Ingenzi Tech | Admin Guide

This document serves as a comprehensive guide for administrators managing the LLM-Powered Philips HDI 5000 Ultrasound Machine Troubleshooting Tool. Follow the steps and guidelines carefully to configure, maintain, and troubleshoot the system effectively.

Table of Contents

---

1. System Overview

The system is a multi-component application designed to assist with troubleshooting the Philips HDI 5000 ultrasound machine. The components include:

- **React Frontend**: A dashboard for users to interact with the system.
- **Flask Backend**: Handles API requests and backend logic.
- **Chainlit Copilot**: An AI-powered assistant accessible via a widget or standalone app.
- **Feedback App**: Allows admins to manage and analyze user feedback.

---

2. Prerequisites

Ensure the following are available and installed:

1. **Software Requirements**:
   - Python (v3.8 or higher)
   - Node.js and npm
   - Chainlit (latest version)
   - Flask and required dependencies (via `requirements.txt`)
2. **Hardware Requirements**:
   - Minimum 8GB RAM
   - At least 2 CPU cores
   - Stable internet connection for API integrations
3. **Accounts and API Access**:
   - Google Cloud account with Calendar API enabled
   - GitHub account for managing repositories
   - OpenAI and Tavily accounts for API keys

---

3. Setup and Configuration

Environment Setup

1. Clone the repository:

```
Unset
git clone https://github.com/ahmedinhotahiru/ingenzi_tech.git

cd ingenzi_tech
```

2. Create and activate python virtual environment

```
Unset
python -m venv capstone_venv

capstone_venv\Scripts\Activate.ps1
```

3. Install Python dependencies:

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

4. Install Node.js dependencies for the frontend:

```
Unset

cd ultrasound-dashboard

npm install
```

API Key Management

The system requires various API keys for external services. Follow these steps:

1. **Google Calendar API**:
   - Navigate to the Google Cloud Console.
   - Enable the Calendar API and generate OAuth 2.0 Client IDs.
   - Replace placeholders in `ultrasound-dashboard/Home.js`:

```
Unset
const CLIENT_ID = 'your-client-id.apps.googleusercontent.com';

const API_KEY = 'your-api-key';
```

2. **GitHub Token**:
   - Create a personal access token from GitHub Settings.
   - Replace the token in `backend/end_points.py`:

```
Unset
g = Github("your-personal-access-token")
```

3. **OpenAI and Tavily API Keys**:
    - Generate keys from OpenAI and Tavily.
    - Replace placeholders in `chat_ultrasound_chroma.py`:

```
Unset
os.environ["TAVILY_API_KEY"] = 'your-tavily-api-key'

embeddings =
OpenAIEmbeddings(openai_api_key="your-openai-api-key")
```

Server Configuration

If any port is occupied, manually update the port configurations:

- React Frontend: Modify the `npm start` command.
- Flask Backend: Update the `app.run` port in `backend/end_points.py`.
- Chainlit Copilot: Adjust the `chainlit run` command.

4. Running the Application

1. **Start the React Frontend**:
    - Navigate to `ultrasound-dashboard` and run:

```
Unset
npm start
```

Screenshot of running app below:

```
Windows PowerShell
Compiled successfully!

You can now view ultrasound-dashboard in the browser.

  Local:            http://localhost:3000
  On Your Network:  http://192.168.1.70:3000

Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully
```

