**Hiring Assistant Chatbot Documentation**

**##Project Overview**

The \*\*Hiring Assistant Chatbot\*\* is an intelligent recruitment assistant designed for "TalentScout," a fictional recruitment agency. The chatbot leverages the power of Google Generative AI (Gemini) to streamline the initial candidate screening process. It gathers essential candidate information, generates tailored technical questions based on their declared tech stack, and provides an intuitive, interactive experience.

**### Key Capabilities:**

- Collects candidate details such as name, contact information, experience, and tech stack.

- Generates relevant technical questions to evaluate candidate proficiency in specific technologies.

- Provides context-aware and seamless interactions.

- Features an intuitive interface using \*\*Streamlit\*\*.

---

**## Installation Instructions**

**### 1. Prerequisites**:

- \*\*Python 3.8+\*\*

- \*\*Google Cloud Account\*\* with access to the Gemini API (Generative AI API).

- \*\*ngrok\*\* account for exposing the local app to the web (optional).

**### 2. Clone or Download the Project**:

Download the project files to your local machine or Google Colab.

**### 3. Install Required Libraries:**

Run the following commands to install dependencies:

```bash

pip install google-generative-ai streamlit pyngrok

```

**### 4. Set Up API Keys:**

- Obtain your \*\*Google Generative AI API Key\*\* from the [Google Cloud Console](https://console.cloud.google.com/).

- Obtain your \*\*ngrok Authentication Token\*\* from [ngrok](https://ngrok.com/).

- Add the API Key and ngrok token to the code:

```python

genai.configure(api\_key="YOUR\_GOOGLE\_GEMINI\_API\_KEY")

ngrok.set\_auth\_token("YOUR\_NGROK\_AUTH\_TOKEN")

```

**### 5. Run the Application:**

Run the Streamlit app locally:

```bash

streamlit run hiring\_assistant.py

```

**### 6. Expose the App with ngrok (Optional):**

Use ngrok to make the app accessible publicly:

```python

from pyngrok import ngrok

public\_url = ngrok.connect(8501).public\_url

print(f"Public URL: {public\_url}")

```

Access the app via the provided public URL.

---

**## Usage Guide**

**### Steps to Use the Chatbot**:

1. Open the chatbot interface in your browser.

2. Fill out the candidate details form, including name, email, experience, and tech stack.

3. Submit the form to generate tailored technical questions based on the tech stack.

4. View the generated questions and use them for candidate evaluation.

5. End the chat session when finished.

---

**## Technical Details**

**### Libraries Used:**

- \*\*google-generative-ai\*\*: For AI-driven content generation.

- \*\*Streamlit\*\*: To create the user interface.

- \*\*pyngrok\*\*: For exposing the local app to the web.

**### Model Details:**

- \*\*Google Gemini Model\*\*: `gemini-1.5-flash` was used to generate content.

**### Architectural Decisions:**

- \*\*Prompt Engineering\*\*: Carefully designed prompts to ensure relevant question generation.

- \*\*Form-Based Input\*\*: Streamlit's form components for intuitive data collection.

- \*\*Error Handling\*\*: Integrated error handling to manage API failures and unexpected inputs.

---

**## Prompt Design**

The prompts were crafted to:

1. Gather candidate details in a structured format.

2. Instruct the model to generate tailored technical questions based on the tech stack.

Example Prompt:

```

You are a recruitment assistant for a technology company. Based on the candidate's information, generate 3-5 technical questions for each technology they are proficient in.

Candidate Details:

Name: John Doe

Tech Stack: Python, Django, MySQL

```

---

**## Challenges & Solutions**

**### 1. Challenge: API Method Mismatch**

- \*\*Problem\*\*: Initially, the wrong API method (`chat`) was used.

- \*\*Solution\*\*: Researched and corrected the implementation to use the `generate\_content` method with the `GenerativeModel` class.

**### 2. Challenge: Dependency Issues**

- \*\*Problem\*\*: Missing or outdated libraries.

- \*\*Solution\*\*: Updated and reinstalled all required libraries with the correct versions.

**### 3. Challenge: Exposing Local App**

- \*\*Problem\*\*: Streamlit app URLs were inaccessible from Colab.

- \*\*Solution\*\*: Used ngrok to expose the app to a public URL.

**### 4. Challenge: Prompt Relevance**

- \*\*Problem\*\*: Generated questions were sometimes generic.

- \*\*Solution\*\*: Improved prompts by including context and structuring inputs clearly.

---

**## Conclusion**

The Hiring Assistant Chatbot demonstrates the potential of AI-driven recruitment tools to streamline candidate evaluation. By leveraging advanced language models and intuitive user interfaces, the chatbot provides an efficient, scalable solution for tech recruitment processes.