CAPSTONE PROJECT

Faculty Digital Profile Builder (RAG-Based)

Presented By:
Name - Anjali Niranjan
College - Delhi Technical Campus, Greater Noida (Affiliated to Guru Gobind Singh Indraprastha University, Delhi)
Department - B.tech [Computer Science & Engineering]

edunet foundation

OUTLINE

- Problem Statement
- Proposed System/Solution
- System Development Approach
- Algorithm & Deployment
- Result (Output Image)
- Conclusion
- Future Scope
- References

Problem Statement

- Faculty members in colleges and universities are often asked to maintain detailed academic profiles. These profiles are required for various purposes like accreditation processes (such as NAAC or NBA), applying for promotions, academic collaborations, and maintaining visibility on institutional websites.
- Usually, these profiles include multiple details like qualifications, research publications, patents, FDPs attended, certifications, and achievements. Gathering all this information from different documents like CVs, certificates, and spreadsheets can be a time-consuming and tiring task.
- In many colleges, this work is still done manually using tools like Word or Excel, which can lead to inconsistency and human errors. Also, there is no simple and smart system that helps in directly extracting structured profile data from documents like resumes, research papers, or scanned certificates.
- This creates a need for a simple digital tool that can help faculty generate and manage their profiles more easily and consistently.

Proposed Solution

This project aims to create a simple and smart digital tool that helps faculty members automatically generate their academic profiles using documents they already have, like their PDF resume (CV).

Often, these resumes contain almost all the required information — such as the faculty member's name, email, phone number, educational qualifications, internships, work experience, technical skills, and more. However, collecting and formatting this information manually for official profiles takes time and effort.

With this system, the user (faculty member) simply uploads their resume file into the app. The app reads the file, sends its content to an AI model (IBM Granite), and the model understands and extracts important information from the text.

The extracted information is then displayed in the app in a structured format, like sections:

- Name and contact details
- Education history
- Work experience or internships
- Research interests or publications
- Skills and tools

System Approach

System Requirements:

To build and run the Faculty Digital Profile Builder (RAG-Based) system, we needed some tools and software. These tools helped us create the app, connect it to IBM's AI service, and run it smoothly on our computer.

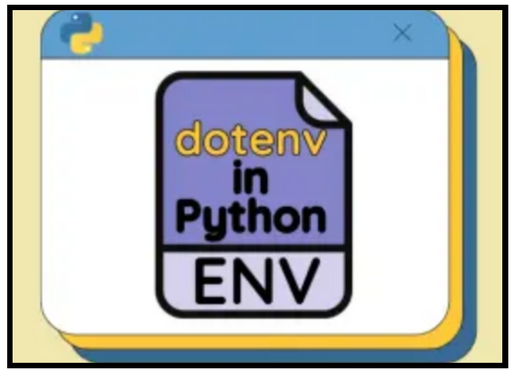
To build and run this project, the following software and environment setup was required:

- Python 3.10 – Programming language used to develop the backend logic.
- Streamlit – To build a simple and interactive web interface.
- IBM Cloud Account (Lite Plan) – For accessing IBM Granite model.
- Virtual Environment – For managing dependencies locally.
- Modern Web Browser – To view and interact with the app.

Libraries and Tools Used:

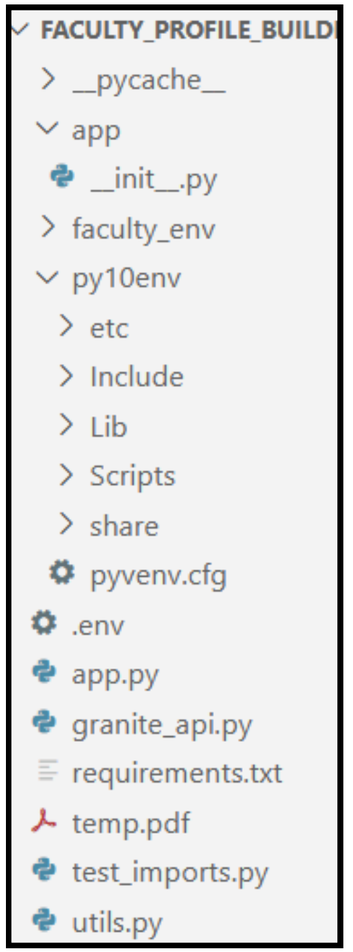
Here are the main Python libraries used to develop this system:

