**INSTRUCTIONS OF RUNNING THE CHATBOT**

**1. Uploading the Resume and Reading Its Content**

**Step 1: Uploading the Document**  
To begin, a simple web interface allows the user to upload their resume file. This could be a .pdf or .docx file. Once the file is uploaded, it is temporarily saved and made ready for processing.

**Step 2: Extracting the Text**  
Next, the content from the uploaded file is read and converted into plain text. This is done using a document reader tool that supports the format of the file. The purpose is to collect all the important information written in the resume so it can be analyzed later.

**2. Preparing and Organizing the Resume Text**

**Step 3: Cleaning the Text**  
The raw text from the resume may have extra spaces, headers, or unwanted symbols. These are removed in this step so that only meaningful information is left. This helps improve accuracy in the next stages.

**Step 4: Breaking Into Chunks**  
To manage the data better, the resume text is split into smaller sections called “chunks.” Each chunk may represent a paragraph or section like “Education,” “Projects,” or “Skills.” This helps in understanding each part separately when answering questions.

**Step 5: Assigning Meaning to Each Chunk**  
Now, each of these chunks is turned into a form that the computer can understand and search through. This form is like a summary or signature for each chunk, allowing the system to compare which chunks are more related to a user’s question.

**3.Asking Questions and Matching the Resume Sections**

**Step 6: Accepting User Questions**  
A question is typed into the chatbot, like “List the candidate's projects” or “What skills does the person have?” The system then takes this question and tries to figure out which part of the resume best answers it.

**Step 7: Comparing for Closest Match**  
The system compares the question to all the chunks of resume text and finds the most relevant parts that likely contain the answer. This is similar to how a search engine works—finding the closest match based on meaning.

**4. Generating the Answer and Responding**

**Step 8: Preparing a Response**  
Once the best parts of the resume are identified, they are used to form a proper answer. The answer is built using the exact information from the resume so that it is accurate and to the point.

**Step 9: Showing the Answer on the Web App**  
The final answer is then displayed on the web page. The user sees the reply as if they are chatting with a real assistant. This makes it very easy to extract useful information from any resume quickly.

**5. Keeping the Chat Smooth and Interactive**

**Step 10: Keeping it Fast and Friendly**  
The chatbot responds quickly and can be customized to use colors, animations, or formatting to make the interface look clean and friendly to users.

**How to Access Streamlit and Get ngrok URL**

To use Streamlit for running your chatbot, you need to have Python installed. First, install Streamlit using the command pip install streamlit. After setting up your chatbot script (for example, app.py), open your terminal, go to the project folder, and run streamlit run app.py. This will start a local server, and your chatbot will open in your default browser at http://localhost:8501. It offers an easy-to-use web interface where you can test and interact with your chatbot.

If you want to share your chatbot with others over the internet, you can use **ngrok**. First, go to <https://ngrok.com> and sign up for a free account. Once registered and logged in, you'll receive an **auth token**. Download and install ngrok, then open a terminal and enter the command ngrok config add-authtoken <your\_token> (replacing <your\_token> with your actual token). Now, start your Streamlit app in one terminal, and in another terminal, run ngrok http 8501. This will generate a

**TOOLS AND TECHNOLOGIES**

|  |  |
| --- | --- |
| CATEGORY | TOOLS |
| Web Interface | Streamlit |
| Programming Languages | Python |
| Document Handling | PyMuPDF, docx2txt, python-docx |
| Text Chunking | Langchain |
| Vector Embedding & Search | FAISS |
| Deployment | Hugging Face Spaces, Streamlit Cloud, Ngrok |
| Optional NLP API | OpenAI API, Hugging Face Transformers, |