



# BRAINSTORM CHATBOT

**A group project for the course:  
AI-based Applications**

**PROJECT DETAILS:**

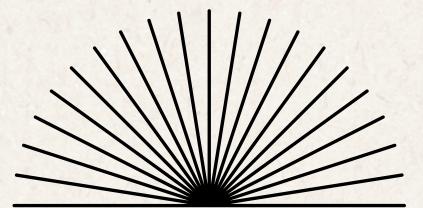
- Trello
- Github

**PRESENTED BY:**

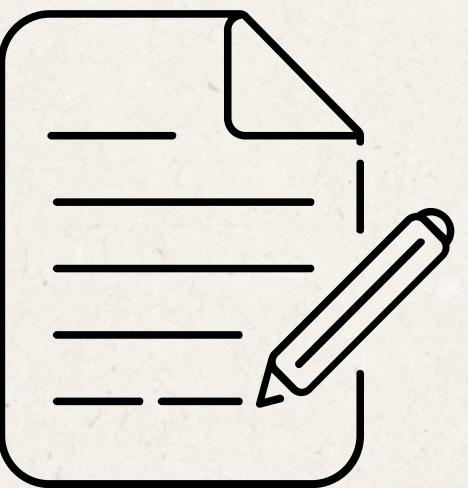
Nadeem Hakimi (4000409)  
Mia Mainka (3047072)  
Marisa Jordan (4000444)  
Daniel Kähm (4001842)

**PRESENTED TO:**

Debayan Banerjee  
Ricardo Usbeck

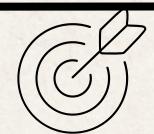


# Table of contents



- 1. Project Overview**
- 2. Procedure**
- 3. Live Demo / Video**
- 4. Frameworks & Technologies**
- 5. System Architecture**
- 6. What worked well**
- 7. Problems**
- 8. Aspects about Speed**
- 9. Contributions**
- 10. Links**
- 11. Q&A**

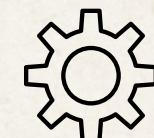
# Project Overview



## 1. Project Goal

web-based application that allows users to:

- Upload a PDF
- Ask questions, receive answers by LLM
- Extract key values



## 2. Core Functionalities Required

- PDF upload interface
- Text extraction, chunking, semantic search
- remote LLM API to generate context-based answers



## 3. Expected Output Formats

- Natural language answers (in chat form)
- Structured JSON with fields like:  
"CO2", "Risks", "Strategy", "Targets", etc.

# Procedure

**1.** Setup

**2.** Zoom-Kickoff

**3.** Task division

**4.** Refining

**5.** Presentation

# 1. Setup

## Trello

A screenshot of a Trello board titled "1. Setup". The board has three main columns: "Backend", "Frontend", and "Team-Aufgaben".

- Backend:**
  - funktionierende Verbindung mit llm
  - api key einlesen
  - in RAG einarbeiten
  - text extraction, in chunks aufteilen
  - chunks in Vektoren verwandeln, in vektorstore speichern
  - Frage an das llm schicken und Kontext hinzufügen
  - JSON
  - Geschwindigkeit verbessern
  - Fragen und Antworten in der gleichen Sprache
- Frontend:**
  - Überschrift und Unterüberschrift einfügen
  - Upload und Chat auf verschiedenen Seiten
  - Vorschau von pdf und c. nebeneinander
  - Design verbessern
  - Buttons wie "Clear Chat", "Start over"
  - PDF nicht bei jeder Frage neu laden
  - Waiting icon während Frage bearbeitet wird
- Team-Aufgaben:**
  - Report erstellen
  - Power-Point für Präsentation erstellen
  - GitHub Repo erstellen
  - Alle Versionen im Repository einfügen
  - SCRUM-Board finalisieren
  - testen, ob irgendwelche Fragen, nicht beantwortet werden
  - Code im Backend kommentieren

At the bottom of each column, there is a "+ Eine Karte hinzufügen" button.

## Git Hub

A screenshot of a GitHub repository named "AI-based-Applications". The repository is public and has 1 branch and 0 tags.

