# InterviewBot Documentation

## Overview

InterviewBot is a Python-based automated interviewing system that uses the Transformers library to conduct interviews, evaluate responses, and provide feedback. The system leverages the TinyLlama model for natural language processing and scoring capabilities.

## Installation Requirements

- transformers library

- PyTorch

- datasets library

- Hugging Face account and token

## Class Structure

### InterviewBot Class

The main class that handles all interviewing functionality.

### Core Methods

def \_\_init\_\_(self, model\_name: str, question\_file: str):  
 """  
 Initialize the InterviewBot with a specific model and questions file.  
   
 Args:  
 model\_name (str): Name of the pretrained model to use  
 question\_file (str): Path to JSON file containing interview questions  
 """

### Key Methods

- \*\*load\_questions(json\_file)\*\*: Loads interview questions from a JSON file

- \*\*ask\_question()\*\*: Returns the current question or ends the interview

- \*\*process\_answer(user\_input)\*\*: Evaluates user responses and generates feedback

- \*\*end\_interview()\*\*: Concludes the interview and calculates final score

- \*\*fine\_tune(training\_data\_path, output\_dir)\*\*: Fine-tunes the model on custom data

## Data Format

The questions file should be in JSON format with the following structure:

[  
 {  
 "question": "Your interview question here",  
 "expected\_answer": "Optional expected answer"  
 }  
]

## Usage Example

from interview\_bot import InterviewBot  
  
model\_name = "TinyLlama/TinyLlama-1.1B-Chat-v1.0"  
question\_file = "/content/questions\_answers.json"  
  
interviewer = InterviewBot(model\_name, question\_file)  
print("Welcome to the Interview Bot!")  
  
# Start the interview loop  
while True:  
 question = interviewer.ask\_question()  
 if "completed" in question:  
 print(question)  
 break  
   
 print("\nQuestion:", question)  
 answer = input("Your answer: ")  
 feedback = interviewer.process\_answer(answer)  
 print("\nFeedback:", feedback)

## Fine-tuning

The system supports model fine-tuning with custom training data. The training data should be in JSON format with 'prompt' and 'response' fields.

### Fine-tuning Parameters

- \*\*training\_data\_path\*\*: Path to training data file

- \*\*output\_dir\*\*: Directory to save the fine-tuned model

- \*\*epochs\*\*: Number of training epochs (default: 3)

- \*\*learning\_rate\*\*: Learning rate for training (default: 5e-5)

## Security Notes

The system requires a Hugging Face token for authentication. Store this token securely and never commit it to version control.

## Limitations

- Requires active internet connection for model loading

- Response evaluation depends on model quality

- Score extraction assumes specific format in model output