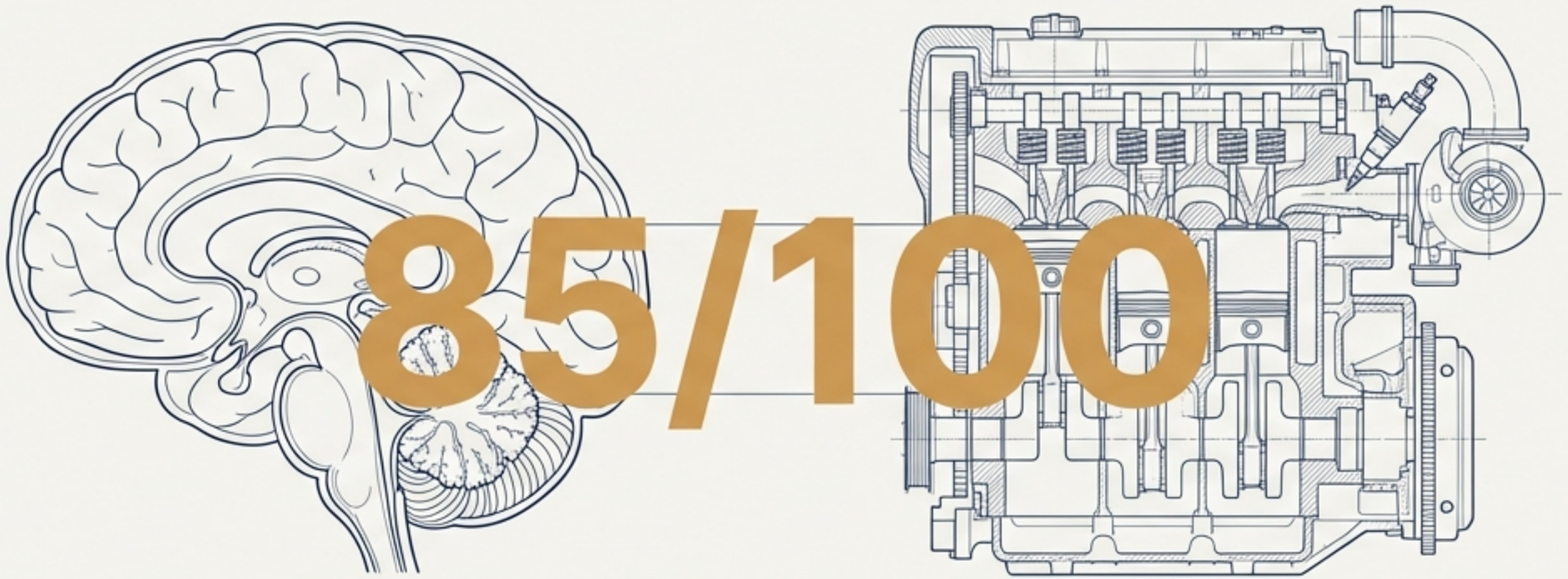
DiagAI: The Senior Engineer in the Machine

