**Multi-Source Data Analytics Chatbot**

This project is a multi-source data analytics chatbot application built using Flask and various Python libraries. The chatbot is capable of handling general questions using Wikipedia, a fine-tuned GPT-2 model, and can analyze various document formats, create visualizations, perform sentiment analysis, summarization, and more.

**Features**

* Generate text responses using a fine-tuned GPT-2 model.
* Analyze and extract information from various document formats (PDF, Excel, CSV, JSON, XML, DOCX, images, DICOM, and TXT).
* Perform web scraping and extract relevant data.
* Convert speech to text and text to speech.
* Handle fact-based and general knowledge questions using Wikipedia.
* Generate visualizations such as histograms, bar graphs, pie charts, and word clouds.
* Perform sentiment analysis and text summarization.

**Project Structure**

* app.py: Main application file containing the Flask routes and core logic.
* services/document\_service.py: Service to handle document reading and analysis.
* services/scraping\_service.py: Service to perform web scraping and content analysis.
* services/voice\_service.py: Service to handle speech-to-text and text-to-speech conversion.
* services/search\_service.py: Service for generating text using GPT-2.
* services/text\_service.py: Service for answering questions using GPT-2 and a question-answering pipeline.
* templates/index.html: HTML template for the web interface.
* static/: Directory containing static files such as CSS, JavaScript, and images.

**Installation**

To install the necessary dependencies, run the following command:

pip install -r requirements.txt

Ensure you have Python 3.6 or higher installed on your machine.

**Usage**

1. **Run the Flask Application**:

python app.py

1. **Access the Web Interface**:

Open your web browser and navigate to http://127.0.0.1:5000.

1. **Interact with the Chatbot**:
   * Type or speak your questions and commands in the provided input fields.
   * Upload documents to analyze their content.
   * Scrape websites to extract and analyze data.
   * View responses and analysis results directly on the web interface.

**API Endpoints**

* **Home**: / - Returns the home page.
* **Generate Text**: /generate\_text (POST) - Generates text based on a given prompt.
* **Analyze Image**: /analyze\_image (POST) - Analyzes an uploaded image.
* **Voice Command**: /voice\_command (POST) - Processes a voice command.
* **Read Document**: /read\_document (POST) - Reads and analyzes an uploaded document.
* **Scrape Website**: /scrape (POST) - Scrapes a website for data.
* **Ask Question**: /ask\_question (POST) - Asks a question based on the current context.
* **Fact Question**: /fact\_question (POST) - Asks a fact-based question.
* **Feedback**: /feedback (POST) - Submits feedback on the chatbot's response.
* **Ask Question Audio**: /ask\_question\_audio (POST) - Asks a question based on an audio file.

**Services**

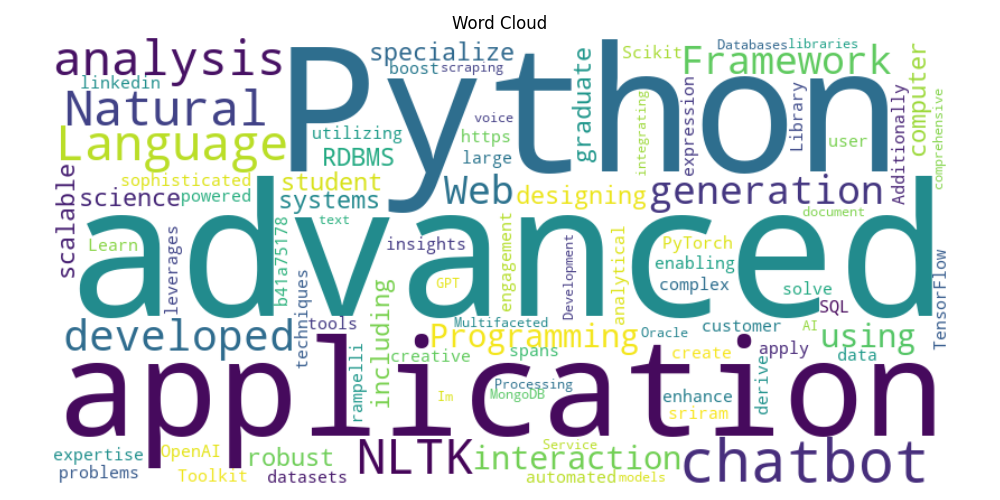
**Document Service**

**File**: services/document\_service.py

Handles reading and analyzing various document formats including PDF, Excel, CSV, JSON, XML, DOCX, images, DICOM, and TXT files. Provides functionalities for text extraction, sentiment analysis, summarization, word cloud generation, and data visualization.

