Configuration Manual

MSc Research Project
Data Analytics

Komal Prakashchandra Patra
Student ID: x22210369

School of Computing
National College of Ireland

Supervisor: Vikas Tomer

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Komal Prakashchandra Patra
Student ID:	x22210369
Programme:	Data Analytics
Year:	2024
Module:	MSc Research Project
Supervisor:	Vikas Tomer
Submission Due Date:	12/08/2024
Project Title:	Configuration Manual
Word Count:	420
Page Count:	7

Signature:	
Date:	15th September 2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Komal Prakashchandra Patra
x22210369

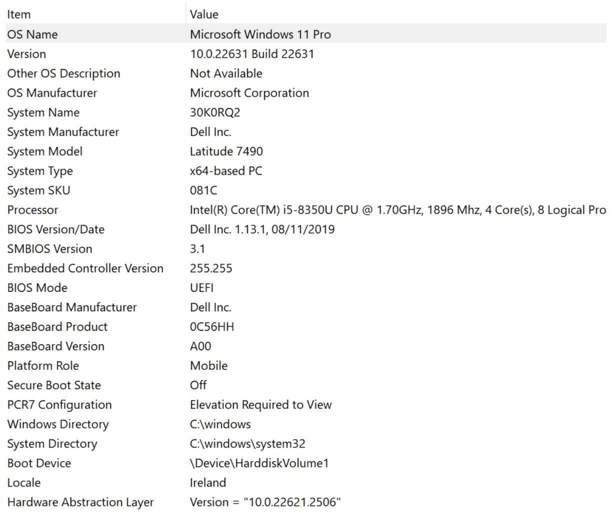
1 Introduction

The configuration Manual explains entire steps needed to implement the Tailored resume generation Using RAG with LLM. The objective behind this research was to implement the concept of RAG and using open-source LLM model for accessibility. The purpose of this setup guide is to deliver information on the programming language that is used, concerning the operating systems and those components and libraries that are fundamental.

2 Software and Hardware Specifications

These section focuses on the hardware and software configuration in which the application was developed. Running LLM model need high compuattaional resources, still the below configuration hardware works with some of the inference platform such as ChatGroq.

