Hiring Assistant Chatbot — Streamlit

Project Overview

The **Hiring Assistant** is a lightweight Streamlit application that collects candidate information and generates targeted technical interview questions based on a declared tech stack. The app is designed as a demo/prototype for interview automation and hiring workflows.

Key features: - Candidate information collection (Full name, Email, Phone, Years of experience, Desired position(s), Location). - Tech stack declaration (languages, frameworks, databases, tools). - LLM-driven generation of 3–5 technical questions per declared technology, with expected answers and difficulty tags. - Context handling via Streamlit session state. - Exit keywords to gracefully end conversations. - Admin dashboard to view anonymized candidate records (password-protected). - Simulated/anonymized local storage of candidate records. - Docker-ready with a `Dockerfile` and CI workflow for Streamlit Cloud deployment.

Table of Contents

Quick Start

1. Clone the repository:

```
git clone <your-repo-url>
cd hiring-assistant-streamlit
```

1. Create and activate a Python virtual environment:

```
python -m venv .venv
source .venv/bin/activate  # macOS/Linux
.\.venv\Scripts\activate   # Windows (PowerShell)
```

1. Install dependencies:

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

1. Set environment variables (see next section).

2. Run the app:

```
streamlit run hiring_assistant_streamlit.py
```

Open http://localhost:8501 in your browser.

Environment Variables

Set these environment variables before running the app or configuring CI:

- `OPENAI_API_KEY` — (optional) API key for OpenAI. If omitted, the app uses deterministic stub output for testing.
- `OPENAI_MODEL` — (optional) model name to use (default `gpt-4o-mini` in examples).
- `HF_API_URL` — (optional) HuggingFace inference endpoint URL if using HF as a fallback.
- `HF_API_TOKEN` — (optional) token for HF endpoints.
- `ADMIN_PASSWORD` — Password for accessing the admin dashboard (required for production).
- `ANON_SALT` — Salt string used for deterministic anonymization of PII.

Important: Never commit secrets to source control. Use repository secrets and environment variable injection for CI/CD and cloud deployments.

Installation

Dependencies are listed in `requirements.txt` (Streamlit, OpenAI, python-dotenv, etc.). Install them with pip as shown in the Quick Start.

If you plan to run inside Docker (recommended for reproducible environments), see the Docker section.

Running Locally

1. Ensure env vars are set (see above).
2. Launch Streamlit: `streamlit run hiring_assistant_streamlit.py`.
3. Use the left pane to enter candidate details and tech stack, then click **Save & Generate Questions**.
4. Admins: open the Admin section, provide `ADMIN_PASSWORD`, and view anonymized records.

Docker

Build and run the Docker image:

```
# Build
docker build -t hiring-assistant:latest .

# Run
docker run -it -p 8501:8501
  -e ADMIN_PASSWORD="supersecret"
  -e ANON_SALT="my_secret_salt"
  -e OPENAI_API_KEY="sk-..."
  hiring-assistant:latest
```

Then visit `http://localhost:8501`.

If you want Docker Compose (not included by default), you can add a `docker-compose.yml` that mounts local files and sets environment variables.

GitHub Actions / Streamlit Cloud

This repo includes a GitHub Actions workflow to validate the app and prepare for deployment (stored under `.github/workflows/`). Streamlit Cloud can connect to the repository and automatically redeploy on new commits to `main`.

Secrets to add to GitHub repo settings: - `OPENAI_API_KEY` (optional) - `ADMIN_PASSWORD` (required) - `ANON_SALT` (recommended) - `STREAMLIT_DEPLOY_TOKEN` (if using automated deployment action)

Alternatively, connect the repo via the Streamlit Cloud dashboard and configure environment variables through their UI.

Prompt Design & LLM Integration

The application uses a small system prompt that instructs the model to produce 3–5 questions per technology and to return an expected answer and difficulty tag. Key prompt design choices:

- Low temperature (`0.2`) for stable, reproducible outputs.
- System prompt explicitly requests JSON output to simplify parsing.
- Heuristic JSON extraction when the LLM returns extraneous text.

Supported providers (in the demo): - OpenAI via `openai` package. - Optional HuggingFace-style endpoint via `HF_API_URL`.

Note: LLMs can be inconsistent in their output format. For production use, prefer stricter output constraints (e.g., JSON schema enforced by a parsing/validation step) and add validation/retry logic.

Data Handling & Privacy

This app stores candidate PII in **anonymized form only** (SHA256 with salt) in a local JSON file `candidates.json`. The raw email and phone are not stored in the repository by default.

For production: - Use encrypted storage (e.g., DB with encryption-at-rest). - Minimize PII collection when possible. - Implement role-based access control and logging. - Comply with local privacy regulations (e.g., GDPR) regarding retention, access, and deletion.

Admin Dashboard

The admin dashboard is accessible from the sidebar and requires `ADMIN_PASSWORD`. It shows a table of anonymized candidate records and allows exporting an anonymized JSON file.

Improvements to consider: - Replace basic password auth with OAuth / Single Sign-On. - Audit logs for access. - Rate-limiting and brute-force protection.

Extending the App

Ideas and next steps: - Add `docker-compose.yml` for local development orchestration. - Replace the admin password with OAuth or integrate with an Identity Provider (Okta, Google Workspace). - Add CI tests and LLM output schema validation. - Enhance prompts for difficulty calibration and role-specific question generation (backend vs frontend vs data engineer). - Deploy to AWS/GCP via ECS/Fargate or to Streamlit Cloud for easy hosting.

Limitations & Challenges

- **LLM output format variability:** The app implements heuristics to extract JSON; however, robust schema validation is recommended.
- **Security & privacy:** This is a demo. Production systems require secure secrets management, encryption, and compliance.
- **Scale:** Streamlit is best for small-medium workloads; for large-scale interview automation, consider microservices and a proper backend API.

License

This project is provided as-is under the MIT License. See `LICENSE` for details.

If you'd like, I can now: - Add a `docker-compose.yml` file - Harden admin auth (OAuth/JWT) - Add automatic LLM output validation (JSON schema) - Produce a one-click Streamlit Cloud deployment guide with screenshots

Which should I do next?