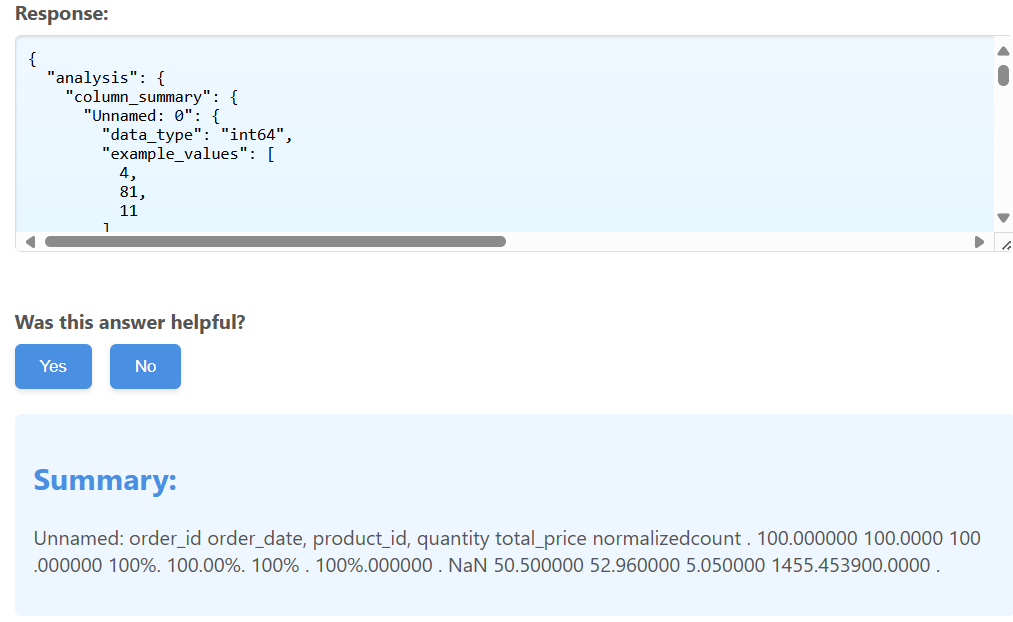
Example For document/pdf

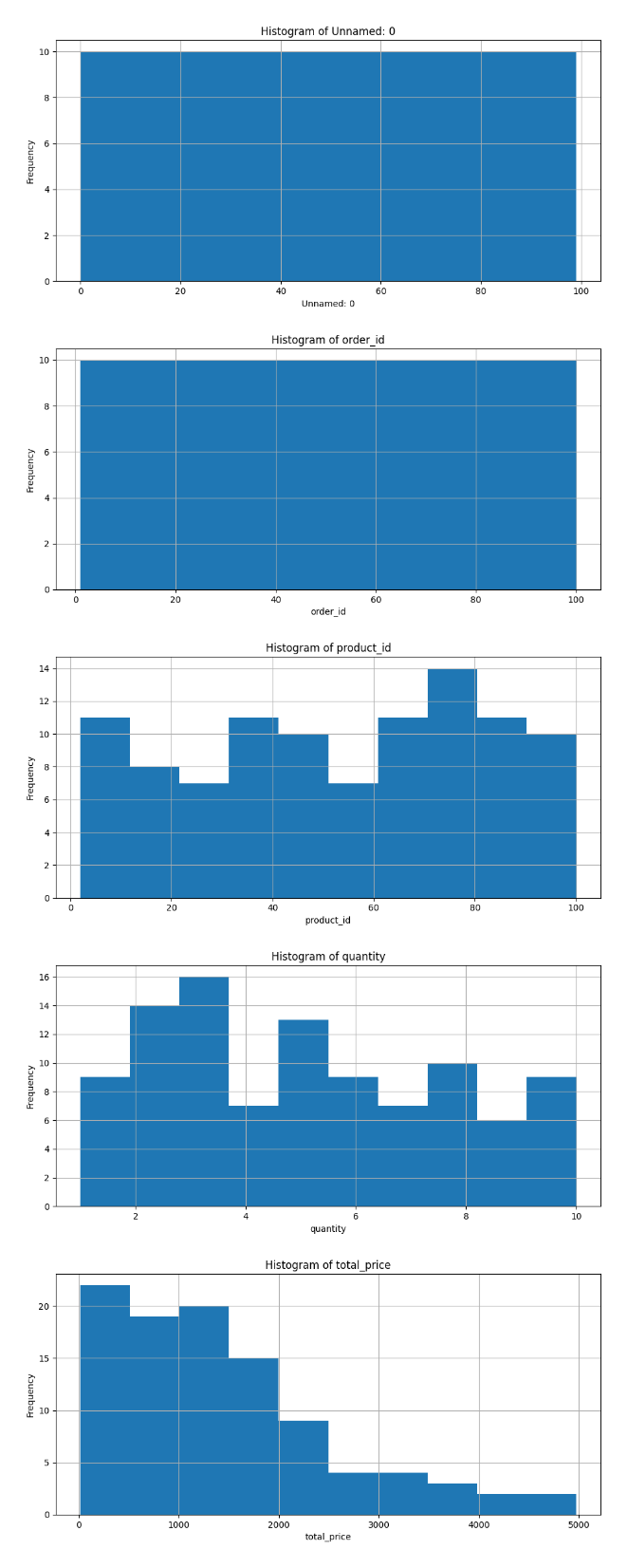
I have uploaded my resume, here are the results