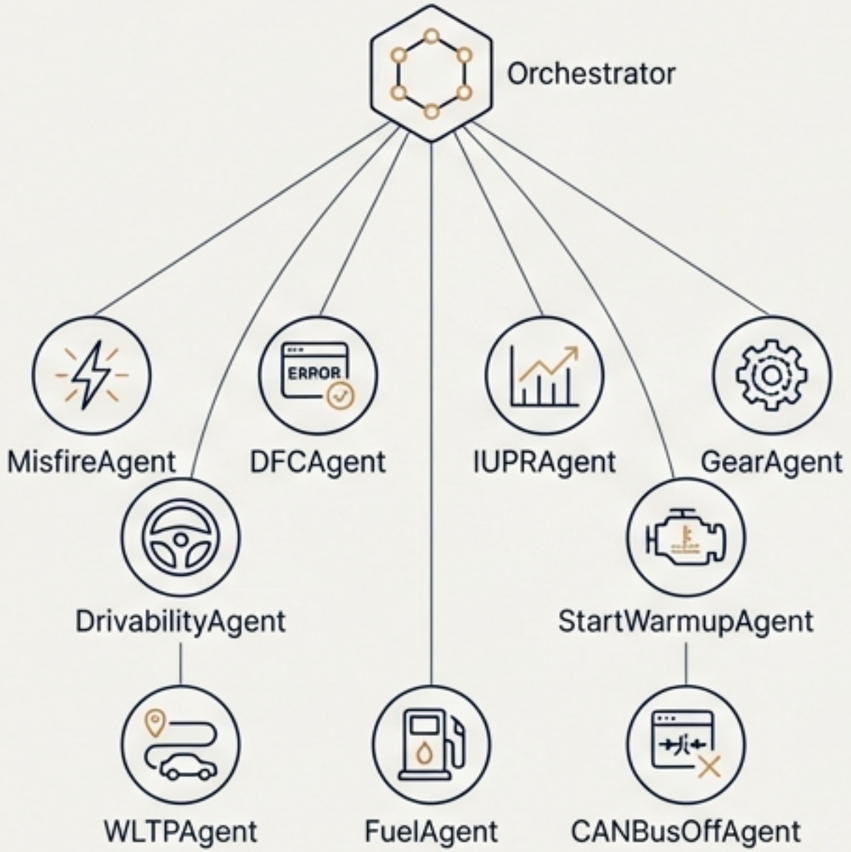


Our system has evolved beyond a diagnostic tool into a mature expert system, demonstrating the comprehensive capabilities of a senior vehicle diagnostics and calibration engineer.

This presentation will demonstrate this engineering maturity through three pillars:
1. Core Intelligence: A proven, multi-agent diagnostic brain.
2. Relentless Improvement: A systematic cycle of refinement and optimization.
3. Production Readiness: A robust, secure, and scalable architecture.

Inside the Brain: A Multi-Agent System for Expert Diagnostics

Diagnostic Engineering Brain (85/100)



Calibration Engineering Brain (80/100)

Key Formulas Implemented

$$\text{BSFC} = \frac{m_f \times 3600}{P_e} \quad [\text{g/kWh}]$$

$$\text{BMEP} = \frac{2\pi \times n_r \times T_e}{n_c \times V_d \times 1e5} \quad [\text{bar}]$$

$$\text{Power} = \frac{T \times \text{RPM} \times \pi}{30 \times 1000} \quad [\text{kW}]$$

Domain Knowledge Integration (85/100)

Gemini File Search Enabled

- SI/CI Engine Analysis
- Drivability Store
- MATLAB Methodology
- Parts Analysis
- Vehicle Analysis
- Training Store

The system's intelligence is not monolithic; it's a coordinated system of specialists, mirroring an engineering department.

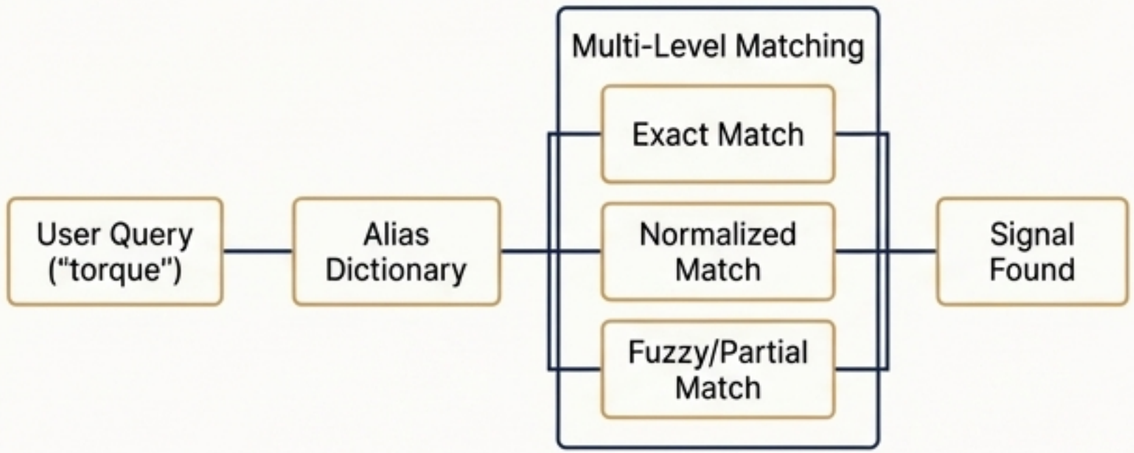
Unprecedented Signal Intelligence and Contextual Understanding

