AI Chatbot for Equipment Troubleshooting

This presentation outlines the development and capabilities of an offline AI chatbot designed to revolutionize industrial equipment troubleshooting. We will delve into its architecture, features, and the significant benefits it offers to engineering professionals.

Project Overview

Offline AI Chatbot for Equipment Troubleshooting

Project Name	Offline AI Chatbot for Equipment Troubleshooting
Presenter	Jocichan Joseph / Amith AK
Affiliation	[AJ Robotics
Date	2025

🎯 Core Objective: Empowering Technicians

Project Goal

Our primary objective is to develop and deploy an intelligent, robust assistant that significantly enhances the efficiency of industrial equipment troubleshooting. This solution focuses on providing immediate, accurate, and secure technical support.

Provide an **intelligent assistant** to help technicians troubleshoot industrial equipment directly on-site, minimizing delays.

Offer **offline, secure**, and **contextual support** by leveraging local documents, ensuring data privacy and operational continuity without internet access.

Reduce equipment **downtime** by providing rapid diagnostic assistance and decrease reliance on external OEM support channels.

Key Capabilities: Enabling Autonomous Troubleshooting

Feature Set

The chatbot integrates several crucial features to deliver a comprehensive and user-friendly troubleshooting experience, focusing on local processing and data security.

Offline Local AI (LLM)

Utilizes LM Studio or other compatible local LLM for secure, internet-free operation.

ChatGPT-style Interface

Built with Streamlit for an intuitive and familiar conversational user experience.

PDF Manual Ingestion

Processes and indexes equipment manuals for context-aware responses and deep knowledge retrieval.

JSON Fault Database Support

Integrates structured data from known issues and their resolutions for precise problem-solving.

Live Streaming Response

Provides immediate, real-time responses from the local LLM, enhancing interactivity.

Session-based Chat History

Maintains conversational context for an uninterrupted user experience.

Reset & Re-upload Capability

Allows users to clear sessions and update knowledge bases as needed.

Domain-specific Prompt Tuning

Optimizes LLM responses for accuracy and relevance within industrial contexts.

⚙️ System Architecture: Local & Integrated

Architectural Overview

The chatbot's architecture is designed for robust offline performance, leveraging local AI models and efficient data processing components.

Core Components:

Local LLM API: (via LM Studio) for all AI reasoning.

PDF Parser: (PyPDF2) for extracting text from manuals.

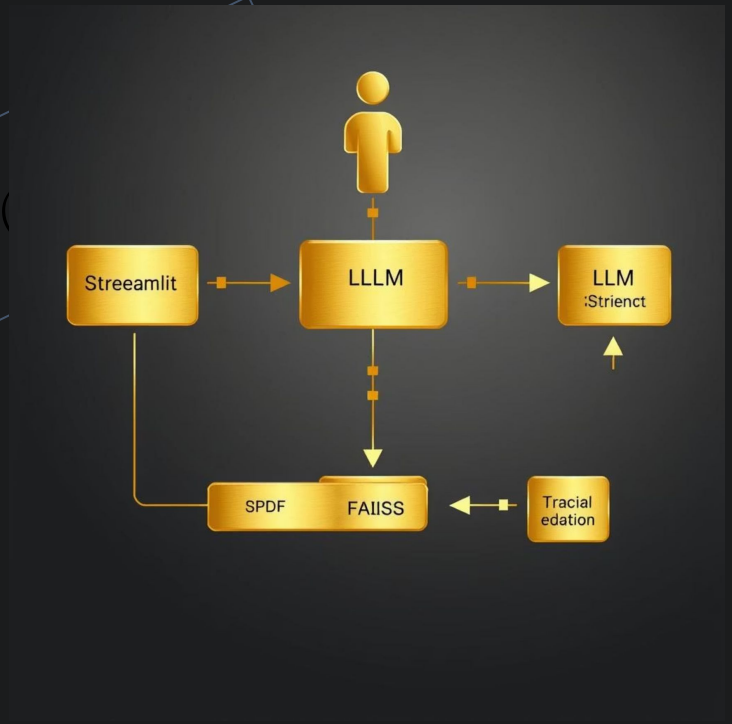
Embedding Model: (MiniLM via sentence-transformers) to convert text into searchable vectors.

FAISS Index: For rapid semantic search and context retrieval.

Streamlit UI: The interactive user interface.

JSON Ingestion: Handles structured fault data for direct query.

Simplified Diagram:



🧩 How It Works: Step-by-Step Troubleshooting

Operational Flow

The chatbot's workflow is streamlined for ease of use, ensuring technicians can quickly access relevant information for diagnosis and repair.