Example for data frames

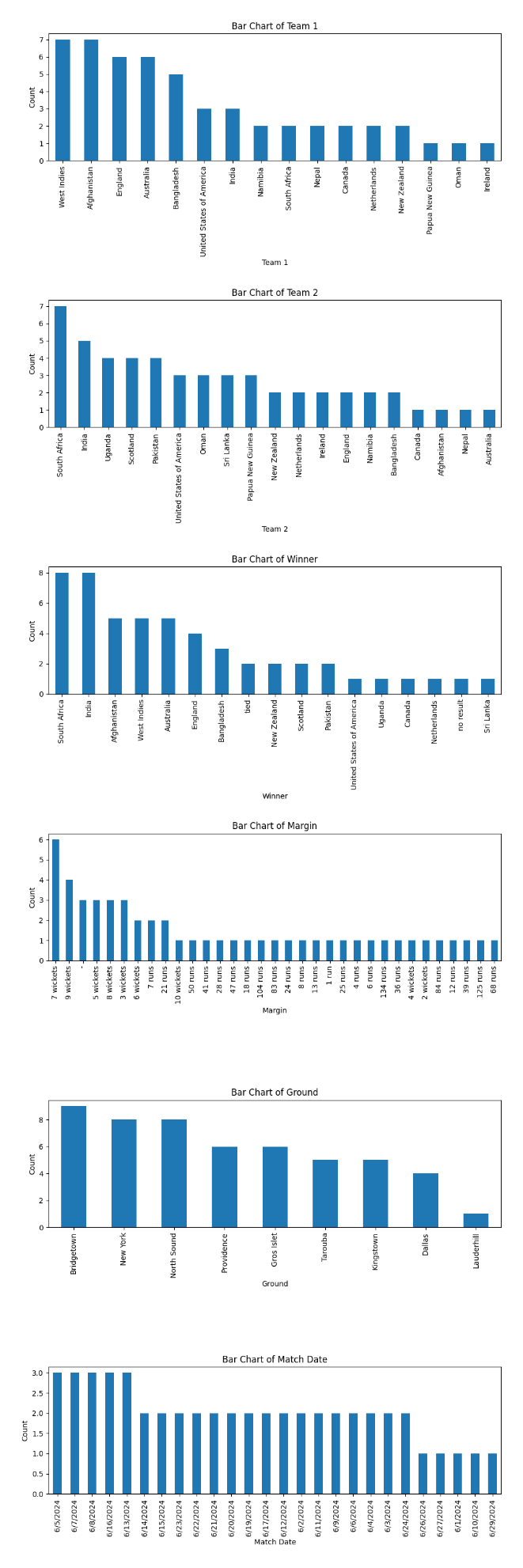
* + 1. With numerical coloums

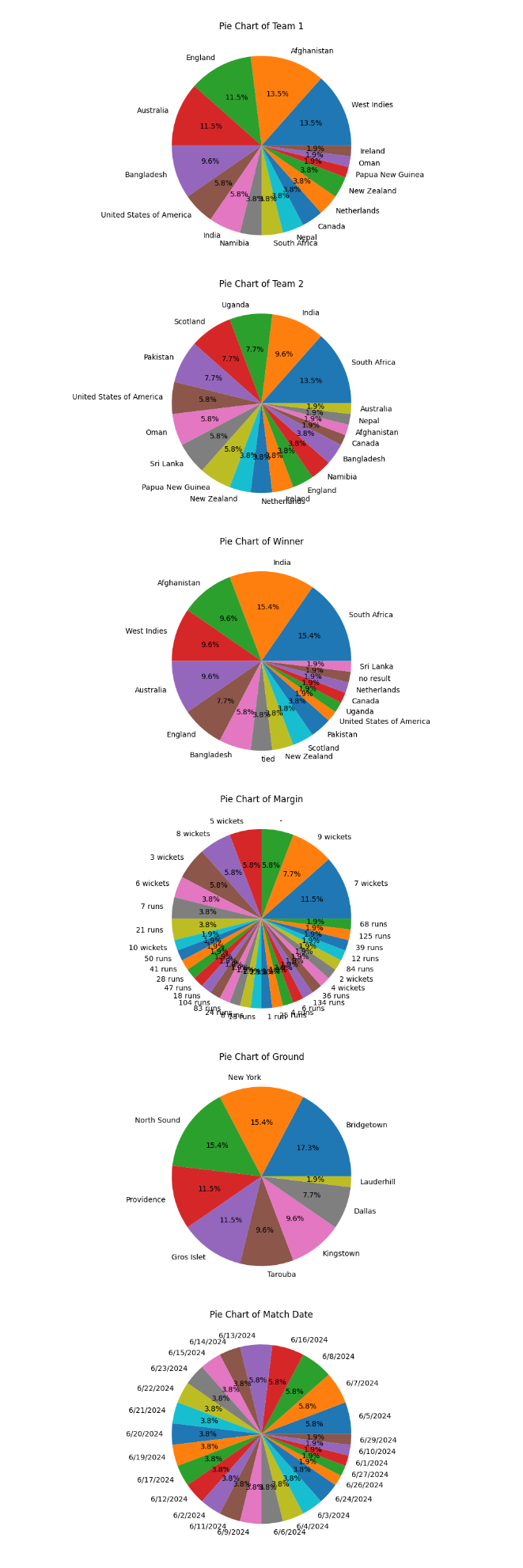




* + 1. For categorical coloums

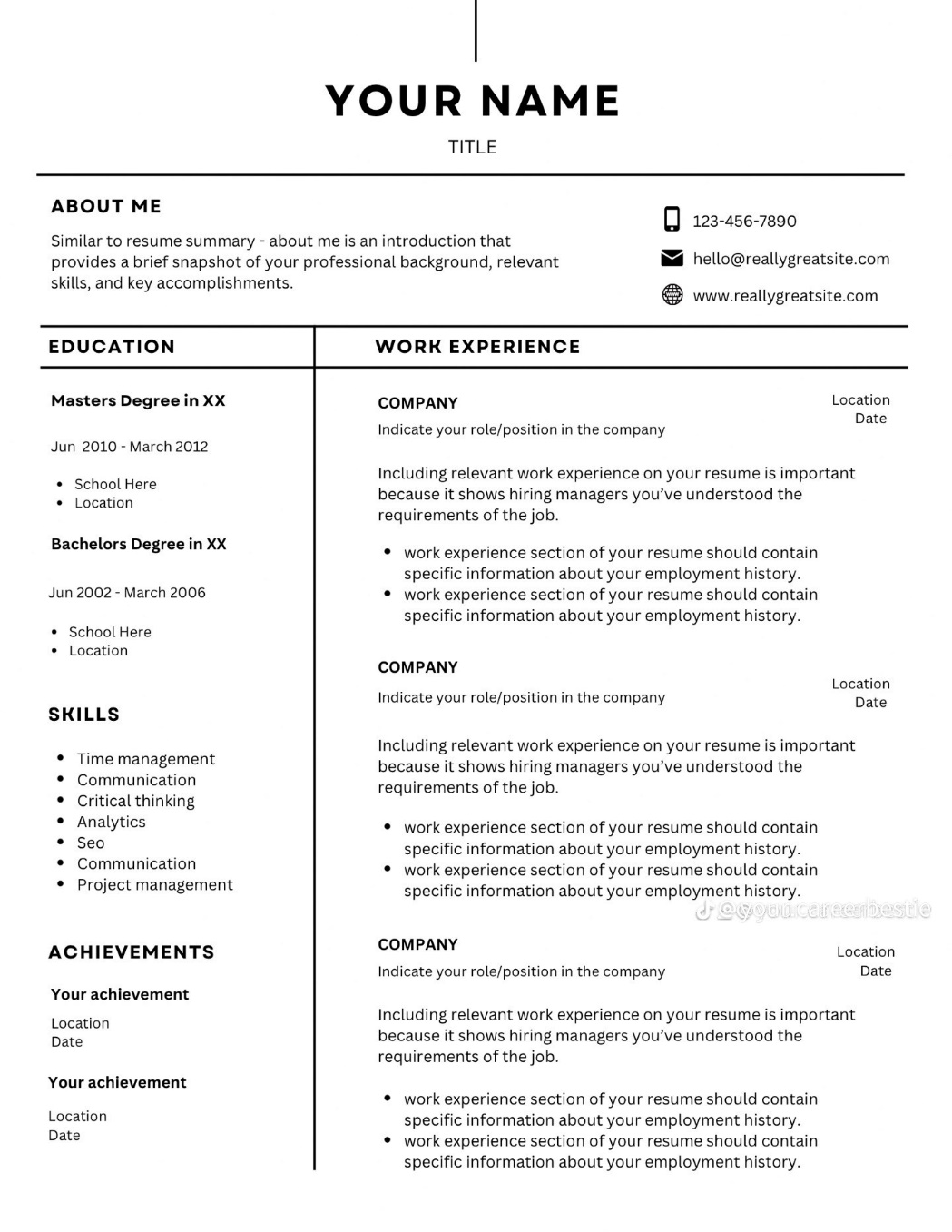


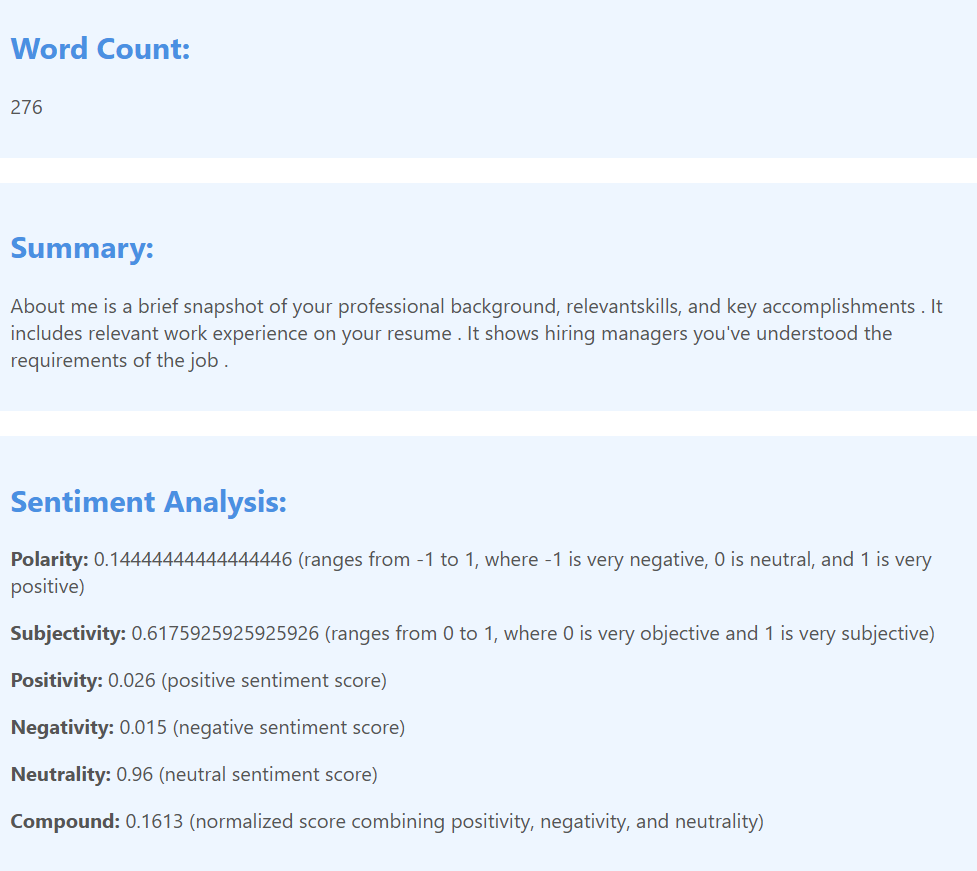




Example for images

Image uploaded:







Example for DCM(medical reports)

Response in json  
  
"metadata": {

"Accession Number": "",

"Acquisition Date": "",

"Acquisition Number": "2",

"Acquisition Time": "",

"Bits Allocated": "16",

"Bits Stored": "16",

"Body Part Examined": "CHEST",

"Columns": "512",

"Content Date": "",

"Content Time": "",

"Convolution Kernel": "SB",

"Data Collection Diameter": "500",

"Distance Source to Detector": "1040",