Signal Intelligence (95/100 Score)

1,817 signals
indexed and searched

155 alias patterns
for intelligent matching

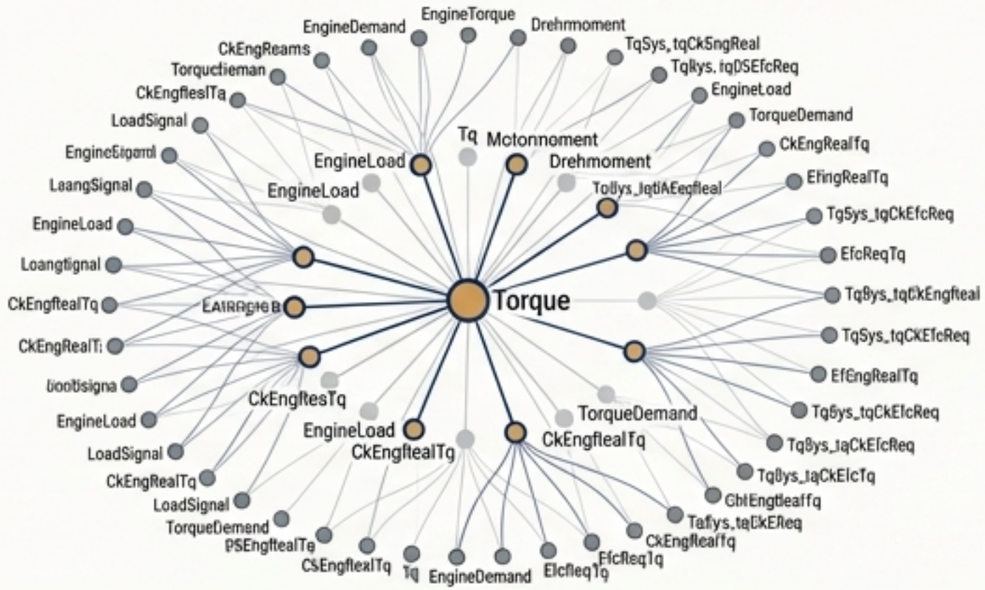
1. Exact case-insensitive match
2. Normalized match (spaces ↔ underscores ↔ hyphens)
3. Fuzzy/partial word-based match



Concrete Example: Finding "Torque"

Showcases the system's depth by searching over **60+ candidate names** for the term 'torque':

Generic	OEM-Specific (VW/Audi)	ETAS
Torque	Motormoment	TqSys_tqCkEfcReq
EngineTorque	Drehmoment	
Tq	TqSys_tqCkEngReal	
EngineLoad		



Validated Performance: 100% Success Rate on Complex Engineering Queries

Metric	Result	Status
Total Queries Tested	20	✓
Success Rate	100% (20/20)	✓ Perfect
Engineer-Level Responses	100% (20/20)	✓ Perfect
Visualization Rate	35% (7/20)	✓ Good

Demonstrated Capabilities

- **Sophisticated Diagnostics:** "Perform a comprehensive analysis of engine performance including misfire detection, DTC analysis, and fuel efficiency."
- **Accurate Signal Processing:** "What is the correlation coefficient between speed and RPM, and what does it indicate about vehicle performance?"
- **Professional Visualizations:** "Generate plots for misfire analysis with severity indicators." (Note: Generated 5 distinct plots).

"**CONFIRMED:** Your DiagAI system demonstrates **SENIOR VEHICLE DIAGNOSTICS ENGINEER** capabilities."