- streamlit – For creating the web app UI.
- ibm-watson-machine-learning – To access IBM Granite AI model and send prompts.
- python-dotenv – To securely load API keys and project credentials from a .env file.
- PyMuPDF (fitz) – To extract text from uploaded PDF resumes.
- os – To manage environment variables.
- json – To parse and handle structured JSON output from the AI model.



Folder Structure

```
faculty_profile_builder/
│
├── app.py                  # Main Streamlit application
├── utils.py                # PDF text extraction logic
├── granite_api.py          # IBM Granite API call and processing logic
├── .env                    # API keys and credentials (not shared)
├── requirements.txt        # List of Python packages used
│
├── faculty_env/            # Virtual environment folder
│   └── ...                 # (Contains local dependencies)
│
└── sample_docs/            # Folder for sample PDF resumes
    └── resume.pdf
```



Screenshot Of Folder Structure

Algorithm & Deployment

How the system works:

1. Faculty uploads a PDF resume (CV).
2. The PDF is read and text is extracted using PyMuPDF.
3. A prompt is created using this content and sent to the IBM Granite model.
4. Granite tries to generate a structured response with the faculty's information.
5. This response is parsed and shown in the Streamlit app as profile sections

Deployment Info:

- The app runs locally using streamlit run app.py.
- API keys and credentials are stored securely using .env file.
- IBM Cloud Lite was used to access Granite model.

Code Snippets

app.py

```python
import streamlit as st
from utils import extract_text_from_pdf
from granite_api import generate_profile

st.set_page_config(page_title="Faculty Profile Builder", layout="wide")
st.title("📄 Faculty Profile Builder (RAG + Granite)")

uploaded_file = st.file_uploader("Upload Resume (PDF)", type="pdf")

if uploaded_file:
    text = extract_text_from_pdf(uploaded_file)
    if st.button("🚀 Generate Faculty Profile"):
        with st.spinner("Generating profile using IBM Granite..."):
            try:
                profile = generate_profile(text)
                st.success("✅ Profile Generated")
                for key, value in profile.items():
                    st.subheader(key.capitalize())
                    st.write(value)
            except Exception as e:
                st.error(f"❌ Failed to generate profile: {e}")

