**Task 1. Resume Parsing and Analysis:**

**- Create a model to extract key information (name, education, work experience, skills) from a set of sample resumes.**

**- Develop an algorithm to match candidate resumes with job descriptions based on the extracted information.**

The **aim** of the task is to extract information from a set of resumes and check for the similarity score between the exracted text and job description.

The approach for this project involved several key steps to effectively parse and analyze resumes, match candidates to job descriptions, and develop a user-friendly interface for these tasks. Firstly, text extraction was implemented for various file formats (PDF, DOCX, TXT) using libraries like PyMuPDF, python-docx, and standard file reading methods. The extracted text was preprocessed by removing the "Objective" section using regular expressions, and named entity recognition with spaCy was utilized to extract candidate names. Specific sections like education, skills, and experience were also identified using regex patterns. The text was further processed by converting it to lowercase, removing non-alphabetic characters, and performing tokenization, stopword removal, lemmatization, and stemming using NLTK. To assess the match between resumes and job descriptions, the text was vectorized using CountVectorizer, and cosine similarity was computed to obtain a match percentage. This entire process was integrated into a Flask web application, allowing users to upload job descriptions and select resume folders for processing, with the results displayed on a web page. This comprehensive approach ensures accurate parsing, effective candidate matching, and an intuitive user experience.

Make sure to install necessary libraries specified in **requirement.txt**.

The Resume parsing folder consists of the code and required files.

The structure of this folder is as follows.

**Resume parsing**

**|\_\_\_\_\_\_resumeparser.py**

**|\_\_\_\_\_\_temp**

**|\_\_\_\_job description files**

**|\_\_\_\_\_\_templates**

**|\_\_\_\_index.html**

Run the **resumeparser.py** file in visual studio.

Copy the local host link(http://127.0.0.1:5000/) to web page. In the webpage give the link to resume folder consisting of resumes(Ex: C:\Users\HP\Desktop\Vyzen\Resume parsing). and select the jobdescription file and press submit.

A table will be displayed in the webpage consisting of Name, Education, Skills, Experience and similarity score.

**Task 3. Chatbot Development:**

**- Design a conversational AI that can answer common candidate queries and guide them through the job application process.**

**- Implement the chatbot to handle interactions effectively using NLP.**

The **aim** of this task is to build a userfriendly chatbot that assist the candidates seeking for jobs.

There are two folders consists of the code and required files.

The provided code implements a job application chatbot using Flask and NLTK for natural language processing. The `JobApplicationChatbot` class initializes with NLTK for tokenization and lemmatization, loading predefined intents for handling user queries such as greetings, job openings, application procedures, and status checks. It includes functionality to preprocess user input, detect intents based on predefined patterns, and generate appropriate responses. The chatbot integrates with Flask to create web routes for user interaction, handling POST requests to receive messages, process them using the chatbot logic, and return responses in JSON format. Overall, it offers a streamlined interface for users to inquire about job-related information and check application statuses interactively.

**Folder 1:bot-manual conversation (this will respond with written text)**

**Folder 2:bot - click and converse (this will respond with selected options)**

The structure of this folder is as follows.

**bot - click and converse/bot-manual conversation**

**|\_\_\_\_\_\_bot.py**

**|\_\_\_\_\_\_application\_status.xlsx**

**|\_\_\_\_\_\_templates**

**|\_\_\_\_\_index.html**

**|\_\_\_\_\_\_static**

**|\_\_\_\_\_chatbot.png**

**|\_\_\_\_\_vyzen\_logo.jfif**

Run the **bot.py** file in visual studio.

Copy the local host link(http://127.0.0.1:5000/) to web page. To the bottom right side of the corner the chatbot is displayed.

Click on it twice. In manual type the questions and in click select the questions.

When the candidate asks to check for status of his/her application, the chatbot asks for the **application\_id(1001-1020).**

Provide the id and press enter/send. It will check the status and display the status.

**Note:** Make sure to change the file/folder path while executing the code. In bot - click and converse(line 60) and in bot-manual conversation(line 77)

**Task 5. Job Recommendation:**

**- Build a job recommendation engine using collaborative filtering or content-based filtering techniques.**

**- Continuously refine the recommendation algorithm based on user feedback and engagement data**

The **aim** of this task is to build a recommendation system based on content based filtering techniques.

This model will recommend the jobs based on candidate profile, prefered job location, job role skills etc.

This Flask application uses data from an Excel file to recommend jobs based on user preferences for skills, preferred job roles, and locations. It preprocesses job data by converting columns to strings and handling NaN values, then applies TF-IDF vectorization to compute skill importance across job listings. Using cosine similarity, it filters and ranks jobs that match user-provided criteria. The application integrates session management to retain user inputs across interactions, ensuring a seamless experience on the web interface where recommendations are dynamically displayed based on user selections. We can use databases like mongodb to store and process the data smoothly.

The structure of this folder is as follows.

**recommendation**

**|\_\_\_\_\_\_\_recommendation.py**

**|\_\_\_\_\_\_\_jobs.xlsx**

**|\_\_\_\_\_\_\_templates**

**|\_\_\_\_\_index.html**

Run the **recom.py** file in visual studio.

Copy the local host link(http://127.0.0.1:5000/) to web page. Enter the profile details and click on update button.

And based on the details entered a recommendation table consisting of 10 jobs will be displayed below in the same webpage.

**Note:** Make sure to change the file/folder path while executing the code.(line 11)