1. Upload Manual (PDF)

PDF manuals are automatically chunked into smaller sections and converted into numerical embeddings, ready for semantic search.

2. Upload Known Faults (JSON)

Structured JSON data, containing known issues and their resolutions, is ingested and stored in memory for direct lookups.

3. Ask a Question

The user's query triggers a search across both the PDF context (FAISS index) and the JSON fault database. Relevant snippets are retrieved and used to format a prompt for the local LLM.

4. Response Streaming Live

The LLM generates a response based on the contextualized prompt, which is then streamed live to the user through the intuitive ChatGPT-style interface for immediate feedback.

🖥️ System Requirements: Ensuring Optimal Performance

Technical Specifications

To ensure seamless operation and efficient local processing, the AI chatbot requires specific software dependencies and minimal hardware specifications.

Software Requirements:

Python 3.8+: The foundational programming environment.

LM Studio (or compatible LLM API server): For hosting the chosen local large language model.

- **Key Python Dependencies:**

streamlit: For the web application framework.

sentence-transformers: For generating text embeddings.

faiss-cpu: For efficient similarity search.

PyPDF2: For PDF parsing capabilities.

requests: For HTTP communication with the LLM API.

dotenv: For managing environment variables securely.

Hardware Requirements:

CPU: 4-core processor or better for adequate processing speed.

RAM: 8GB minimum, with 16GB recommended for larger LLMs or document sets.

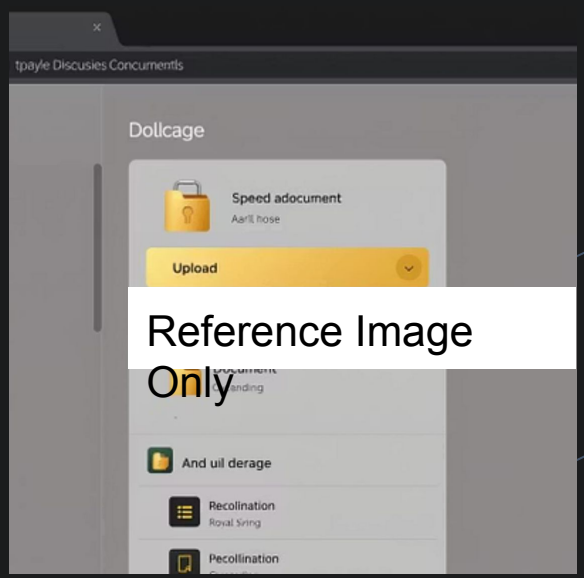
Disk Space: Approximately 500MB free disk space for application files and models.

GPU (Optional): A dedicated GPU with VRAM is highly recommended for faster LLM inference, significantly reducing response times.

💻 Demo Screenshots: A Glimpse into the Interface

User Interface

The user interface is designed for clarity and ease of navigation, providing a familiar chat experience with integrated document management.



These screenshots illustrate the intuitive Streamlit interface, showcasing the document upload sidebar and the interactive chat window with contextualized responses.

✅ Key Benefits: Revolutionizing Maintenance Operations

Impact & Value

The offline AI chatbot delivers substantial advantages across various stakeholders, enhancing efficiency, security, and operational autonomy.

For Technicians

Fast Answers: Instant access to diagnostic and repair information.

No Internet Needed: Operates autonomously in remote or secure environments.

Contextual Responses: Provides highly relevant advice based on specific equipment and faults.

For Maintenance Teams

Reduced Support Calls: Decreases reliance on central support desks.

Less Downtime: Expedites troubleshooting, getting equipment back online faster.

Supports Local Docs/Data: Utilizes internal knowledge bases securely.

For OEMs/Manufacturers

Smarter Service Enablement: Enhances field service capabilities.

Better Field Team Tools: Equips technicians with advanced on-site resources.

No Cloud Data Privacy Concern: Ensures sensitive operational data remains on-premises.

🧪 Future Enhancements: Expanding Capabilities

Roadmap

Our development roadmap includes several enhancements aimed at further improving the chatbot's utility and integration within industrial environments.

Voice-to-Text Input

Enabling hands-free interaction, crucial for technicians working in demanding environments.

Image Upload (e.g., Faults on Screen)

Allowing visual diagnosis by enabling users to upload photos of error codes or physical damage for analysis.

CSV/Excel Data Support

Expanding data ingestion to include structured operational data logs or maintenance records for richer context.

Role-based Access (Operator vs Engineer)

Implementing permissions to tailor information access based on user roles and expertise levels.

Integration with MES/SCADA Logs

Connecting to manufacturing execution systems (MES) or supervisory control and data acquisition (SCADA) for real-time operational data context.