"Distance Source to Patient": "570",

"Exposure": "61",

"Exposure Time": "9528",

"Filter Type": "SB",

"Frame of Reference UID": "1.2.156.14702.1.1000.16.3.20200311113603953",

"Gantry/Detector Tilt": "0",

"High Bit": "15",

"Image Orientation (Patient)": "[1, 0, 0, 0, 1, 0]",

"Image Position (Patient)": "[-192, -211.5, -722.700012]",

"Image Type": "['ORIGINAL', 'PRIMARY', 'AXIAL', 'HELICAL']",

"Instance Creation Date": "20200311",

"Instance Creation Time": "113912.234",

"Instance Number": "122",

"Institution Address": "",

"Institution Name": "",

"KVP": "120",

"Largest Image Pixel Value": "3196",

"Manufacturer": "",

"Manufacturer's Model Name": "",

"Modality": "CT",

"Operators' Name": "",

"Patient Comments": "",

"Patient ID": "98.12.21",

"Patient Position": "HFS",

"Patient's Age": "",

"Patient's Birth Date": "",

"Patient's Name": "",

"Patient's Sex": "",

"Patient's Size": "0",

"Patient's Weight": "0",

"Photometric Interpretation": "MONOCHROME2",

"Pixel Representation": "0",

"Pixel Spacing": "[0.826172, 0.826172]",

"Position Reference Indicator": "",

"Protocol Name": "Chest Helcal",

"Reconstruction Diameter": "423",

"Referring Physician's Name": "",

"Rescale Intercept": "-1024",

"Rescale Slope": "1",

"Rotation Direction": "CW",

"Rows": "512",

"SOP Class UID": "1.2.840.10008.5.1.4.1.1.2",

"SOP Instance UID": "1.3.6.1.4.1.9590.100.1.2.35425483210115498342516842690793149184",