The commit history shows the following activity:

- marisa75 Update backend5.py · af46816 · 2 days ago · 14 Commits
- brAlnstorm final · Update backend5.py · 2 days ago
- old\_versions · organizing · 4 days ago
- Ai-Basted-Applications.mp4 · Add files via upload · last week
- Main (Neu).py · Add files via upload · last month
- main(alt).py · Add files via upload · last month

# Procedure

**1.** Setup

**2.** Zoom-Kickoff

**3.** Task division

**4.** Refining

**5.** Presentation

# Live Demo / Video

The screenshot shows a PyCharm IDE interface with the following details:

- Project:** PythonProjectVideo
- Files:** requirements.txt, backend.py, config.ini
- Content of backend.py:**

```
import re
</
app = Flask(__name__)
app.secret_key = "your-secret-key"

#ordner für uploads und vektoren
UPLOAD_FOLDER = "uploads"
VECTOR_FOLDER = "tmp/faiss_index"
JSON_FOLDER = "jsons"
os.makedirs(JSON_FOLDER, exist_ok=True)
os.makedirs(UPLOAD_FOLDER, exist_ok=True)
os.makedirs(name="tmp", exist_ok=True)

question_cache = {}

#api konfiguration
config = configparser.ConfigParser()
config.read("config.ini")
API_KEY = config["DEFAULT"]["KEY"]
API_URL = config["DEFAULT"]["ENDPOINT"] + "/chat/completions"
MODEL = "meta-llama-3.1-8b-instruct"

#pdf-text extrahieren
def extract_documents_from_pdf(filepath): usage
```

- Run:** backend
- Terminal Output:**

```
* Serving Flask app 'backend'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
C:\Users\danie\PycharmProjects\PythonProjectVideo\backend.py:9: LangChainDeprecationWarning: Importing HuggingFaceEmbeddings from langchain.embeddings is deprecated. Please replace deprecated imports:
>> from langchain.embeddings import HuggingFaceEmbeddings
with new imports of:
```

- Status Bar:** Die Aufnahme wurde begonnen (The recording has begun)

<https://youtu.be/ZTpCEU-y3g8>

AI-Based Applications 10.07.2025

# Choice of Frameworks



## 1. Web Framework

**Flask** → Handles backend routing, sessions, and API communication

**HTML/CSS** – Frontend layout, PDF upload, and chat UI

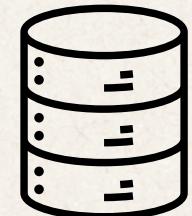
**Localhost** – Local deployment during development



## 2. Document Handling

**PyMuPDF** – Extracts text from PDF files

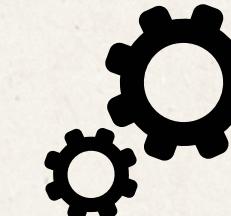
**LangChain** – Handles chunking  
(RecursiveCharacterTextSplitter) and context retrieval  
**HuggingFace MiniLM** – Creates vector embeddings (used  
within LangChain)