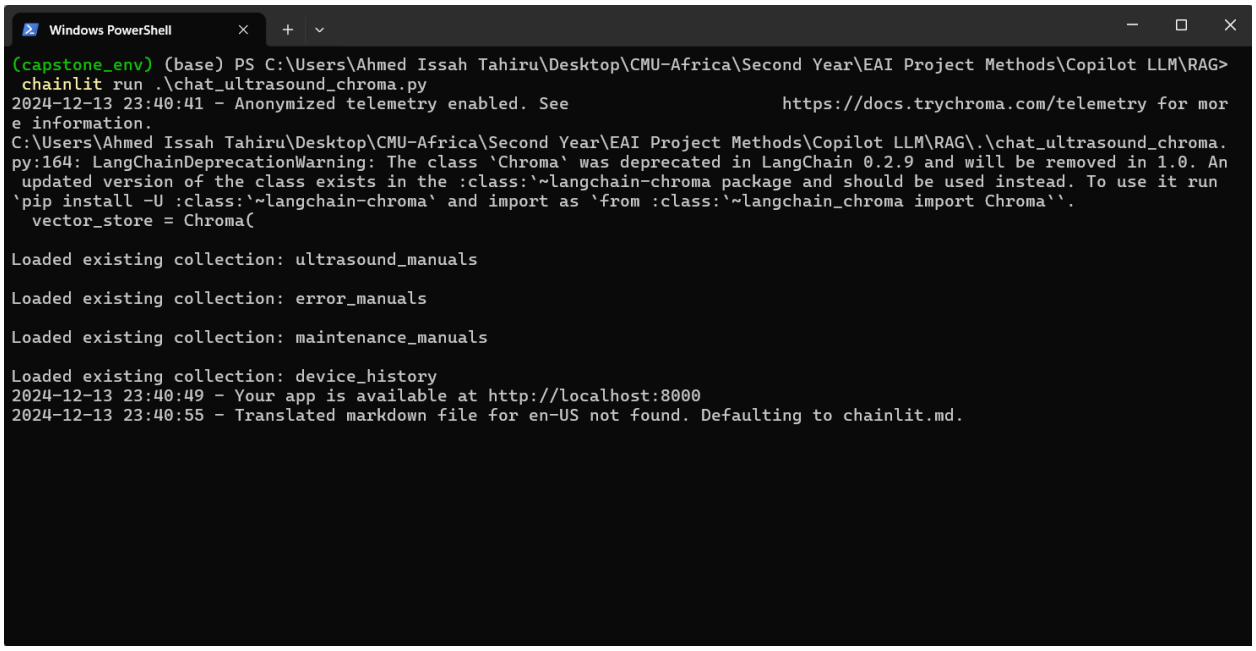
2. **Start the Flask Backend**:
   - Navigate to `backend` and run:

```
Unset
python .\end_points.py
```

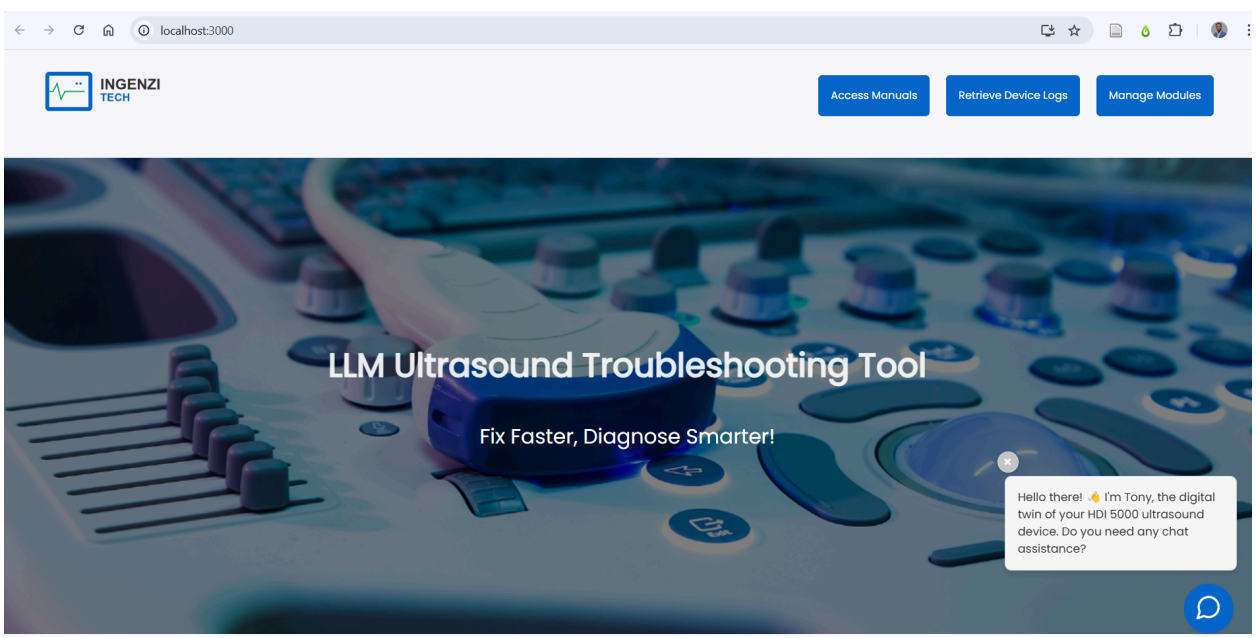
```
Windows PowerShell
(capstone_env) (base) PS C:\Users\Ahmed Issah Tahiru\Desktop\CMU-Africa\Second Year\EAI Project Methods\Copilot LLM\back
end> python .\end_points.py
 * Serving Flask app 'end_points'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 941-580-033
127.0.0.1 - - [13/Dec/2024 23:39:51] "GET /api/last-service-date HTTP/1.1" 200 -
127.0.0.1 - - [13/Dec/2024 23:39:51] "GET /api/last-service-date HTTP/1.1" 200 -
```

3. **Start the Chainlit Copilot**:
    - Navigate to RAG and run:

```
Unset
chainlit run .\chat_ultrasound_chroma.py
```