"Samples per Pixel": "1",

"Scan Options": "HELICAL",

"Series Description": "Lung 1.5",

"Series Instance UID": "1.2.156.14702.1.1000.16.1.2020031111365293700020003",

"Series Number": "4",

"Slice Location": "722.7",

"Slice Thickness": "1.5",

"Smallest Image Pixel Value": "0",

"Software Versions": "",

"Spatial Resolution": "0.33",

"Specific Character Set": "ISO\_IR 100",

"Station Name": "",

"Study Date": "",

"Study Description": "LUNG",

"Study ID": "93725",

"Study Instance UID": "1.2.156.14702.1.1000.16.0.20200311113603875",

"Study Time": "",

"Table Height": "377.3",

"Window Center": "-500",

"Window Width": "1400",

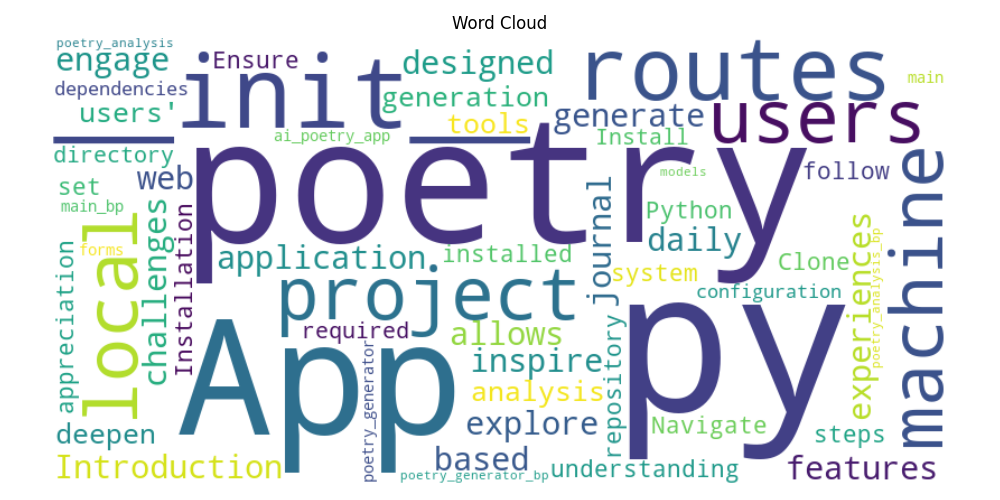
"X-Ray Tube Current": "81"

},

"summary": " Image Type: ['ORIGINAL', 'PRIMARY', 'AXIAL', 'HELICAL', 'ORDER', 'PATENT', 'PEAT', 'PLATENT' Image Image Image: [1, 0, 0.0.1.4.5.1] Image Image Size: \"Hestest Image Pixel Value: 3196\" Image Pixel Size: \"CHEST,\" \"HEST\", \"PEAT\" Image Type Type:"

