIETZEL
update system promt LEO STRIETZEL
add past chats and sidebar with tumbnails LEO STRIETZ...
update requirments for thumbnails LEO STRIETZEL
Update working_backend_final.py adding comments vi...
Merge branch 'main' of https://github.com/LeoS3000/...
Update README.md detailed steps viccap
add pdf viewer LEO STRIETZEL
Update main.py Data disclaimer added viccap
Update README.md mit allen Schritten für das Setup vi...
changed the import from working_backend to working_...

Link to Git Hub: https://github.com/Leo3OOO/RAG_eco_reports/blob/main/working_backend_final.py

Group Work – Git Hub Read me

README

📦 Benutzung

🛠️ Schritte zum Funktionieren

1. **Repository klonen**
   Klone unser Git-Repository in einen gewünschten Ordner und navigiere anschließend in diesen Ordner:

```
bash cd "folder_name"
```

2. **Setup (für Mac oder Linux)** Führe im Terminal (im gewählten Ordner) folgende Befehle aus:

- `chmod +x setup_env.sh`
- `./setup_env.sh`
- `source newenv/bin/activate`

3. **.env-Datei erstellen** Erstelle eine Datei mit dem Namen `.env` und füge den API-Key wie folgt ein:

```
API_KEY='dein Eintrag'
```

Speichere anschließend die Datei.

🚀 Schritte zum Ausführen der Datei (aktuelle Nutzung)

```
1.      Im Terminal zu dem Passenden Ordner navigieren
2.      Folgenden Befehl im Terminal ausführen:
```

```
python working_backend.py
```

🔄 Wiederholte Benutzung

Wenn du das Projekt erneut öffnen möchtest, musst du das Virtual Environment neu aktivieren: 1. Terminal in den passenden Ordner leiten 2. Virtual Environment aktivieren und Code ausführen:

```
source newenv/bin/activate
```
```
python working_backend.py
```

🗂️ Trello

Link to the Trello Board

https://trello.com/b/8bUm8R9D/ai-application-project

Formalities

Student Numbers:
Joschua Levi Prieß (40001214)
Leo Strietzel (4000690)
Margareta Marie Dittrich (4001343)
Mahsa Alaem (4000450)
Victor Caplan (4001191)

Examining Authority:
Dr. Debayan Banerjee

Examination Subject:
AI-based Applications

LEUPHANA
UNIVERSITÄT LÜNEBURG

Universitätsallee 1
21335 Lüneburg
Phone 04131.677-0