2.1 Hardware Requirements

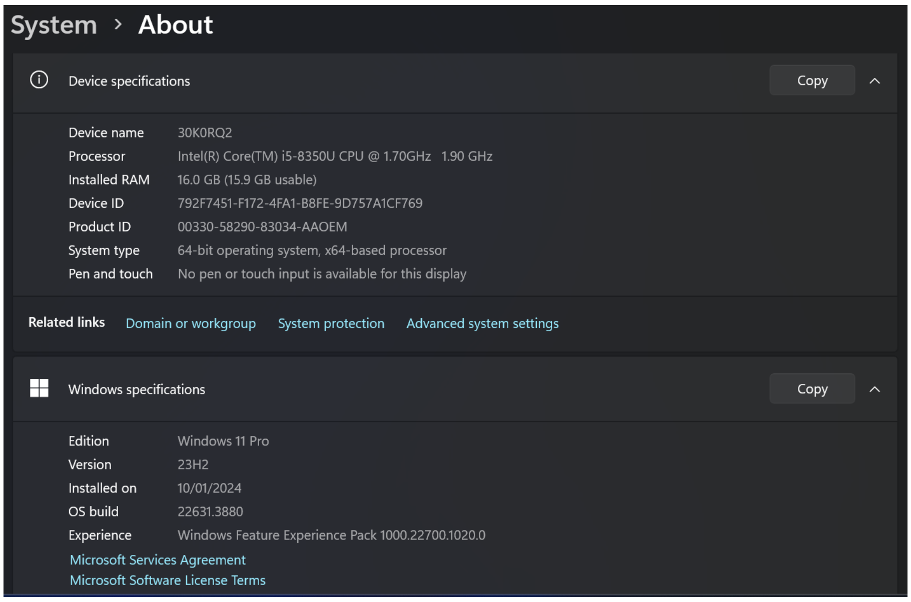


Item	Value
OS Name	Microsoft Windows 11 Pro
Version	10.0.22631 Build 22631
Other OS Description	Not Available
OS Manufacturer	Microsoft Corporation
System Name	30K0RQ2
System Manufacturer	Dell Inc.
System Model	Latitude 7490
System Type	x64-based PC
System SKU	081C
Processor	Intel(R) Core(TM) i5-8350U CPU @ 1.70GHz, 1896 Mhz, 4 Core(s), 8 Logical Pro
BIOS Version/Date	Dell Inc. 1.13.1, 08/11/2019
SMBIOS Version	3.1
Embedded Controller Version	255.255
BIOS Mode	UEFI
BaseBoard Manufacturer	Dell Inc.
BaseBoard Product	0C56HH
BaseBoard Version	A00
Platform Role	Mobile
Secure Boot State	Off
PCR7 Configuration	Elevation Required to View
Windows Directory	C:\windows
System Directory	C:\windows\system32
Boot Device	\Device\HarddiskVolume1
Locale	Ireland
Hardware Abstraction Layer	Version = "10.0.22621.2506"

Figure 1: System Summary

3 Applications used to run/execute the Appliccation

To implement the code, the application and IDE infrastructure is very important. Then the application used are:


System > About

Device specifications

Device name	30K0RQ2
Processor	Intel(R) Core(TM) i5-8350U CPU @ 1.70GHz 1.90 GHz
Installed RAM	16.0 GB (15.9 GB usable)
Device ID	792F7451-F172-4FA1-B8FE-9D757A1CF769
Product ID	00330-58290-83034-AAOEM
System type	64-bit operating system, x64-based processor
Pen and touch	No pen or touch input is available for this display

Related links Domain or workgroup System protection Advanced system settings

Windows specifications

Edition	Windows 11 Pro
Version	23H2
Installed on	10/01/2024
OS build	22631.3880
Experience	Windows Feature Experience Pack 1000.22700.1020.0

Microsoft Services Agreement
Microsoft Software License Terms

Figure 2: Hardware and device info

- Anaconda version
- Google Colab
- MongoDB
- Pinecone
- ChatGroq

4 Libraries Used

The below libraries are kept in requirements.txt file so that at single command, all the packages/libraries will be installed directly.
The command is; pip install -r requirements.txt

- PyMuPDF
- PyPDF2==3.0.1
- nltk
- sentencepiece
- transformers
- pypdf==4.1.0
- typing-inspect==0.9.0
- typing_extensions==4.11.0
- langchain==0.1.16

- langchain-anthropic==0.1.11
- langchain-community==0.0.34
- langchain-core==0.1.46
- langchain-groq==0.1.3
- langchain-openai==0.1.4
- langchain-text-splitters==0.0.1
- streamlit
- sentence_transformers
- pymongo
- accelerate
- langchain_groq
- stqdm
- dotenv

5 Coading files of the Project

5.0.1 Resume Parsing

Resume helpers: This file converts the PDF unto string and extract the text. The text are nalysed and extract the section of the resume with ts structure information. 4 LLM models has been taken as dict to chhose the resume parsing model for a comparison. The prompt template has been used to instruct the model what to do and how to do. After invoking chain it convert them into JSON format.

Parser app: In this file, the other supported files are call to execute the Resume parsing steps.

resume field extraction: This file contains the format structure of how the resume should be extracted based on section and sub- section of resume.

5.0.2 RAG

ingestion.py: This files stores all those JSON files into MongoDB as a vector store as a whole 1 document.

retrieval.py:

This model generates the embeddings for the user query and retrieve the top 10 resume similar and relevant as per the job description.

generation.py:

This files takes the user query and the retrieved document and provide them as an input to the model using promp template to provide instruction and generate the resumes.