}

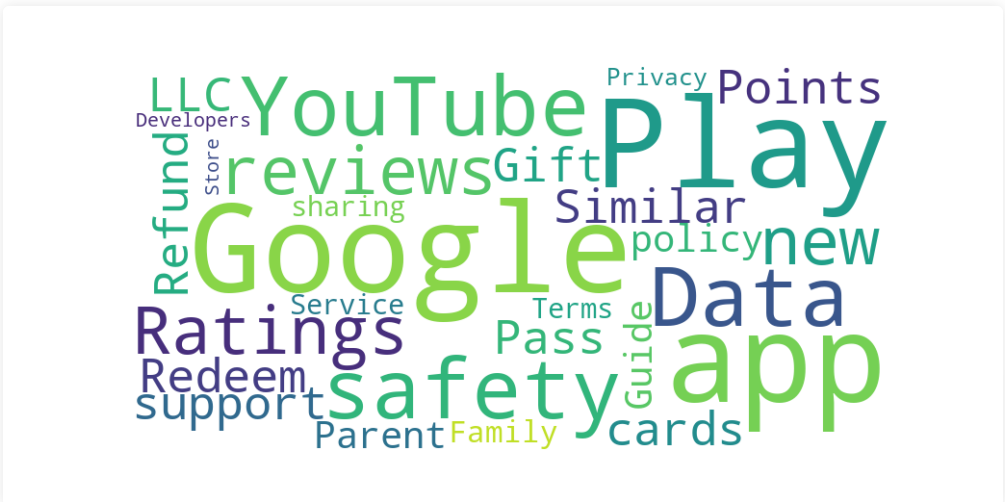
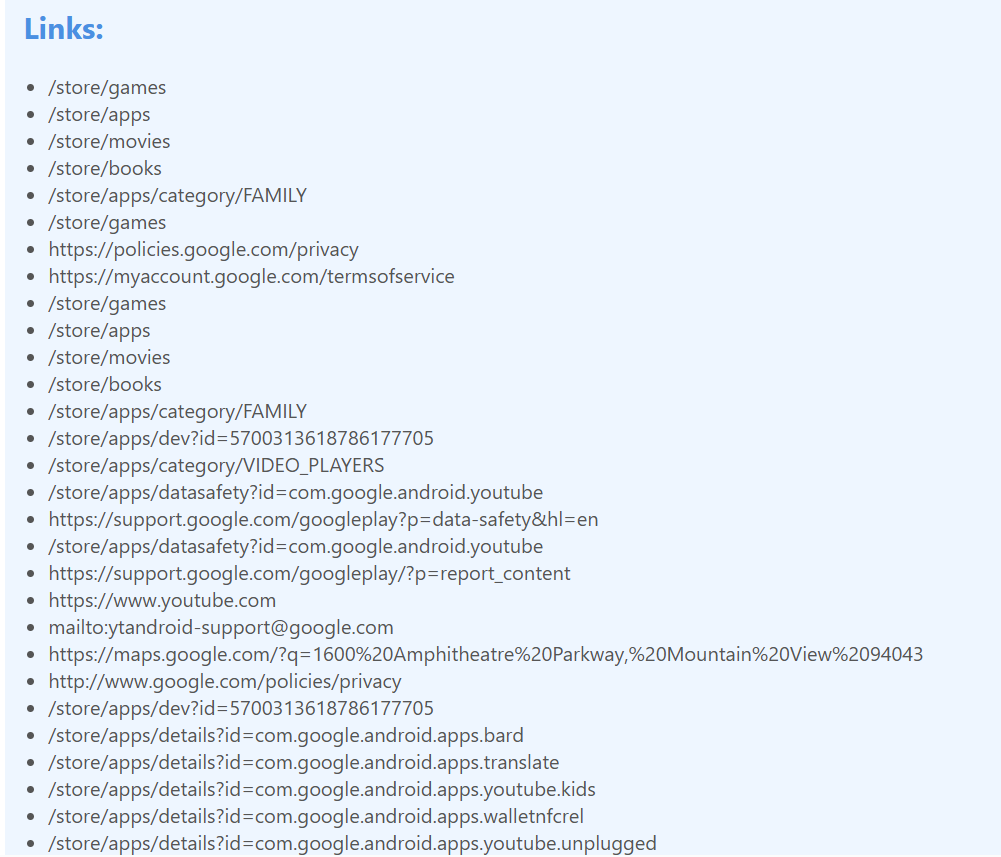
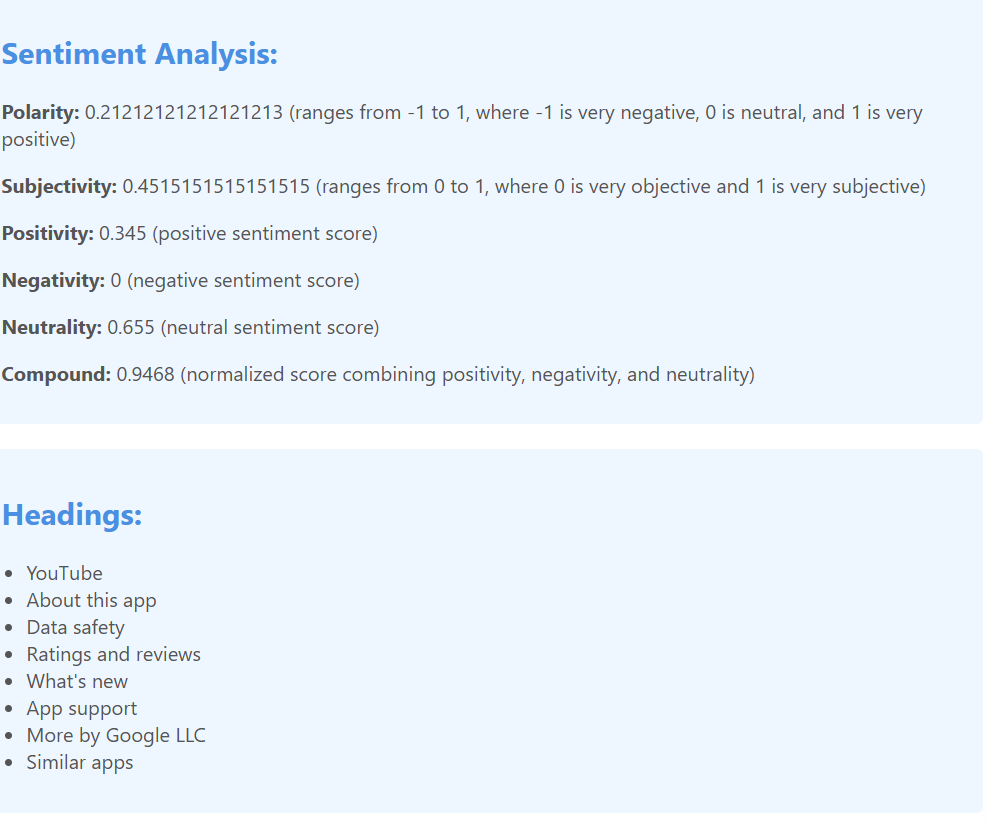
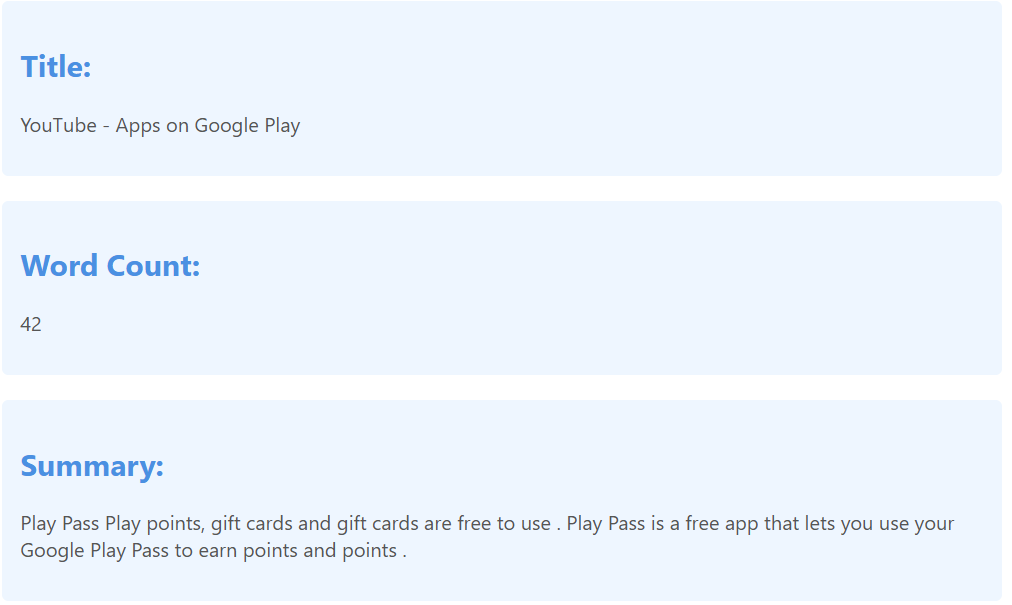
Example for text files



