# TalentScout Hiring Assistant - Documentation

## Overview

TalentScout Hiring Assistant is a chatbot-based Streamlit application that interacts with candidates to gather their details and assess their technical skills based on their tech stack. The application integrates with Groq's API for AI-driven question generation.

## Features

* Interactive chatbot for candidate information collection
* Integration with Groq's API for AI-based response generation
* Automated technical question generation based on the candidate's tech stack
* Streamlit-powered UI for a seamless experience
* Docker support for easy deployment

## Prerequisites

Ensure you have the following installed:

* Python 3.9+
* pip for dependency management
* Streamlit for UI rendering
* Langchain and dotenv for AI integration
* Docker for containerization (if required)

## Installation

### 1. Clone the Repository

$ git clone <repository-url>

$ cd talent-scout-hiring-assistant

### 2. Install Dependencies

$ pip install -r requirements.txt

### 3. Set Up Environment Variables

Create a .env file and add the required API key:

GROQ\_API\_KEY=<your-groq-api-key>

Alternatively, you can set the API key directly in the script using:

os.environ['GROQ\_API\_KEY'] = 'your-api-key-here'

## Application Structure

### 1. ****Groq API Integration****

The application utilizes Langchain and ChatOpenAI from langchain\_community.chat\_models to interact with Groq's API for AI-based text generation.

#### Function: get\_groq\_client()

* Configures and returns a ChatOpenAI instance for API interaction.
* Uses Groq API to generate responses.
* Function: generate\_response(prompt: str)
* Calls get\_groq\_client() to obtain a response based on the input prompt.
* Returns the AI-generated content.

### 2. ****Session State Management****

The chatbot maintains conversation history and candidate data using Streamlit's session state (st.session\_state).

#### Session Variables:

#### messages: Stores chat history.

#### candidate\_data: Stores user details including:

#### full\_name, email, phone, years\_of\_experience, desired\_position, location, tech\_stac

### 3. ****Chatbot Conversation Flow****

#### Function: handle\_conversation(user\_input: str)

#### Collects user details step by step.

#### If all details are collected, generates technical questions based on the provided tech stack.

#### Stores messages in session state for a smooth conversational flow.

### 4. ****Streamlit UI****

The UI is built using Streamlit components:

* st.title() for the chatbot header.
* st.chat\_message() for displaying messages
* st.chat\_input() for user input.
* st.rerun() to refresh the conversation dynamically.

## Running the Application

Start the Streamlit application:

$ streamlit run chatboatproject.py

## Docker Setup

### 1. ****Building the Docker Image****

$ docker build -t talent-scout-bot .

### 2. ****Running the Docker Container****

$ docker run -p 8505:8505 talent-scout-bot

### 3. ****Health Check****

The container includes a health check mechanism:

HEALTHCHECK --interval=30s --timeout=30s --start-period=5s --retries=3 \

CMD curl --fail http://localhost:8505/\_stcore/health || exit 1

Ensures the app is running properly inside the container.

## Conclusion

This project simplifies the hiring process by automating candidate screening and skill assessment. It is a scalable solution that can be deployed locally or in a cloud environment using Docker.