Conversation.py: This files uses the Conversational Buffer memory to store the history and content of previous response, so that the user can make their current changes bases on their previous tailored resume.

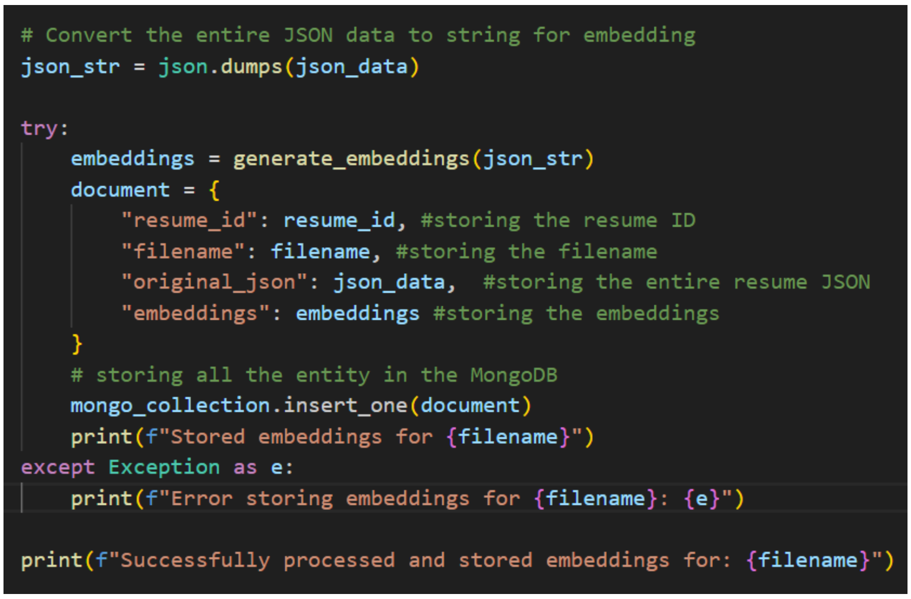
app.py: This files is using streamlit to store the session and create the Simple UI for a user to communicate with the application.

5.0.3 Pinecone

BM25Reranking.ipynb and CrossEncoder Reranking.ipynb: This jupyetre notebooks stores all the resume data into pinecone database to retrieve top k documents and perform the re-ranking Approach using BM25 and Cross encoder re-ranking to enhance the performance of retrieval to get the efficient generated resume.

6 Store the data into Vector databases

Storing the Resume PDFs into MongoDB:



```python
# Convert the entire JSON data to string for embedding
json_str = json.dumps(json_data)

try:
    embeddings = generate_embeddings(json_str)
    document = {
        "resume_id": resume_id, #storing the resume ID
        "filename": filename, #storing the filename
        "original_json": json_data,  #storing the entire resume JSON
        "embeddings": embeddings #storing the embeddings
    }
    # storing all the entity in the MongoDB
    mongo_collection.insert_one(document)
    print(f"Stored embeddings for {filename}")
except Exception as e:
    print(f"Error storing embeddings for {filename}: {e}")

print(f"Successfully processed and stored embeddings for: {filename}")
```

Figure 3: Inserting the mebedding vectors into MongoDB

Using the Vector search pipelines, the top k documents are retrieved from where 130 documenta are firstly select among which 4 documents are retrieved:

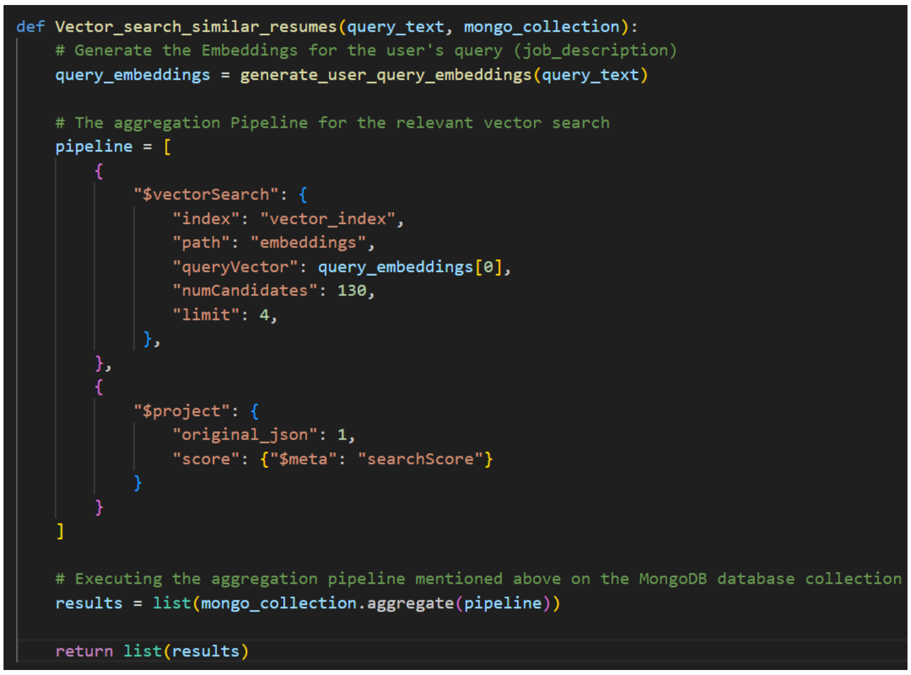
Storing the chunks of PDF into Pinecone:

The documents are then retrieved using below code:

7 Generation

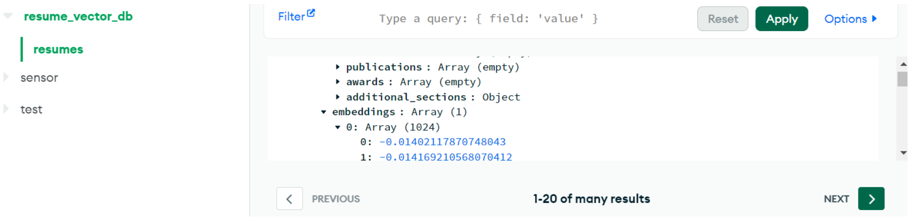
The inputs are needed to be provided in a PromptTemplate:

The LLM chain are invoked for the inputs and the prompts:



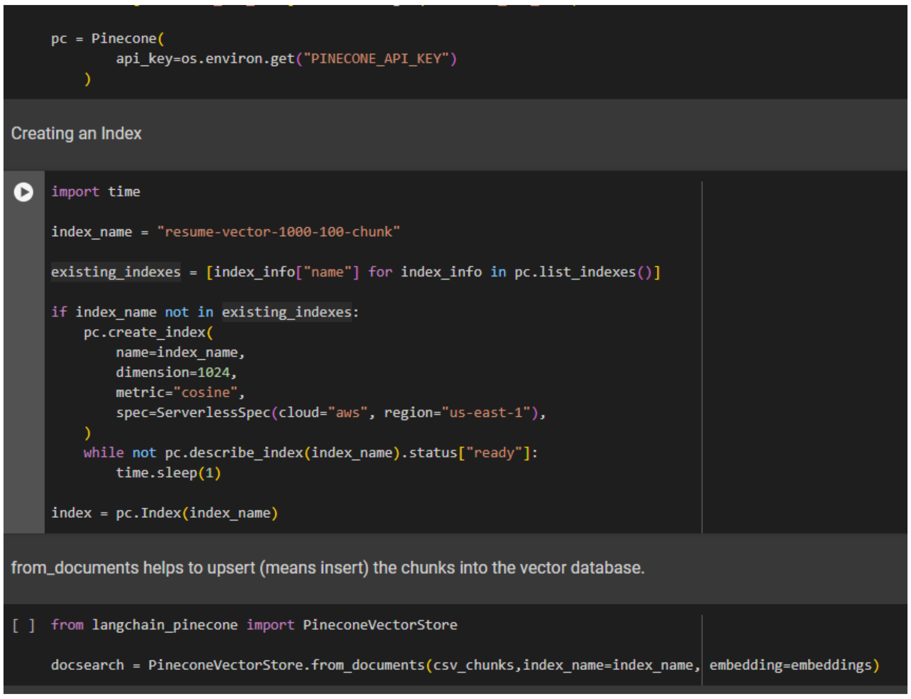
```python
def Vector_search_similar_resumes(query_text, mongo_collection):
    # Generate the Embeddings for the user's query (job_description)
    query_embeddings = generate_user_query_embeddings(query_text)

    # The aggregation Pipeline for the relevant vector search
    pipeline = [
        {
            "$vectorSearch": {
                "index": "vector_index",
                "path": "embeddings",
                "queryVector": query_embeddings[0],
                "numCandidates": 130,
                "limit": 4,
            },
        },
        {
            "$project": {
                "original_json": 1,
                "score": {"$meta": "searchScore"}
            }
        }
    ]

    # Executing the aggregation pipeline mentioned above on the MongoDB database collection
    results = list(mongo_collection.aggregate(pipeline))

    return list(results)
```

