ChatGPT

Interview Simulator - README & Project Report

Overview

This is a Streamlit-based AI Interview Simulator app that allows users to:

- Practice interview questions by selecting roles (Data Analyst, Data Scientist, ML Engineer)
- Generate custom or random questions
- Auto-generate sample answers using a local LLM (FLAN-T5)
- Submit their answers for AI-powered feedback, score, and category assessment
- Download session history

Features

- Local LLM (FLAN-T5 via HuggingFace Transformers)
- No API key or internet dependency for inference
- Role-based and custom questions
- Descriptive feedback + score out of 10
- Communication vs Technical categorization
- CSV export of session history

Project Structure

```
Interview Simulator GenAI/
├── app.py                  # Main Streamlit app script
├── interview_log.csv       # Auto-generated session history
├── README.md               # This file
└── requirements.txt        # Python dependencies
```

► How to Run This App

1. Create a Virtual Environment

```
python -m venv venv
venv\Scripts\activate     # Windows
source venv/bin/activate  # Mac/Linux
```

2. Install Dependencies

```
pip install -r requirements.txt
```

3. Run the App

```
streamlit run app.py
```

4. App URL

Open in browser: http\://localhost:8501

Dependencies

```
streamlit
transformers
pandas
python-dotenv
```

🏗 Project Architecture

```
graph TD
    A[User] --> B[Streamlit Interface]
    B --> C[Role/Question Input]
    B --> D[Buttons: Get Answer / Get Feedback]
    D --> E[FLAN-T5 Model Pipeline (local)]
    E --> F1[Answer Generation]
    E --> F2[Feedback + Score Parsing]
    F2 --> G1[Score Extraction (Regex)]
    F2 --> G2[Category Identification]
    F1 --> H[User Answer Box]
    G1 --> I[Feedback Display]
    G2 --> I
    I --> J[Session Log (CSV)]
```

📈 Project Report

Objective

To build an AI-powered interview simulator for role-based practice using open-source LLMs (no API keys required).

Technologies Used

- Streamlit (frontend UI)
- HuggingFace Transformers (FLAN-T5 model)
- pandas (logging & CSV)
- Regex (score & feedback parsing)

Key Functionalities

- Custom question support with AI-generated answers
- Local inference using FLAN-T5 (no external call latency)
- Session logging and download
- Scoring with feedback categories

Challenges Solved

- Parsing model outputs into separate feedback, score, and category
- Maintaining session history across multiple questions
- Managing Streamlit state for questions vs answers

Limitations

- FLAN-T5 model sometimes outputs vague or partial responses
- Feedback parsing relies on pattern matching
- Not intended for production-grade accuracy (demo-level quality)

Future Enhancements

- Switch to more powerful local models like DeepSeek or Mistral (if GPU available)
- Add user profile saving
- Real-time interview simulations with timers
- Graphical feedback visualization (score bars)

💻 Author

Minal Pawar

Built with ❤ using Streamlit + Transformers

License

MIT License (add if needed)