

PRESSURE WASHER EQUIPMENT SUPPORT AI

Task List & Project Roadmap

Last Updated: February 19, 2026

COMPLETED

Phase 1 — Static HTML Troubleshooting App

- ✓ Built initial troubleshooting web page with keyword search
- ✓ Extracted data from 9 General Pump PDFs (Troubleshooting, ServiceManual, 47 Series, KFMZ Owner Manual, Oil Recommendations, TS2021 Data Sheet, MK Repair Manual, 2018 Catalog)
- ✓ Created interactive UI with sidebar navigation and section pages
- ✓ Added General Pump product catalog (372 pump models across 20+ series)
- ✓ Added 851 accessories across 35 categories
- ✓ Added oil recommendation lookup by pump model
- ✓ Added service procedures (valve, plunger, V-packing)
- ✓ Added maintenance schedules and checklists
- ✓ Added system design reference (motor sizing, pulleys, unloaders, filters, spray tips)
- ✓ Added warranty info section
- ✓ OCR support for scanned/image-based PDFs (Tesseract)

Phase 2 — Alkota Integration

- ✓ Extracted data from 11 Alkota equipment manuals
- ✓ Added 91 unique troubleshooting cause/remedy pairs from Alkota docs
- ✓ Added Alkota-specific sections: maintenance, installation, start-up/shutdown
- ✓ Integrated Alkota Service School Book (245 pages — pumps, unloaders, electrical, coils, burners, VFS, radiant heaters, warranty)
- ✓ Added collapsible brand sections (General Pump / Alkota) in sidebar
- ✓ Brand badges (GP / ALK) throughout UI and search results
- ✓ Cross-brand search (queries return results from both brands)

Phase 3 — Architecture Upgrade (HTML + JS Data File)

- ✓ Split app into index.html (UI) + knowledge-base.js (data)
- ✓ Built build-kb.py script to regenerate knowledge base from PDFs
- ✓ Supports both text-based and scanned PDFs
- ✓ Redesigned UI with light theme and expandable/collapsible sections

Phase 4 — SQLite FTS5 Database Pipeline

- ✓ Built ingest_pdfs.py — PDF ingestion pipeline with SQLite FTS5
- ✓ Recursive folder scanning for PDFs
- ✓ Intelligent chunking (section headers, paragraphs, sentences — ~800 word max, 100 word overlap)
- ✓ Auto brand detection (General Pump, Cat Pumps, Alkota, Honda, Briggs & Stratton, AR, Comet, U dor, Giant, Karcher)
- ✓ Auto document type detection (service manual, owner manual, parts list, data sheet, troubleshooting, catalog, installation, safety)
- ✓ Auto chunk type classification (text, troubleshooting, specs, maintenance, procedure, safety, parts, table)
- ✓ SHA256 file deduplication (safe to re-run without reprocessing)
- ✓ FTS5 with Porter stemming and Unicode tokenizer
- ✓ Auto-sync triggers between chunks table and FTS index
- ✓ CLI: ingest, search, model lookup, stats, brand filter, reset
- ✓ Table extraction alongside text
- ✓ Tested against Cat Pumps 67DX, Honda GX120/GX160, Briggs & Stratton docs

Phase 5 — Search API + AI Agent

- ✓ Built search_api.py — Flask REST API
- ✓ CORS enabled for React frontend
- ✓ Direct FTS5 search endpoint (GET/POST /api/search)
- ✓ Model number lookup endpoint (/api/model/<number>)
- ✓ Stats, brands, documents, health check endpoints
- ✓ Claude AI integration via Anthropic API (POST /api/ask)
- ✓ Smart web search fallback: if <3 local chunks found, enables Claude web search tool
- ✓ System prompt tuned for pressure washer/pump equipment support
- ✓ Conversation history support for multi-turn chat
- ✓ SSE streaming support for real-time chat display
- ✓ Contact info suppression rule (no phone/fax/email/address/URLs from docs)
- ✓ Auto-installs missing Python packages

Phase 5.5 — Data Collection

- ✓ Organized 800MB+ of PDFs into G:\psupp_info with 8 subfolders (Alkota, burners, engines, Gpump, Installation_and_Service, pumps, Resource_Information, Technical_Data)
- ✓ Built BUILD_PROMPT.txt — single-prompt recreation document
- IN PROGRESS: Running full ingestion of G:\psupp_info into knowledge.db