```

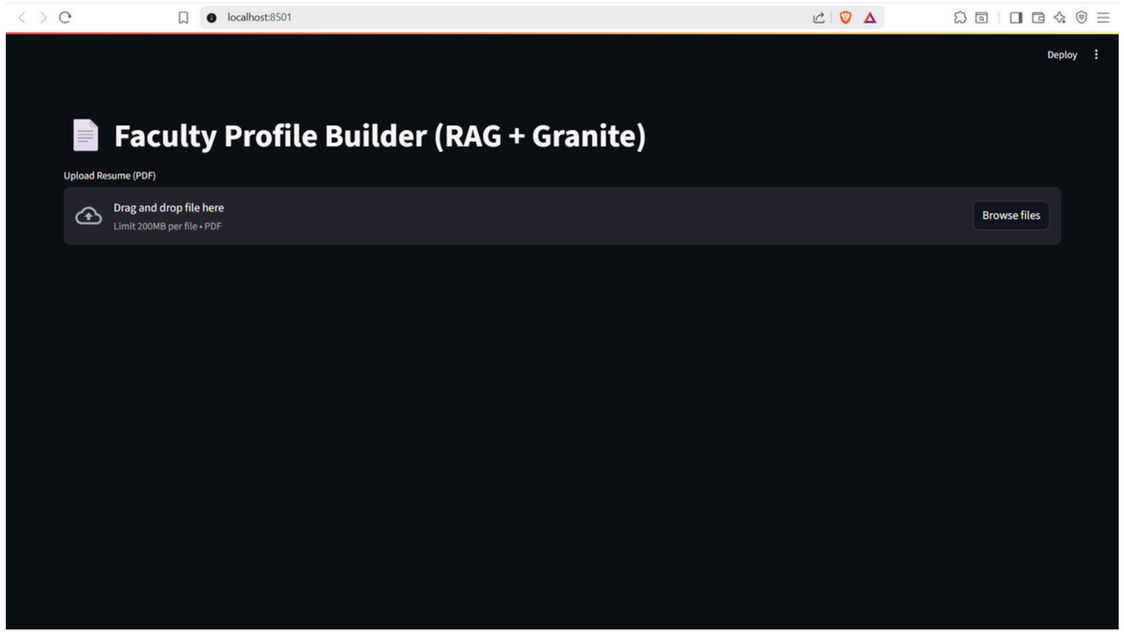
app.py

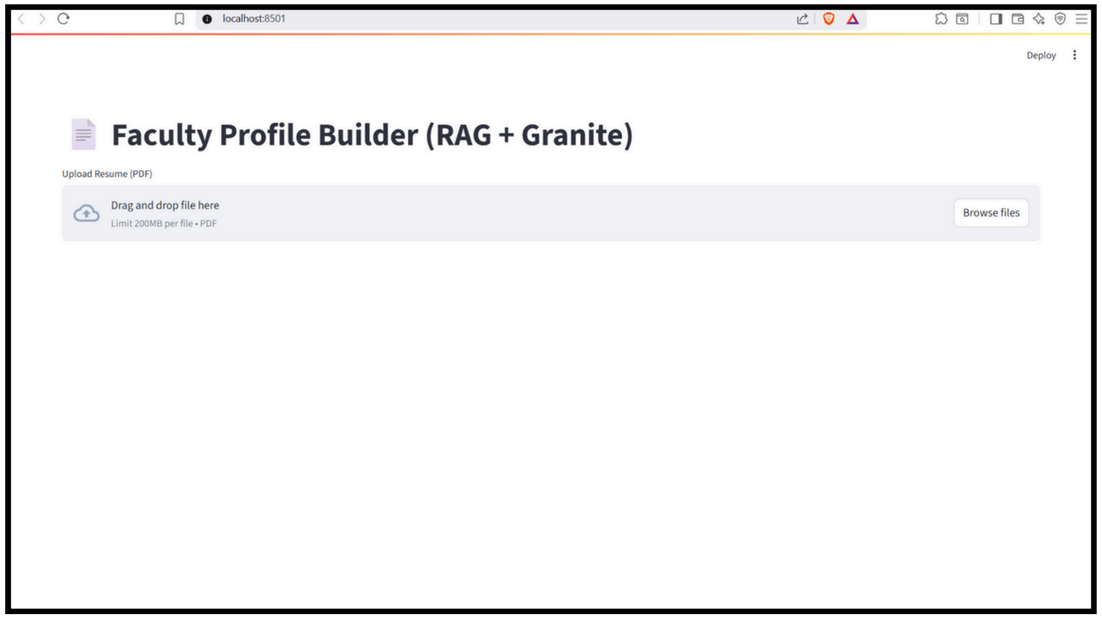
granite_api.py

```python
from ibm_watson_machine_learning.foundation_models import Model
from dotenv import load_dotenv
import os
import json
import re

# Load environment variables
load_dotenv()

API_KEY = os.getenv("API_KEY")
PROJECT_ID = os.getenv("PROJECT_ID")
URL = os.getenv("URL")

print("DEBUG: API_KEY =", API_KEY[:6] + "..." if API_KEY else None)
print("DEBUG: PROJECT_ID =", PROJECT_ID)
print("DEBUG: URL =", URL)

if not API_KEY or not PROJECT_ID or not URL:
    raise Exception("❌ API_KEY, PROJECT_ID, or URL not found. Check .env file.")

model = Model(
    model_id="ibm/granite-3-3-8b-instruct",
    credentials={"apikey": API_KEY, "url": URL},
    project_id=PROJECT_ID,
)

def generate_profile(text):
    prompt = f"""Extract a structured JSON faculty profile from the following resume text:
{text}
The JSON must have these keys: name, email, phone, education, experience, research_interests, publications, skills.
Ensure the JSON is valid and complete.
    """
    params = {"decoding_method": "greedy", "max_new_tokens": 800}
    response = model.generate(prompt=prompt, params=params)

    raw_output = response.get("results", [{}])[0].get("generated_text", "")
    print("\n🔍 Raw model output:\n", raw_output)

    # Try to extract the first valid JSON object using regex
    match = re.search(r"\{[\s\S]+\}", raw_output)
    if not match:
        raise ValueError("❌ No JSON object found in model response.")

    json_text = match.group(0)
```