Figure 4: The vector search top-k document retrieval



```
resume_vector_db
  resumes
sensor
test

Filter    Type a query: { field: 'value' }    Reset  Apply  Options

      ▸ publications : Array (empty)
      ▸ awards : Array (empty)
      ▸ additional_sections : Object
    ▾ embeddings : Array (1)
      ▾ 0: Array (1024)
          0: -0.01402117870748043
          1: -0.014169210568070412

PREVIOUS    1-20 of many results    NEXT
```

Figure 5: Vector Embeddings in MongoDB



```python
    pc = Pinecone(
            api_key=os.environ.get("PINECONE_API_KEY")
        )
```

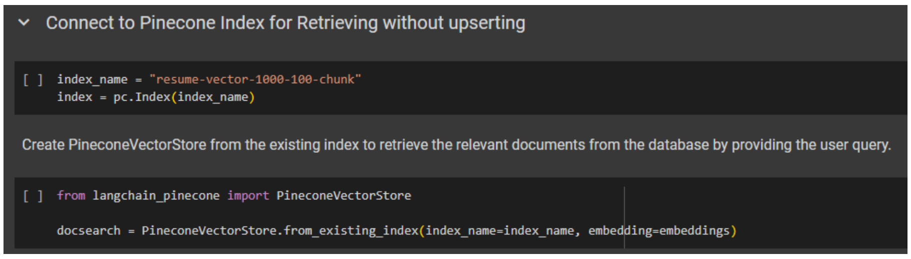
Creating an Index

```python
import time

index_name = "resume-vector-1000-100-chunk"

existing_indexes = [index_info["name"] for index_info in pc.list_indexes()]

if index_name not in existing_indexes:
    pc.create_index(
        name=index_name,
        dimension=1024,
        metric="cosine",
        spec=ServerlessSpec(cloud="aws", region="us-east-1"),
    )
    while not pc.describe_index(index_name).status["ready"]:
        time.sleep(1)

index = pc.Index(index_name)
```

from_documents helps to upsert (means insert) the chunks into the vector database.

```python
[ ] from langchain_pinecone import PineconeVectorStore

    docsearch = PineconeVectorStore.from_documents(csv_chunks,index_name=index_name, embedding=embeddings)
```

