Title: AI Document Analyzer and Keyword Extractor

College Name: BMS Institute of Technology and Management

Team Members:

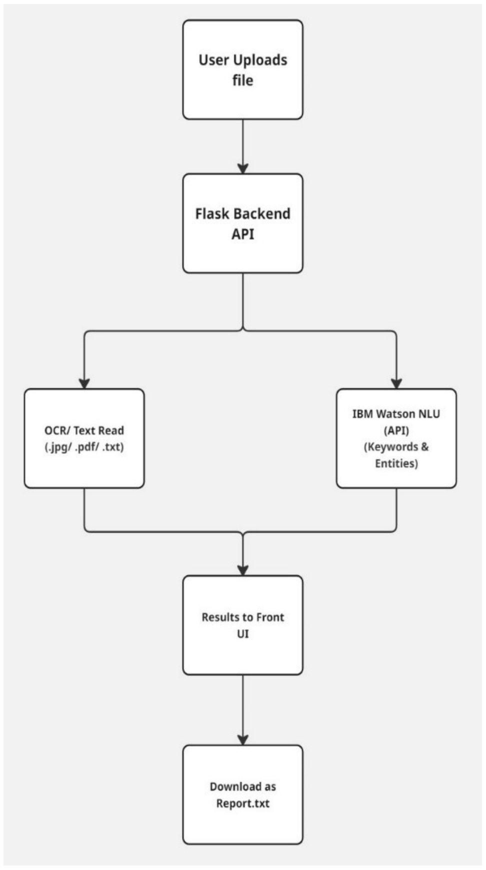
1. Priyanshu Sharma - CAN_35606461
2. Satish Mallappa B Patil – CAN_35717314
3. Kishor C – CAN_35990034
4. Albin Akkara – CAN_35608542

Blueprint of the Project:

The AI Document Analyzer application is structured into three main layers:

1. **Frontend (User Interface):**
    - Allows user to upload .pdf, .txt, .jpg, .png files.
    - Displays extracted text, keywords, and entities.
    - Provides download option for analyzed result.
2. **Backend (Flask Application):**
    - Receives file input.
    - Performs OCR/text extraction.
    - Sends extracted text to IBM Watson NLU.
    - Formats and sends response to the frontend.
3. **Cloud Services (IBM Cloud):**
    - IBM Watson NLU: NLP services for keyword/entity extraction.

Flow Diagram of Plan:


User Uploads file
Flask Backend API
OCR/ Text Read (.jpg/ .pdf/ .txt)
IBM Watson NLU (API) (Keywords & Entities)
Results to Front UI
Download as Report.txt

Services Used:

Service/Tool	Why It's Used
IBM Watson NLU	Extracts structured data (keywords/entities) from unstructured text.
pytesseract (OCR)	Converts scanned images into readable text.
PyMuPDF (fitz)	Extracts text from PDF documents.
Flask	Lightweight Python backend to manage API requests and file uploads.

Step by Step Execution Process:

Step 1 – Environment Setup

- Install Python and required libraries:

➔ pip install flask pytesseract Pillow PyMuPDF ibm-watson

- Install Tesseract OCR

Step 2 – Configure IBM Watson NLU

- Create an IBM Cloud account.
- Launch Watson NLU service.
- Get **API Key** and **Service URL**.
- Use these credentials in app.py to authenticate.

Step 3 – Backend Development (Flask)

- Create app.py:
  - Handles file uploads
  - Extracts text
  - Sends text to Watson NLU
  - Returns JSON with keywords and entities

Step 4 – Frontend Development

- Create templates/index.html:
  - Upload input form
  - JavaScript to send file to /analyze
  - Display response: extracted text, keywords, entities
  - Button to download result from /download-result

Step 5 – Download Feature

- In app.py, save results to a file (static/result.txt)

- Create a Flask route /download-result to download the file

Step 6 – Test the App

- Run using: python app.py
- Go to http://127.0.0.1:5000/
- Upload file, view results, download report

Future Enhancements:

- Use **IBM Object Storage** for storing file history.
- Use **Watson Studio + ML** to train a classification model (e.g., legal vs. medical document).
- Add sentiment/emotion analysis.