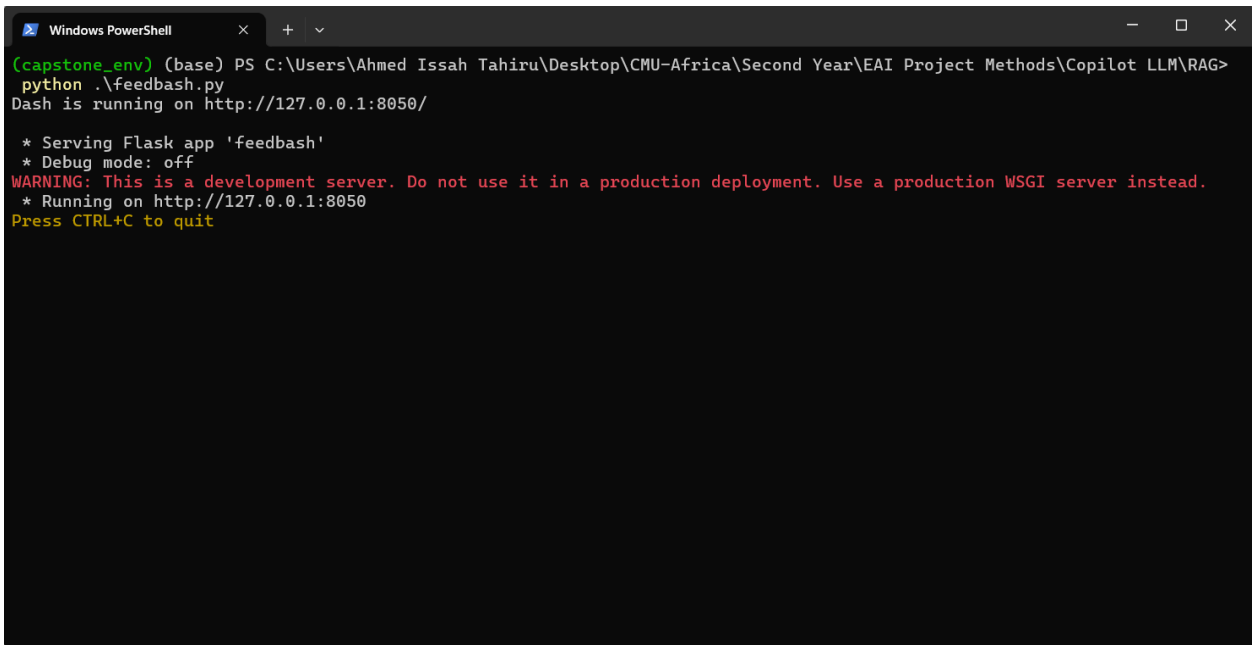
The copilot now runs on top of the react application as a chat widget shown below:



4. **Start the Feedback App**:
   - Navigate to `RAG` and run:

```
Unset
python feedback_app.py
```

Screenshot of running feedback shown below:



The running feedback app looks like this:

Ingenzi Feedback

ID	User Prompt	Feedback	Helpful
11c1bbc3-c77a-4dfc-8bb9-b67c1ae561e3	i have error 0060	This is the correct error code description for error 0060	
2b7599ef-409c-4b24-af3d-db8dbf93c0ba	the image is not clear when i scan	This was a great resolution to improving scan quality	
cc954d17-5243-4b29-9c23-2d1b650056ff	retrieve device logs	Worked great	
7f6c3986-d161-4ce1-839a-22f19902272b	give me best practices	Too generic response	

5. Maintenance and Monitoring

1. **Logs**:
    - Monitor logs for each component to identify issues.
    - Flask logs: `backend/logs`
    - React logs: Displayed in the terminal during `npm start`.
2. **API Quotas**:
    - Monitor usage limits for Google, OpenAI, and Tavily APIs.
3. **Updating Dependencies**:
    - Periodically update Python and Node.js dependencies:

```
Unset

pip install --upgrade -r requirements.txt

npm update
```

6. Troubleshooting Common Issues

Issue	Possible Cause	Solution
Frontend not loading	React server not running	Ensure `npm start` is executed.
Backend API not responding	Flask server not running	Check logs and rerun `end_points.py`.
Chainlit copilot not accessible	Port conflict	Change the port and rerun.
Feedback app not opening	Port `8050` is occupied	Change the port in `feedback_app.py`.
API key authorization errors	Invalid or expired keys	Regenerate keys and update configuration files.

7. Best Practices

1. **Security**:
    - Store sensitive keys in environment variables or `.env` files.
    - Avoid committing API keys to version control.
2. **Backups**:
    - Regularly back up critical files and databases.
3. **Documentation**:
    - Maintain updated documentation for all custom configurations.