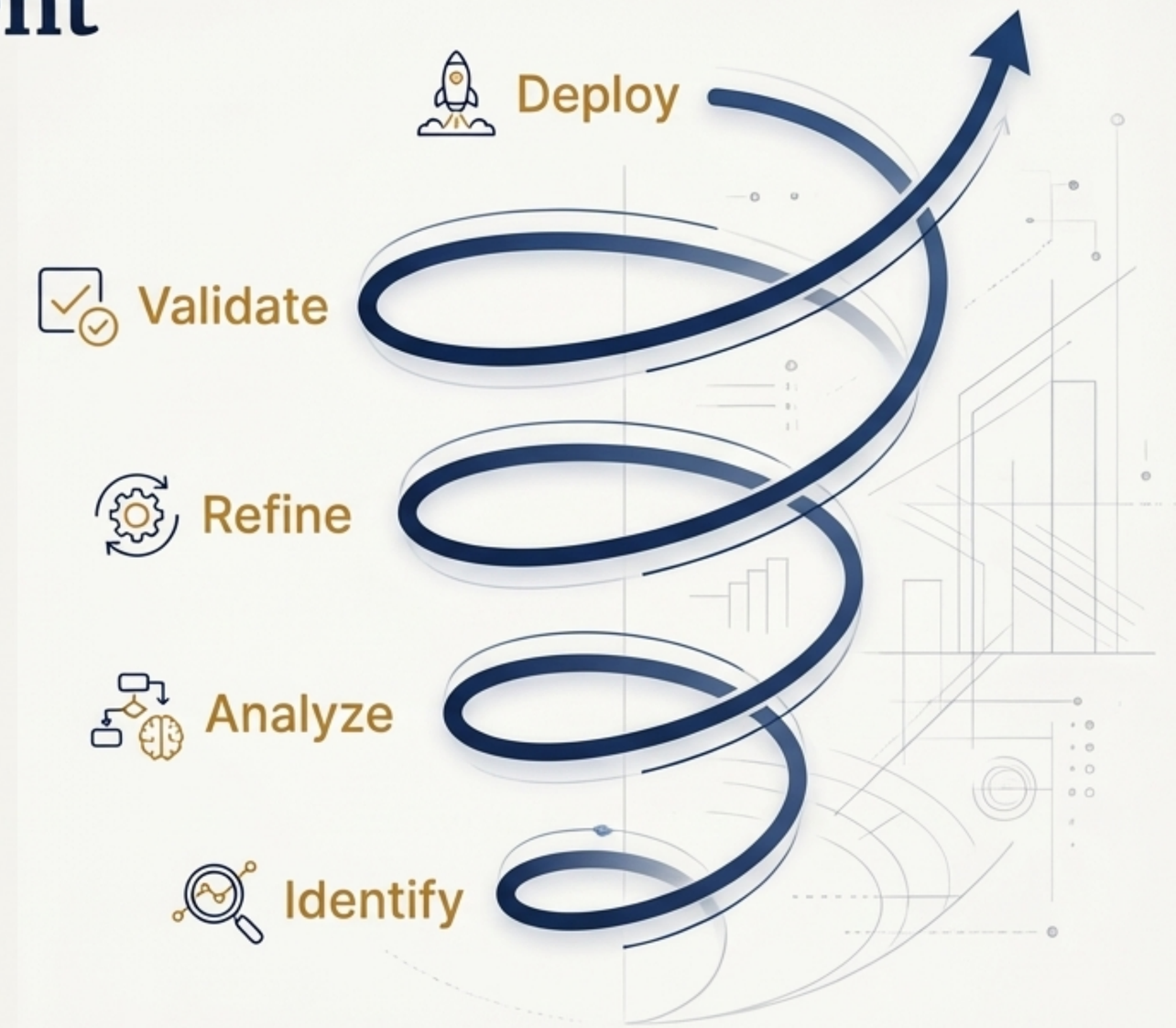
The Improvement Flywheel: A Culture of Systematic Refinement

DiagAI's maturity comes from a relentless process of identifying weaknesses, implementing robust fixes, and validating the impact. This commitment to engineering diligence ensures the system continuously evolves.