Figure 6: Pinecone vector database storage



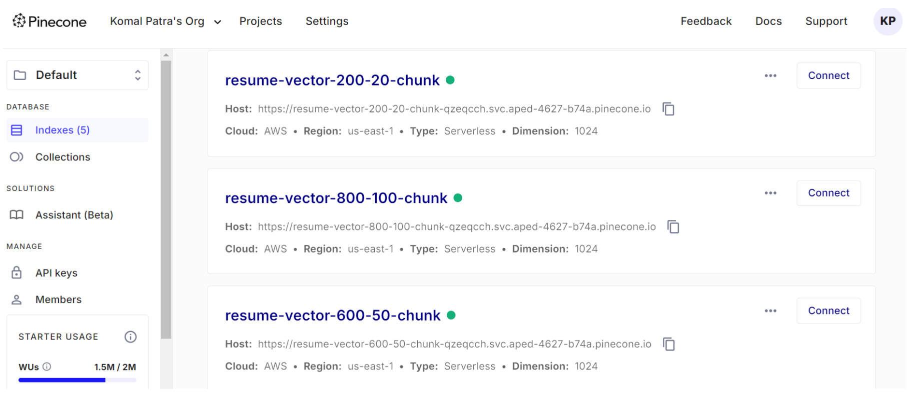
Connect to Pinecone Index for Retrieving without upserting

```
[ ] index_name = "resume-vector-1000-100-chunk"
    index = pc.Index(index_name)
```

Create PineconeVectorStore from the existing index to retrieve the relevant documents from the database by providing the user query.

```
[ ] from langchain_pinecone import PineconeVectorStore

    docsearch = PineconeVectorStore.from_existing_index(index_name=index_name, embedding=embeddings)
```

Figure 7: Pinecone Retrieval code

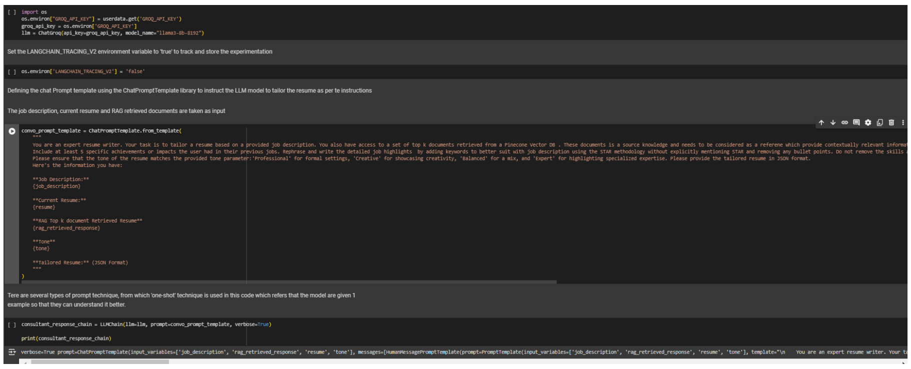


resume-vector-200-20-chunk
Host: https://resume-vector-200-20-chunk-qzeqcch.svc.aped-4627-b74a.pinecone.io
Cloud: AWS • Region: us-east-1 • Type: Serverless • Dimension: 1024

resume-vector-800-100-chunk
Host: https://resume-vector-800-100-chunk-qzeqcch.svc.aped-4627-b74a.pinecone.io
Cloud: AWS • Region: us-east-1 • Type: Serverless • Dimension: 1024

resume-vector-600-50-chunk
Host: https://resume-vector-600-50-chunk-qzeqcch.svc.aped-4627-b74a.pinecone.io
Cloud: AWS • Region: us-east-1 • Type: Serverless • Dimension: 1024

Figure 8: Pinecone Vector DB



```
[ ] import os
    os.environ["GROQ_API_KEY"] = userdata.get('GROQ_API_KEY')
    groq_api_key = os.environ['GROQ_API_KEY']
    llm = ChatGroq(api_key=groq_api_key, model_name="llama3-8b-8192")
```

Set the LANGCHAIN_TRACING_V2 environment variable to 'true' to track and store the experimentation

```
[ ] os.environ['LANGCHAIN_TRACING_V2'] = 'false'
```