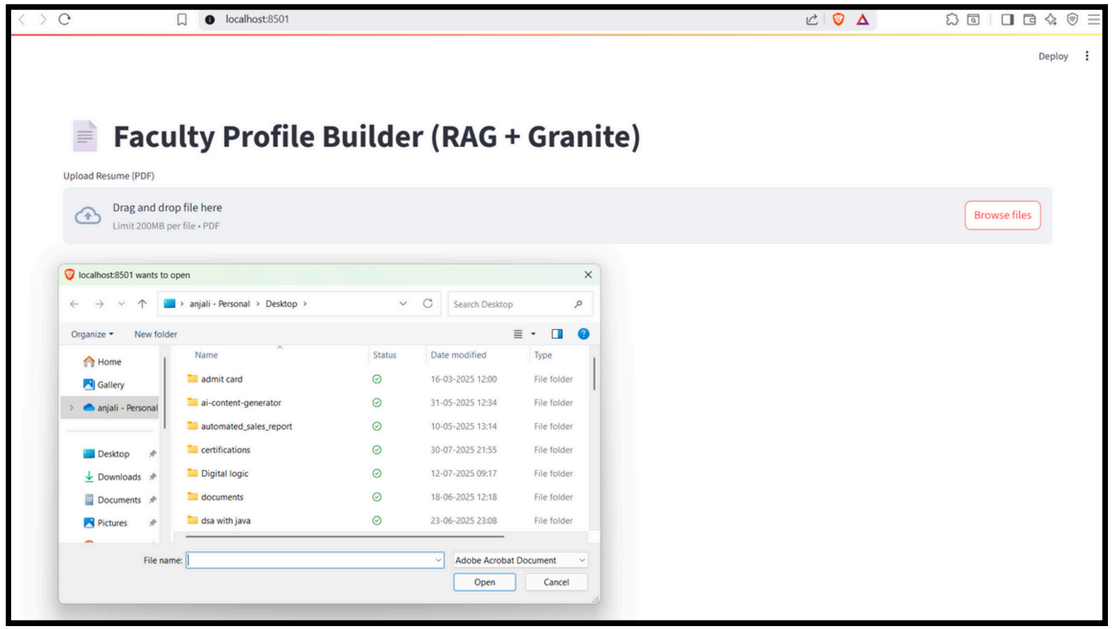
granite_api.py

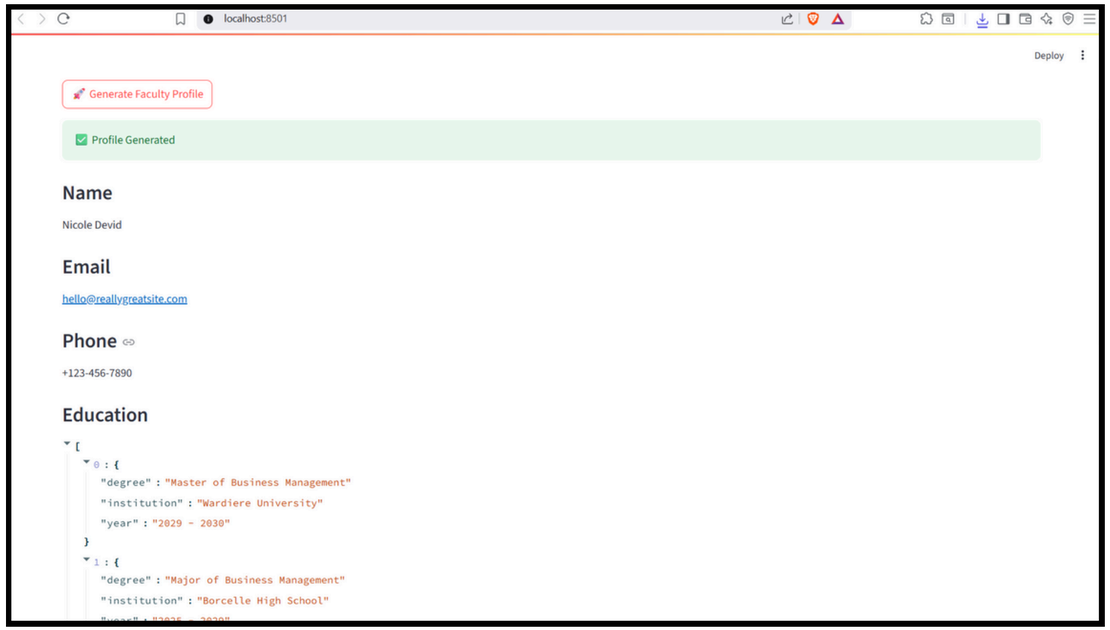
Result



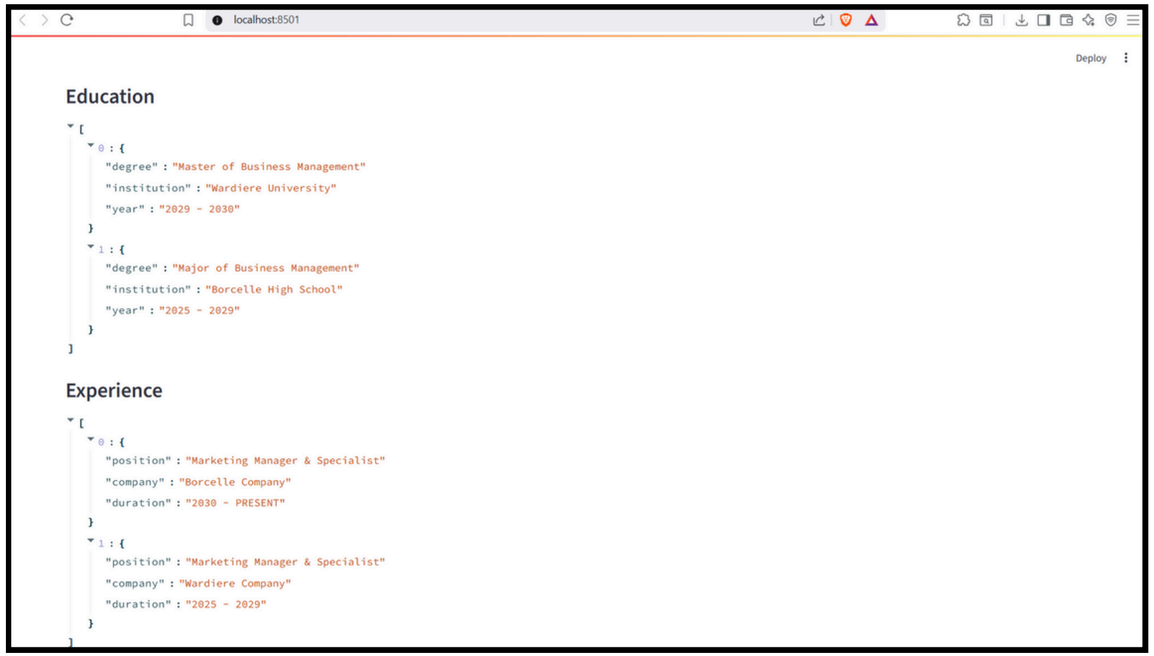


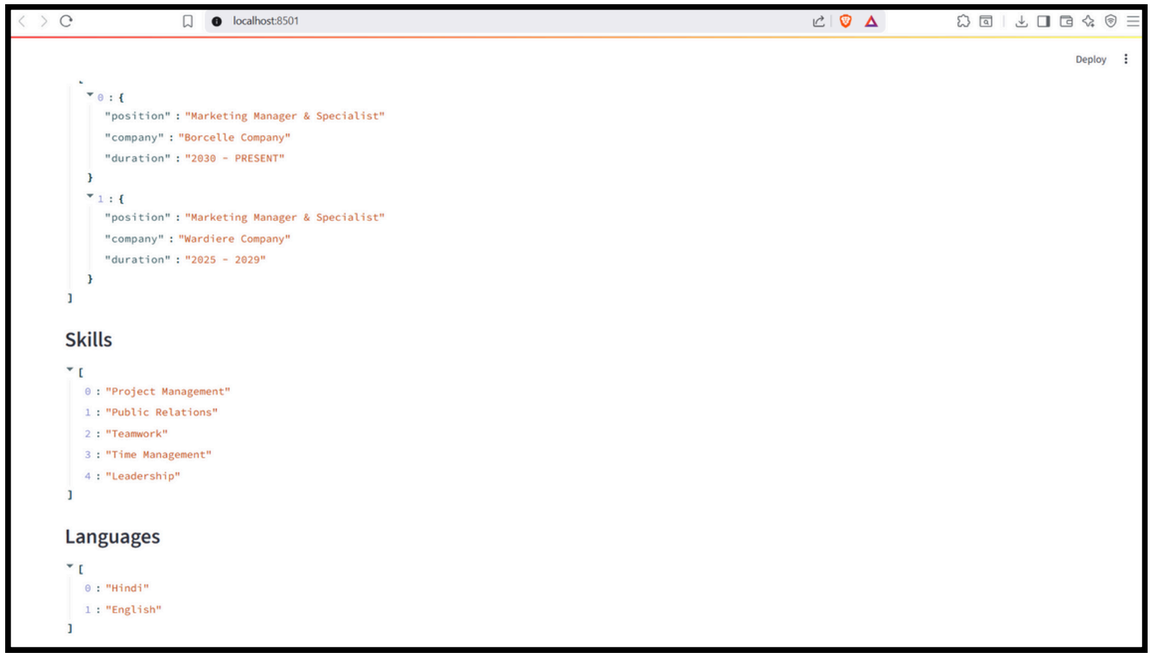
Result





Result

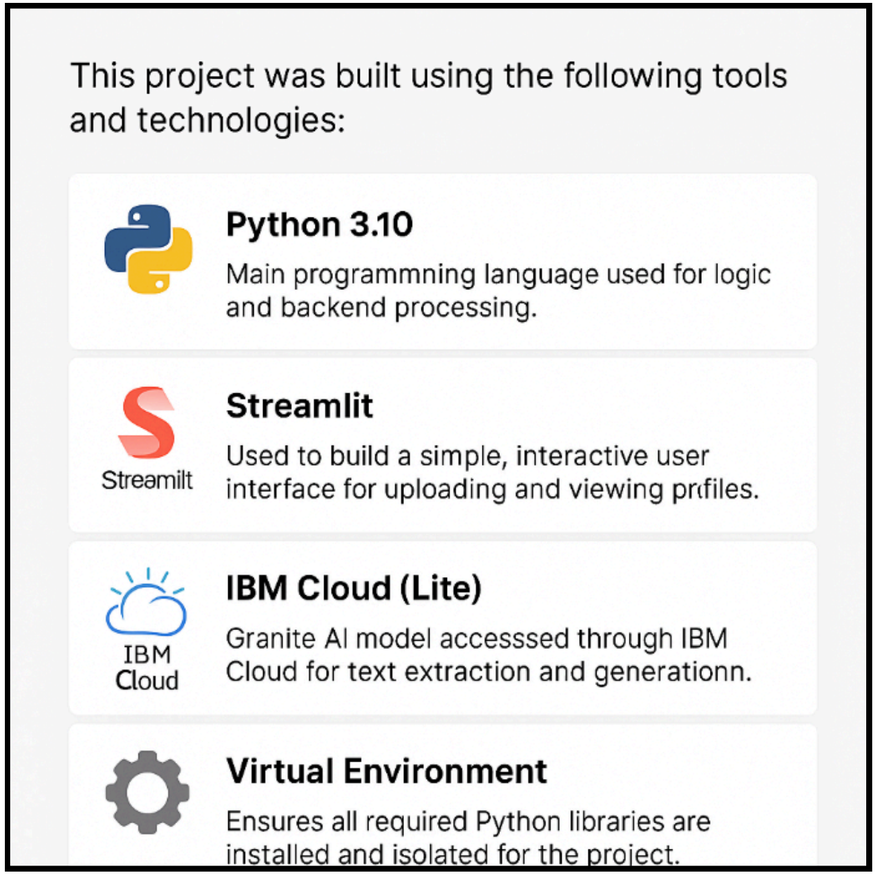




Conclusion

Traditionally, creating a faculty profile means spending hours manually writing and formatting details from various documents like resumes, certificates, or research papers. This process can be slow, repetitive, and prone to human errors.
With our proposed system, the process becomes fast, easy, and smart.

- It uses AI technology to read and understand documents just like a human would.