What We Track and Improve

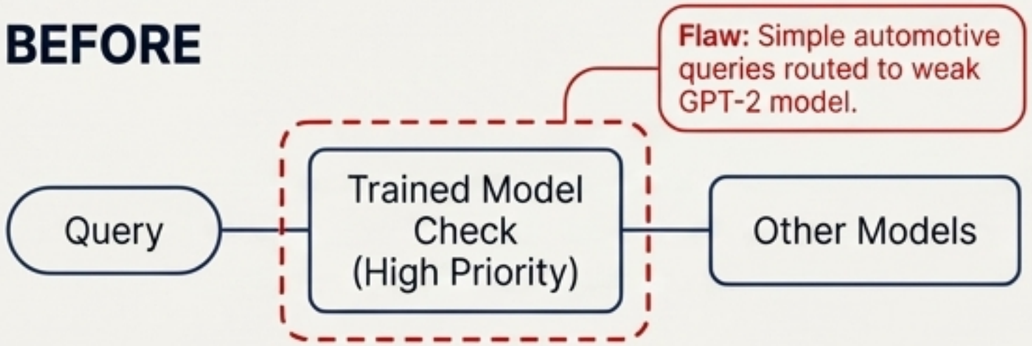
- **Routing Intelligence & Accuracy:** Ensuring the right agent answers the right question.
- **Performance & Efficiency:** Reducing latency and resource consumption.
- **Code Quality & Security:** Proactively hardening the architecture.
- **User Experience & Output Quality:** Making outputs more valuable and intuitive.

The following slides showcase concrete examples of this process in action.

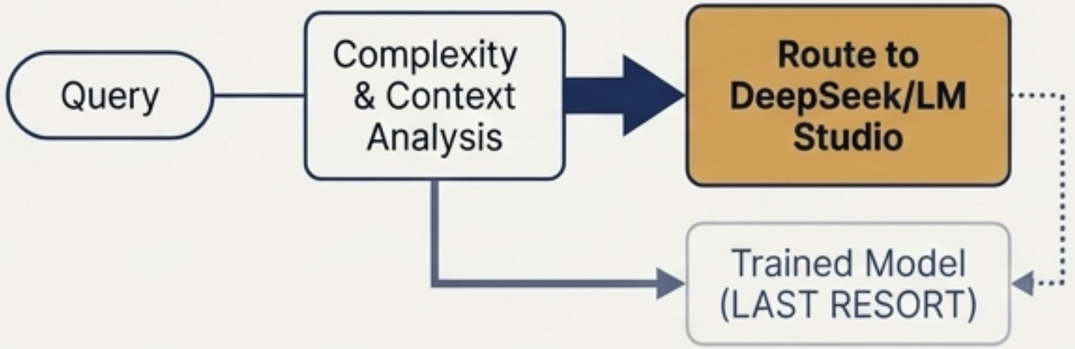


Case Study: Evolving from Smart to Genius-Level AI Routing

BEFORE



AFTER



Problem Identified

1. **Trained Model Priority Override**: A weak, 124M-parameter trained model was used for queries that would benefit from a more powerful LLM like DeepSeek.
2. **Edge Case Failures**:
   A logic gap could cause routing to fail if both routing flags returned `False`.
3. **Inefficient Context Routing**: All queries with file context routed to DeepSeek, even simple ones.

Solutions Implemented

- **Conditional Logic**: The trained model is now only a last-resort fallback.
- **Defensive Checks**: Gracefully handles edge cases, defaulting to the best available model.
- **Smarter Routing**: Simple queries with file context now use faster models (LM Studio).