Defining the chat Prompt template using the ChatPromptTemplate library to instruct the LLM model to tailor the resume as per te instructions

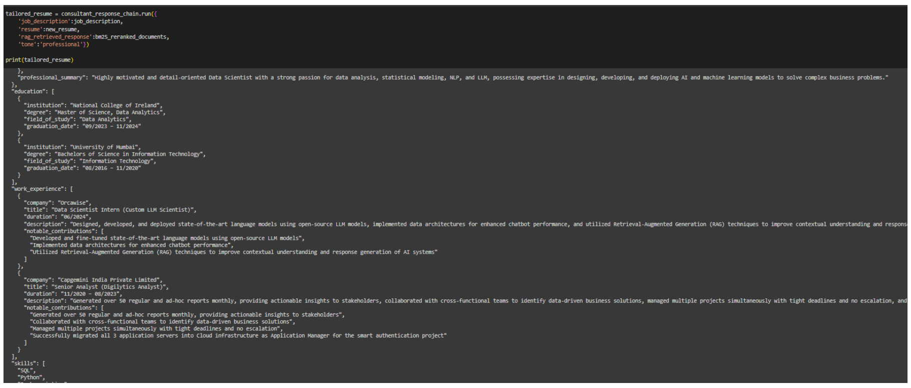
The job description, current resume and RAG retrieved documents are taken as input

```
convo_prompt_template = ChatPromptTemplate.from_template(
    """
    You are an expert resume writer. Your task is to tailor a resume based on a provided job description. You also have access to a set of top k documents retrieved from a Pinecone Vector DB . These documents is a source knowledge and needs to be considered as a referene which provide contextually relevant informa
    Include at least 5 specific achievements or impacts the user had in their previous jobs. Rephrase and write the detailed job highlights  by adding keywords to better suit with job description using the STAR methodology without explicitly mentioning STAR and removing any bullet points. Do not remove the skills a
    Please ensure that the tone of the resume matches the provided tone parameter:'Professional' for formal settings, 'Creative' for showcasing creativity, 'Balanced' for a mix, and 'Expert' for highlighting specialized expertise. Please provide the tailored resume in JSON format.
    Here's the information you have:

    **Job Description:**
    {job_description}

    **Current Resume:**
    {resume}

    **RAG Top k document Retrieved Resume**
    {rag_retrieved_response}

    **Tone**
    {tone}

    **Tailored Resume:** (JSON Format)
    """
)
```

Tere are several types of prompt technique, from which 'one-shot' technique is used in this code which refers that the model are given 1
example so that they can understand it better.

```
[ ] consultant_response_chain = LLMChain(llm=llm, prompt=convo_prompt_template, verbose=True)

    print(consultant_response_chain)

verbose=True prompt=ChatPromptTemplate(input_variables=['job_description', 'rag_retrieved_response', 'resume', 'tone'], messages=[HumanMessagePromptTemplate(prompt=PromptTemplate(input_variables=['job_description', 'rag_retrieved_response', 'resume', 'tone'], template="\n    You are an expert resume writer. Your ta
```

