TalentScout – Hiring Assistant

**1. Introduction**

**1.1 Project Overview**

**TalentScout** is an intelligent recruitment assistant designed to assist recruitment agencies, particularly in the tech industry, with the candidate screening process. The application gathers essential information from candidates, assesses their technical skills based on their self-declared tech stack (e.g., programming languages, frameworks, tools), and generates technical questions tailored to each candidate’s expertise. By leveraging **Large Language Models (LLMs)** like GPT-3/4, the system automates the hiring process, saving time and improving the quality of initial candidate evaluations.

**Key Features:**

* Gathers candidate details like name, email, contact information, and years of experience.
* Collects information on the candidate's tech stack (e.g., Python, Java, React, etc.).
* Dynamically generates technical questions based on the candidate's specified skillset.
* Maintains conversational context to simulate a human-like interaction.
* Provides follow-up questions based on responses, with a graceful conclusion at the end of the session.

**2. Installation**

**2.1 Requirements**

To run **TalentScout**, you need:

* **Python 3.x** installed on your machine.
* **Streamlit** for the frontend interface.
* **Hugging Face's Transformers library** for interacting with pre-trained LLMs.

**2.2 Clone the Repository**

* First, clone the repository from GitHub to your local machine:

git clone https://github.com/your-username/talentscout.git

cd talentscout

**2.3 Install Dependencies**

* Make sure you have **Python** installed, then install the necessary libraries using the following command:

pip install -r requirements.txt

**2.4 Running the Application**

* streamlit run app.py

**3. Usage Guide**

**3.1 Introduction to the Chatbot**

Upon initiating the app, the **TalentScout** chatbot will introduce itself and explain the purpose of the interaction, which is to gather the candidate's information and assess their technical skills.

**3.2 Information Collection**

The chatbot will ask for the following candidate information:

* **Full Name**: To personalize the interaction.
* **Email Address**: For further communication.
* **Phone Number**: An optional input for contact purposes.
* **Years of Experience**: To understand the candidate's level of expertise.
* **Desired Position(s)**: To know what type of roles the candidate is seeking.
* **Current Location**: For regional preferences or constraints.

**3.3 Declaring Tech Stack**

The chatbot will ask the candidate to specify their tech stack, including:

* Programming Languages (e.g., Python, JavaScript, Java, C++)
* Frameworks (e.g., React, Angular, Django, Flask)
* Tools (e.g., Docker, Kubernetes, Git)
* Databases (e.g., MySQL, MongoDB)

**3.4 Technical Question Generation**

Based on the tech stack provided by the candidate, the chatbot will generate **3-5 technical questions** tailored to the candidate’s expertise. For example:

* If the candidate mentions **Python**, the chatbot might ask:
  + "Explain the difference between a list and a tuple in Python."
  + "What is a decorator in Python?"
* If the candidate mentions **Django**, the chatbot might ask:
  + "How does Django’s ORM work?"
  + "Explain the difference between a model and a view in Django."

**3.5 Ending the Conversation**

Once the technical questions are asked, the chatbot will end the conversation by thanking the candidate and providing information on the next steps in the recruitment process (e.g., an interview or further evaluation).

**4. Technical Details**

**4.1 Libraries Used**

* **Streamlit**: Streamlit is used to build the user interface of the application. It provides a simple way to create interactive web apps for data science and machine learning projects.
* **Transformers (Hugging Face)**: This library is used to interact with pre-trained LLMs, specifically GPT-3/4 models. The models are used to generate tailored technical questions based on the candidate’s input.
* **Python**: Python is the primary programming language used for backend logic and data management. The application’s structure uses Python functions for gathering user input, generating questions, and managing the flow of conversation.

**4.2 Model Details**

The chatbot uses a **pre-trained GPT-3/4 model** from OpenAI (or similar LLM) to generate technical questions based on the candidate's declared tech stack. The questions are dynamically crafted to assess the candidate’s proficiency in each technology listed.

**4.3 Architecture**

The application is designed in the following way:

1. **Frontend (UI)**: Built using **Streamlit**, where candidates interact with the chatbot.
2. **Backend (Logic)**: Python manages the chatbot’s flow, including information gathering, question generation, and conversation management.
3. **LLM Integration**: The system interacts with GPT-3/4 via the **Transformers** library to dynamically generate relevant technical questions based on the provided tech stack.

**5. Prompt Design**

**5.1 Information Gathering Prompts**

To ensure efficient data collection, the prompts for gathering candidate information were designed to be clear and concise:

* **Example**: "Please provide your full name."
* **Example**: "What is your current location?"

**5.2 Technical Question Generation Prompts**

For generating tailored technical questions, the prompts were designed to be specific to each technology:

* **Example for Python**: "Generate three technical questions to assess a candidate's proficiency in Python."
* **Example for React**: "Generate three technical questions to assess a candidate's proficiency in React."

The model is prompted to generate 3-5 questions based on the technology listed by the candidate. The questions range from basic to intermediate, ensuring the candidate’s skills are tested thoroughly.

**6. Challenges & Solutions**

**6.1 Challenge: Maintaining Context**

One of the challenges was maintaining context throughout the conversation, especially with multiple steps. **Streamlit’s session\_state** was used to store the user's input and track the conversation. This helped in preserving the state across interactions.

**6.2 Challenge: Generating Relevant Questions**

Another challenge was generating relevant technical questions for various tech stacks. This was addressed by crafting specific prompts for each technology, ensuring that the LLM understood the intricacies of the technologies.

**6.3 Challenge: Handling Unexpected Inputs**

Handling unexpected or incomplete inputs was another challenge. To mitigate this, the application was designed to prompt the candidate to clarify or re-enter information if the input did not match expected formats. Clear error messages were provided in such cases.

**6.4 Challenge: Data Privacy**

Since sensitive candidate information is being collected, it was essential to ensure compliance with privacy standards (e.g., **GDPR**). **Anonymized or simulated data** was used during development, and a proper data storage and deletion policy was implemented to protect user information.

This documentation provides all the essential details about the **TalentScout** application, including installation, usage, technical details, prompt design, challenges, and future enhancements. It's designed to give users and developers a clear understanding of how the system works and how to use it effectively.