**Scraping Service**

**File**: services/scraping\_service.py

Performs web scraping using BeautifulSoup and requests. Extracts titles, headings, paragraphs, and links from a given URL. Provides summarization and sentiment analysis of the scraped content along with word cloud generation.

Example: scaping site: [YouTube - Apps on Google Play](https://play.google.com/store/apps/details?id=com.google.android.youtube&hl=en)  
  


**Voice Service**

**File**: services/voice\_service.py

Converts speech to text using the SpeechRecognition library and Google Web Speech API. Converts text to speech using the pyttsx3 library.

**Search Service**

**File**: services/search\_service.py

Generates text using the GPT-2 pipeline provided by the transformers library.

When I ask about the president of india, the response is  
A screenshot of a computer

Description automatically generated

**Text Service**

**File**: services/text\_service.py

Answers questions using a question-answering pipeline provided by the transformers library. Generates text based on a given prompt using the GPT-2 pipeline.

**Dependencies**

The project depends on several Python libraries. Ensure you have the following installed:

* Flask
* transformers
* wikipedia-api
* wikipedia
* aiohttp
* SpeechRecognition
* PyPDF2
* pandas
* python-docx
* textblob
* vaderSentiment
* matplotlib
* wordcloud
* pillow
* pytesseract
* pydicom
* beautifulsoup4
* requests
* pyttsx3
* torch

**Copy Rights**

This project is under the LTU-intellectual property, as it is done by me (sriram rampelli) and Lawrence technological university as a collaborative project under the guidance of professor Dr. Wasim Bukatia

**Acknowledgements**

* Hugging Face for the Transformers library.
* Wikipedia for the API and data.
* The various open-source libraries used in this project.

**Contact**

Sriram Rampelli [mail](sriramrampelli15@gmail.com) [linkdln](https://www.linkedin.com/in/sriram-rampelli-b41a75178/)