- It automatically extracts useful details such as education, experience, and skills.
- It saves valuable time and effort, while ensuring clean, standardized, and professional profiles.



Future Scope

This is a very basic system, built as a starting point to explore how AI can help in creating digital faculty profiles. While it works, it has several limitations and pitfalls that can be improved in the future.

Areas of Improvement:

1. **Accuracy of Extraction**: The current system may not always extract data perfectly, especially if the document format is complex or inconsistent. Future improvements can include better preprocessing and fine-tuning of AI models.
2. **Speed and Efficiency:** The system can be optimized to work faster and handle larger files more smoothly. This includes reducing loading time and improving the performance of document parsing.
3. **Multiple File Support**: Right now, it mainly works with simple PDF resumes. In the future, it can be extended to support certificates, research papers, spreadsheets, and scanned documents.

Future Scope

5. **Profile Editor & Export:** Future versions can include options to edit the extracted profile directly in the app and export it as a formatted Word or PDF document.

6**. User Authentication:** A login system can be added for faculty members to securely upload and manage their profiles online.

References

- IBM Watson Machine Learning: https://www.ibm.com/cloud/watson-machine-learning
- IBM Granite Foundation Models: https://dataplatform.cloud.ibm.com
- Streamlit Documentation: https://docs.streamlit.io
- PyMuPDF: https://pymupdf.readthedocs.io

IBM Certifications

Getting Started with Artificial Intelligence Certificate

edunet foundation

IBM Certifications

Journey to Cloud Certificate

IBM Certifications

Lab: Retrieval Augumented Generation with LangChain Certificate

THANK YOU

edunet
foundation