Documentation: AI Policy Assistant

AI Policy Assistant is a secure, multilingual, on-premises tool that empowers employees to instantly get answers to HR and company policy questions by searching and analysing internal documents—using natural language, voice, and multiple file formats.

Project Overview

AI Policy Assistant is a lightweight, private, and interactive assistant for internal SOP and policy search. Employees can upload company documents (PDF, DOCX, images, etc.), ask questions in any language or by voice, and receive accurate, referenced answers—without sending data to the cloud.

Example queries:

- "What is our notice period policy?"
- "Can I encash unused leave?"
- "Who do I contact for laptop issues?"

Target users:
All company employees seeking instant, accurate policy answers.

Key Features

- **Conversational Q&A:** Natural chat interface for asking policy questions in plain language.
- **Multi-document & Multi-format Upload:** Supports PDF, DOCX, TXT, spreadsheets, and images (with OCR), and can search across multiple documents at once.
- **Voice-to-Text Input:** Ask questions by speaking—no typing required.
- **Multilingual Support:** Ask and receive answers in 50+ languages.
- **Semantic Search:** Finds the most relevant policy text using fast, local embeddings.

- **Source Citations:** Every answer is accompanied by document and page references.
- **Entity Extraction:** Highlights key terms, dates, and names in answers.
- **Chat History:** Review your previous questions and answers at any time.
- **Local & Secure:** All processing happens on your machine—no cloud, no data leaks.
- **Query Logging:** All Q&A interactions are logged locally for audit and improvement.

Solution Architecture

1. Document Ingestion & Indexing

- **Parsing:** Extracts text from PDFs, DOCX, TXT, and images (OCR via Tesseract).
- **Chunking:** Splits documents into logical sections for better retrieval.
- **Embeddings:** Uses a quantized ONNX SentenceTransformer for fast, multilingual semantic search.
- **Vector Store:** Stores embeddings in a local FAISS index for instant retrieval.

2. Query Handling (RAG Pipeline)

- **User Query:** Accepts questions via chat or voice.
- **Retrieval:** Embeds the query, retrieves top-k relevant chunks from the vector store, supporting multi-document context.
- **Answer Generation:** Local LLM (Llama 3.2 via Ollama) generates a conversational answer, citing sources.
- **Citation:** Displays the answer with references to original document sections.
- **Logging:** Stores each Q&A pair with metadata for future reference.

3. Advanced Features

- **Multi-document Support:** Synthesizes answers from multiple uploaded documents in a single response.
- **Multilingual Q&A:** Embeddings and answer generation work in multiple languages.
- **Logging:** Every Q&A is logged locally with timestamp, query, answer, sources, and detected language.

- **Chat History:** Scrollable, persistent chat history panel for reviewing past interactions.
- **OCR Image Extraction:** Extracts text from scanned images and photos using Tesseract OCR, with preprocessing for better accuracy.
- **Voice Input:** Employees can ask questions by voice, transcribed locally with Vosk.

4. Security & Privacy

- **Local Deployment:** All components run on company servers or your local machine.
- **No External Calls:** No cloud APIs; all data stays private.
- **Access Control:** Restrict document access and logging to authorized users.
- **Encryption:** Store logs and document data securely.

Model Details

- **Embedding Model:**
    - **Type:** Quantized ONNX SentenceTransformer (paraphrase-multilingual-MiniLM-L12-v2)
    - **Parameters:** 12 transformer layers, 384 hidden sizes, supports 50+ languages.
    - **How Used:** Converts document chunks and user queries into dense semantic vectors for similarity search.
    - **Why ONNX?** Quantization enables fast inference and low memory usage on local machines.
- **Vector Search:**
    - **Library:** FAISS (FlatL2 index)
    - **How Used:** Stores and searches embeddings for instant retrieval of relevant chunks from multiple documents.
- **Language Model:**
    - **Type:** Llama 3.2 (via Ollama)
    - **Parameters:** 8 Billion of parameters, supports nuanced, human-like responses in multiple languages.

    - **How Used:** Synthesizes answers from retrieved document context, generates natural language responses, and always cites sources.
- **Prompt Engineering:**
    - Prompts instruct the LLM to synthesize information from all retrieved document chunks, answer in the user's language, and always cite sources for transparency.

Tech Stack

- **Python Libraries:**
  streamlit, sentence-transformers, optimum[onnxruntime], onnxruntime, transformers, faiss-cpu, pytesseract, Pillow, PyPDF2, python-docx, langdetect
- **External Tools:**
  Tesseract OCR, Ollama (for Llama 3.2), Vosk (for speech recognition)
- **ONNX Model:**
  Quantized multilingual SentenceTransformer (paraphrase-multilingual-MiniLM-L12-v2)
- **Vector Search:**
  FAISS (FlatL2 index)
- **NER:**
  spaCy or custom entity extraction

How to Run

1. **Prerequisites**
    - Python 3.8+
    - Tesseract OCR (for image/PDF text extraction)
    - Ollama with the Llama 3.2 model (ollama pull llama3.2)
    - Vosk model for speech recognition
2. **Install dependencies**

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

3. **Model setup**
   - Place the Vosk model in **src/models/vosk-model-small-en-us-0.15/**
   - Place the ONNX embedding model in **src/models/onnx_model_quantized/**
4. **Configuration**
   - Create a .env file (if needed) with:

     ```
     TESSERACT_CMD_PATH=C:/Program Files/Tesseract-OCR/tesseract.exe
     ONNX_MODEL_PATH=src/models/onnx_model_quantized
     ```
5. **Start the assistant**

   ```
   streamlit run src/app.py
   ```

Usage

1. Upload your policy documents (PDF, DOCX, images, etc.)
2. Ask your question—type it or use the voice button
3. Get instant answers—with references to the exact document and page

Project Structure

```
AIPolicyAssistant/
├── src/          # Application source code
|    ├── backend/
|    ├── models/
|    └── tests/
├── docs/         # Documentation
├── scripts/      # Installation scripts
├── data/         # Sample documents
├── requirements.txt
├── README.md
└── .gitignore
```

Security Measures

- All processing is local: No data leaves your machine or company network.
- No cloud dependencies: All models and vector stores are local.
- Logs are local and can be encrypted.

Testing & Validation

- **Unit Tests:** For text extraction, chunking, embedding, retrieval, and logging.
- **Manual Tests:** Upload documents/images in various languages, ask questions, and verify answer correctness and language.

Troubleshooting

- **OCR issues:** Double-check the Tesseract install and .env path.
- **Model errors:** Ensure ONNX and Vosk models are in src/models/.
- **Voice recognition:** Grant microphone permissions to your terminal/app.
- **Ollama errors:** Make sure Ollama is running and the Llama 3.2 model is pulled.
- **File errors:** Ensure all folders and models are correctly placed as per the structure above.

Team AInsteins:

- Ananya Mehta
- Anshika Ranjan
- Eshita Khare

Contact:
Ananya Mehta
Email: amehta@parkar.digital

Anshika Ranjan
Email: aranjan@parkar.digital

Eshita Khare
Email: ekhare@parkar.digital

Built for secure, instant, and intelligent policy answers.