Week 2 Report: Hydraulic Assistant

Project Overview

- This week's primary achievement is a functional prototype of the Flutter-based
 "Hydraulic Assistant," a chat application that answers domain-specific questions using a RAG (Retrieval-Augmented Generation) pipeline.
- Created a cross-platform app (Android, iOS, web, with desktop scaffolding present).
- Initialized Firebase and included authentication screens.

Data and Knowledge

- Established the core knowledge base by embedding domain-specific PDFs, making them accessible to the application for analysis.
- Implemented a simple embedding fallback in embedding_service.dart to generate 384-dimensional vectors when an external model is unavailable.
- Integrated with Pinecone vector database for upserting and querying document embeddings.

LLM and Retrieval

- Implemented the **PineconeGrokService**:
 - Manages upsert/query operations to Pinecone via REST API, using keys and URLs from environment variables.
 - Integrated GROQ for chat completions.
 - Supports streaming completions (SSE) with askGroqStream to yield tokens incrementally.
- These components were integrated to create a complete, end-to-end "answerUserQuery" pipeline: the user's query is embedded, relevant context is retrieved from Pinecone, and both are sent to GROQ to generate an informed answer.

Chat UI/UX

• ChatPage Features:

- Streaming assistant responses with a slow, professional typewriter effect.
- Enhanced the user experience with intelligent auto-scrolling, which only activates when the user is near the bottom of the chat, preventing jarring screen jumps while they are reading previous messages.
- Increased message font size for better readability and maintained a consistent red/white theme.
- Added a top-right "Clear chat history" action with a confirmation dialog and snackbar feedback.
- Included a network diagnostics dialog that runs quick tests for internet, Pinecone, and GROQ connectivity.

State and Persistence

- Implemented chat history persistence using shared preferences.
- Messages are saved after each send/receive cycle.
- Chat history is restored on page load, ensuring the conversation remains intact after tab changes or app restarts.

Configuration and Security

- Managed environment variables using flutter dotenv.
 - Created a .env file for PINECONE_API_KEY, PINECONE_BASE_URL, GROQ_API_KEY, and GROQ_BASE_URL.
 - o Developed an EnvConfig wrapper for typed access and validation.
 - o Documented and tracked .env.example; .env is ignored via .gitignore.
- Scrubbed all hardcoded secrets from the codebase and provided guidance on key rotation.
- Created ENV_SETUP.md with instructions for setting up environment files.

Testing and Diagnostics

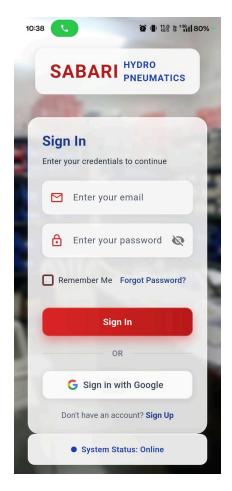
- Developed test_chat_integration.dart to validate embeddings, Pinecone queries, the GROQ API, and the full integration.
- Created network_test.dart for basic connectivity checks against Google, Pinecone, and GROQ.

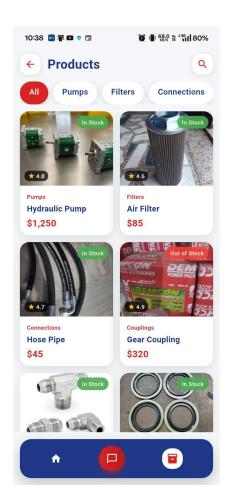
Notable Files Touched

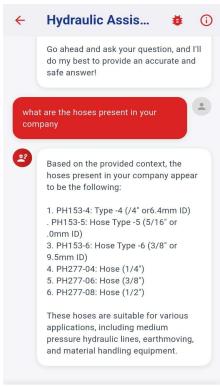
- **lib/chat_page.dart**: Implemented streaming/typewriter effect, persistence, clear history, font size changes, and scrolling fixes.
- **lib/pinecone_grok_service.dart**: Handled Pinecone + GROQ integration with the streaming API.
- **lib/embedding_service.dart**: Added simple embedding fallback and environment variable usage.
- **lib/config/env_config.dart**: Centralized environment loader and getters.
- lib/main.dart: Loads environment variables on startup.
- **pubspec.yaml**: Added flutter_dotenv, shared_preferences, and ensured assets were included.

Current Status

The week concludes with the delivery of a functional, end-to-end RAG chat application for hydraulic topics. The prototype features persistent chat history, secure API key management, and a polished, streaming user experience.







Ask about hydraulic hose pressure...