Figure 9: Prompt template with one-shot technique

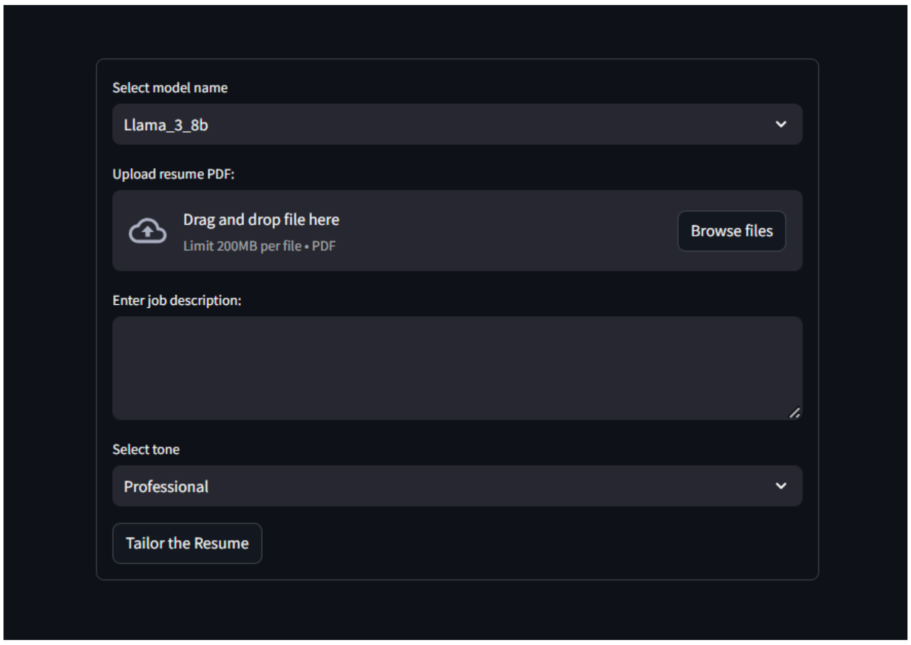


```
tailored_resume = consultant_response_chain.run({
    'job_description':job_description,
    'resume':new_resume,
    'rag_retrieved_response':bm25_reranked_documents,
    'tone':'professional'})

print(tailored_resume)
    },
    "professional_summary": "Highly motivated and detail-oriented Data Scientist with a strong passion for data analysis, statistical modeling, NLP, and LLM, possessing expertise in designing, developing, and deploying AI and machine learning models to solve complex business problems."
  },
  "education": [
    {
      "institution": "National College of Ireland",
      "degree": "Master of Science, Data Analytics",
      "field_of_study": "Data Analytics",
      "graduation_date": "09/2023 - 11/2024"
    },
    {
      "institution": "University of Mumbai",
      "degree": "Bachelors of Science in Information Technology",
      "field_of_study": "Information Technology",
      "graduation_date": "08/2016 - 11/2020"
    }
  ],
  "work_experience": [
    {
      "company": "Orcawise",
      "title": "Data Scientist Intern (Custom LLM Scientist)",
      "duration": "06/2024",
      "description": "Designed, developed, and deployed state-of-the-art language models using open-source LLM models, implemented data architectures for enhanced chatbot performance, and utilized Retrieval-Augmented Generation (RAG) techniques to improve contextual understanding and respons
      "notable_contributions": [
        "Developed and fine-tuned state-of-the-art language models using open-source LLM models",
        "Implemented data architectures for enhanced chatbot performance",
        "Utilized Retrieval-Augmented Generation (RAG) techniques to improve contextual understanding and response generation of AI systems"
      ]
    },
    {
      "company": "Capgemini India Private Limited",
      "title": "Senior Analyst (Digilytics Analyst)",
      "duration": "11/2020 - 08/2023",
      "description": "Generated over 50 regular and ad-hoc reports monthly, providing actionable insights to stakeholders, collaborated with cross-functional teams to identify data-driven business solutions, managed multiple projects simultaneously with tight deadlines and no escalation, and
      "notable_contributions": [
        "Generated over 50 regular and ad-hoc reports monthly, providing actionable insights to stakeholders",
        "Collaborated with cross-functional teams to identify data-driven business solutions",
        "Managed multiple projects simultaneously with tight deadlines and no escalation",
        "Successfully migrated all 3 application servers into Cloud infrastructure as Application Manager for the smart authentication project"
      ]
    }
  ],
  "skills": [
    "SQL",
    "Python",
```

Figure 10: Tailored Resume Generation

8 Interactive chatbot

The User interface is desgned using streamlit:



Select model name

Llama_3_8b

Upload resume PDF:

Enter job description:

Select tone

Professional

Figure 11: Interactive UI chatbot