# Hiring Assistant Chatbot – Project Documentation

📚 Project Overview

The Hiring Assistant Chatbot is an intelligent system built using Flask and Streamlit to automate the candidate evaluation process for a fictional recruitment agency named TalentScout. The chatbot collects candidate information, generates relevant technical questions based on their technology stack, evaluates the responses, and provides feedback.

🎯 Objective

The goal of this project is to:  
- Automate candidate evaluation.  
- Generate relevant technical questions dynamically.  
- Assess candidate responses and provide accurate feedback.  
- Maintain context during candidate interactions.

🛠️ Technologies Used

1. Python: Core programming language for backend logic.  
2. Flask: Web framework for handling API requests and serving the chatbot.  
3. Streamlit: To create an interactive and user-friendly frontend.  
4. OpenAI API / LLM Models: For generating technical questions and evaluating candidate responses.  
5. Pandas: To handle and process CSV files for questions and candidate data.  
6. CSV Files: Storing candidate responses and question data.  
7. Virtual Environment: Isolated Python environment (hiring\_assistant\_env).

🔥 Key Features

1. Candidate Data Collection:  
 - Gathers candidate information such as name, email, position, and tech stack.  
 - Stores candidate responses in a candidate\_data.csv file.

2. Question Generation:  
 - Dynamically generates technical questions based on the candidate’s selected position and skill set.  
 - Retrieves questions from the questions.csv file.

3. Evaluation and Feedback:  
 - Evaluates candidate responses using OpenAI API or a pre-trained LLM model.  
 - Provides relevant feedback based on answers.

4. CSV Management:  
 - Stores and manages questions and candidate responses in CSV files (questions.csv and candidate\_data.csv).  
 - Loads and validates questions with error handling to prevent incorrect formats.

5. Document Generation:  
 - Generates and serves a project\_doc.docx file with project details and candidate evaluation reports.

📝 How It Works

🎤 Step 1: Collect Candidate Information  
- The chatbot asks for the candidate's name, email, desired position, and tech stack.  
- Data is stored in candidate\_data.csv.

📚 Step 2: Generate Technical Questions  
- Based on the selected position and skill set, questions are loaded from questions.csv.  
- If no relevant questions are found, the system generates questions using an LLM model.

✨ Step 3: Evaluate Candidate Responses  
- The candidate answers the generated questions.  
- Answers are processed and evaluated using a pre-trained LLM or defined rules.

📊 Step 4: Provide Feedback  
- The system provides feedback on candidate responses and highlights areas of improvement.  
- Results are stored in candidate\_data.csv for future reference.

📂 File Structure

/ hiring\_assistant\_chatboat  
├── chatbot.py # Main application logic  
├── project\_doc.docx # Project documentation  
├── candidate\_data.csv # Candidate response storage  
├── questions.csv # Question dataset  
└── hiring\_assistant\_env/ # Virtual environment folder

🚀 How to Run the Project

1. Activate Environment:  
```  
cd C:\Users\dell\flask\_project\hiring\_assistant\_chatboat  
hiring\_assistant\_env\Scripts\activate  
```  
2. Run Flask App:  
```  
python chatbot.py  
```  
3. Run Streamlit App (if applicable):  
```  
streamlit run chatbot.py  
```  
4. Open in Browser:  
```  
http://127.0.0.1:5000/  
```  
🎁 Download Project Documentation  
To download the project documentation as a .docx file, visit:  
```  
http://127.0.0.1:5000/download/project\_doc.docx  
```