## 3. Vector Storage & Embeddings

**HuggingFace MiniLM** – Converts text chunks into vector embeddings for similarity search

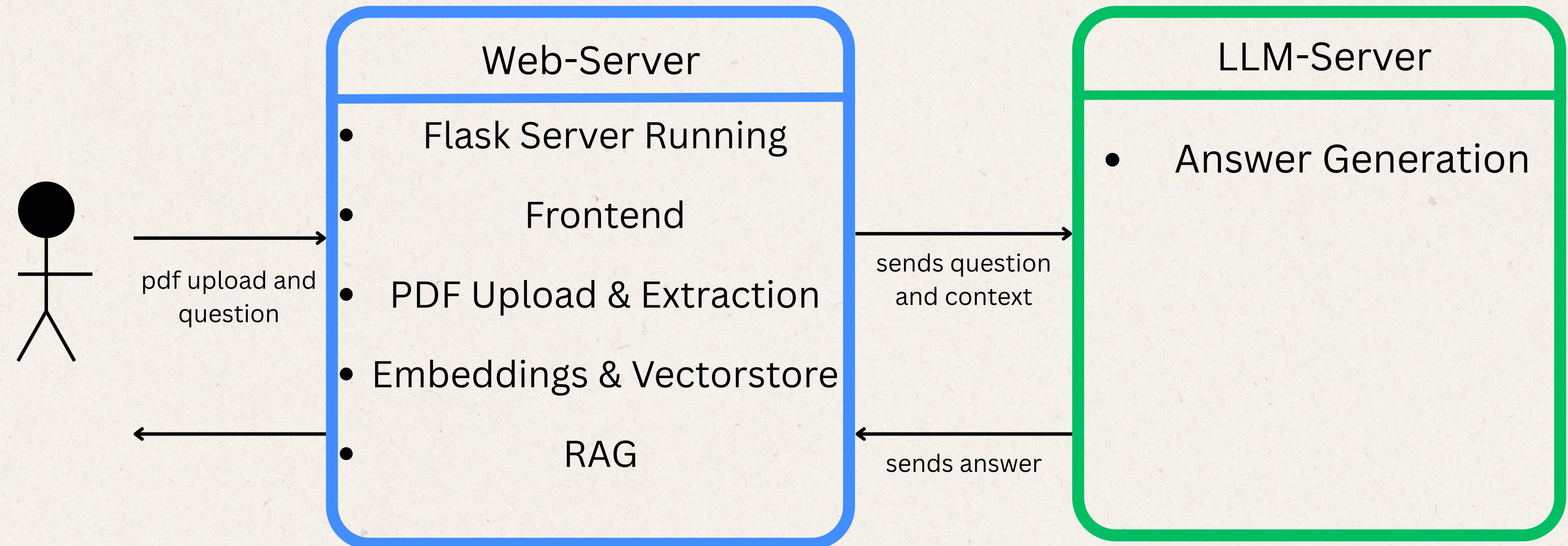
**FAISS** – Stores these vectors and retrieves the most relevant chunks based on user queries



## 4. LLM Integration

**GWDG API (Meta-LLaMA 3.1)** – Processes user queries with context and returns answers

# System Architecture



# **What worked well**

- **Independent division of tasks within the group**
- **Good group dynamics**
- **Integration of all members based on their strengths**
- **Independently working on the project outside of our sessions**

# Problems

- **Issues with understanding GitHub**
- **Errors occurred when transferring data with Python**
- **Overall issues with the code**
- **Adjustment of speed**

# Aspects about Speed

```
# Transform to chunks
splitter = RecursiveCharacterTextSplitter(chunk_size=1200, chunk_overlap=20)
documents = splitter.create_documents([full_text])
```

- **Differences of Chunk-Sizes**
- **Answers are saved, and if the same question is asked again, saved answer is sent back**
- **General structure of backend**

# **Contributions**

**Backend & LLM Integration - Nadeem, Marisa**

**Frontend & UI Design - Everyone**

**Prompt & JSON Output - Everyone**

**SCRUM & GitHub - Everyone**

**Presentation - Mia, Daniel**

**Demo & Recording - Daniel**

**Most areas were developed collaboratively, with individual members focusing more deeply on specific topics.**

# Links

## Trello

<https://trello.com/invite/b/681dd6ea555321118bfa12c2/ATTIe4da5f5b879ec56af2724870310de71a17EADBF2/ai>

## Git Hub Repository

<https://github.com/DanielLeuphana/AI-based-Applications>

# Questions?



# **Thank you for your attention!**

**PRESENTED BY:**

Nadeem Hakimi (4000409)  
Mia Mainka (3047072)  
Marisa Jordan (4000444)  
Daniel Kähm (4001842)

**PRESENTED TO:**

Debayan Banerjee  
Ricardo Usbeck

# Page Number

- extract page number from question
- Only load first 50 pages
- If user asks specifically for a page number, those pages get loaded as well

```
#findet Seitenzahl in der gestellten Frage heraus
def extract_page_number_from_question(question): 1 usage
    match = re.search(pattern: r"(Seite|page)\s*(\d+)", question, re.IGNORECASE)
    if match:
        return int(match.group(2))
    return None
```

## Outcome

- fast extraction (because only 50 pages)
- Within these first 50 pages LLM could estimate the pages, so questions like “What is on page 40?” were possible
- If user doesn’t mention the page number and asks a context-based question about later pages, LLM can’t answer