IMMEDIATE NEXT STEPS

Ingestion Completion

- Complete 800MB PDF ingestion and verify results
- Review stats: total docs, chunks, brands detected, any failures

- Identify scanned/image PDFs that need OCR and add OCR support to pipeline
- Fix any misdetected brands or document types
- Test search quality across all brands and document types

Wire Up Frontend

- Connect existing React chat UI to search_api.py endpoints
- Implement /api/ask calls from chat input
- Display AI responses with source citations (document name, page, brand)
- Show visual indicator for web search vs local KB answers
- Implement SSE streaming for real-time response display
- Add brand filter dropdown (populated from /api/brands)

API Key Setup

- Set ANTHROPIC_API_KEY environment variable on dev machine
- Test AI-powered Q&A end-to-end
- Test web search fallback with queries not in local KB

SHORT-TERM IMPROVEMENTS

Search Quality

- Add synonym handling (e.g., "leaking" matches "leak", "water dripping" matches "water leak")
- Add model number normalization (e.g., "TS 2021" matches "TS2021")
- Improve chunking for troubleshooting tables (problem/cause/solution as linked triples)
- Add boost weights: troubleshooting chunks rank higher for diagnostic queries
- Test and tune WEB_SEARCH_THRESHOLD (currently 3 chunks)

OCR Pipeline

- Add Tesseract OCR fallback for scanned/image PDFs in ingest_pdfs.py
- Auto-detect image-only pages and route to OCR
- Test OCR quality on actual scanned manuals from the 800MB corpus

Data Quality

- Review and expand brand detection patterns for any missed brands
- Add more document type patterns if needed
- Handle multi-language documents (some Alkota manuals have Spanish sections)
- Strip duplicate content (headers/footers repeated on every page)
- Improve title extraction for documents without clear titles

UI/UX

- Redesign React chat UI for the new API-powered architecture
- Add "Sources" panel showing which documents were referenced
- Add quick-action buttons (common queries by brand)
- Mobile-responsive design for shop floor tablet use
- Dark mode option
- Add loading/thinking animation during AI responses

MEDIUM-TERM FEATURES

Semantic Search (Vector Embeddings)

- Add embedding generation to ingestion pipeline (local model or API)
- Store vectors in SQLite using sqlite-vec extension or separate FAISS index
- Hybrid search: combine FTS5 keyword results + vector similarity results
- Re-rank combined results before sending to Claude

Conversation Memory

- Store chat history per session in SQLite
- Pass relevant history to Claude for context continuity
- Allow users to reference previous answers ("you mentioned earlier...")

Service Ticket System

- Add ability to create service tickets from chat
- Capture: equipment model, symptoms, diagnostic steps taken, recommended parts
- Export tickets as PDF or email

Parts Lookup

- Extract part numbers and cross-references from all manuals
- Build dedicated parts search (by part number, pump model, or description)
- Link parts to service procedures ("you'll need part #X for this repair")

Multi-User / Deployment

- Add user authentication (optional)
- Deploy to company intranet or cloud server
- Add rate limiting for API calls
- Usage logging and analytics (what are users asking about most?)
- Admin panel to re-run ingestion, view stats, manage documents

LONG-TERM VISION

Offline Mode

- Bundle SQLite database with frontend (using sql.js) for fully offline use
- Sync mechanism when connection is available
- Progressive Web App (PWA) for tablet installation

Image/Diagram Support

- Extract diagrams and exploded views from PDFs
- Link images to relevant procedures and parts
- Visual part identification ("point to the part on the diagram")

Voice Interface

- Speech-to-text for hands-free queries in the shop
- Text-to-speech for reading back procedures while working

Training Mode

- Interactive quizzes based on manual content
- Step-by-step guided repair walkthroughs
- Certification tracking for technicians

FILES REFERENCE

File	Purpose	Status
ingest_pdfs.py	PDF to SQLite FTS5 ingestion pipeline	Complete
search_api.py	Flask API + Claude AI + web search	Complete
knowledge.db	SQLite FTS5 database	Demo (3 docs)
BUILD_PROMPT.txt	Single-prompt app recreation document	Complete
index.html	Static HTML troubleshooting app (v1)	Legacy
knowledge-base.js	JS data file for HTML app (v1)	Legacy
build-kb.py	PDF to JS knowledge base builder (v1)	Legacy
React frontend	Chat UI connected to API	To build