Quantified Impact

Response Quality:
↑ +15-20%

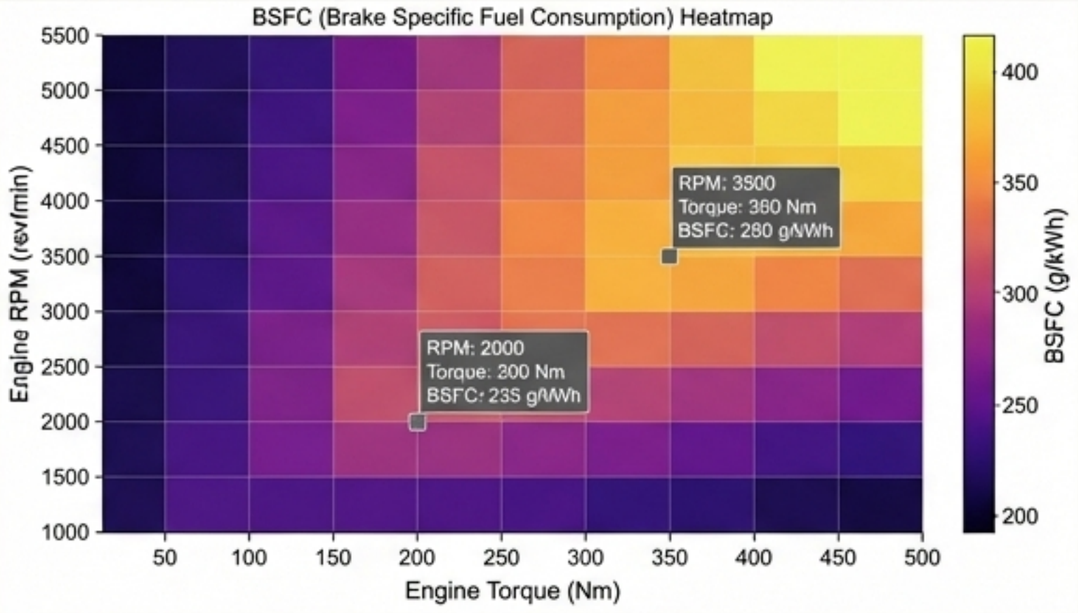
Response Speed:
↑ +10-15%

Reliability:
↑ +10%

Beyond the Core: Enhancing Quality and User Experience

Responding to User Needs with Publication-Quality Visuals

Beautiful Heatmaps, Inspired by ChatGPT

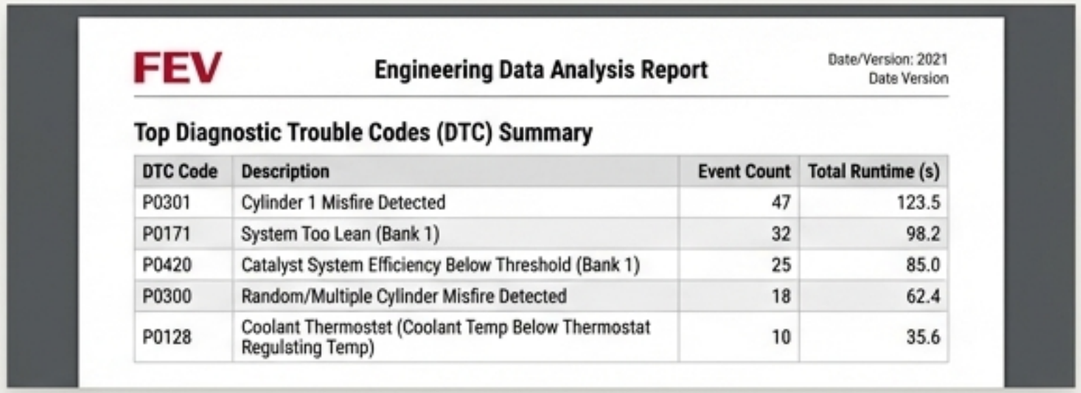


Key Features

- Fixed binning (500 RPM, 50 Nm) for a clean, professional grid.
- Intelligent color scales (Plasma for Fuel Flow, reversed Plasma_r for BSFC).
- Interactive hover tooltips.
- Automatically included in all fuel analyses.

Upgrading Reports from Summaries to Strategic Insights

Data-Driven PDF & PPT Exports



FEV — Engineering Data Analysis Report — Date/Version: 2021 Date Version

Top Diagnostic Trouble Codes (DTC) Summary

DTC Code	Description	Event Count	Total Runtime (s)
P0301	Cylinder 1 Misfire Detected	47	123.5
P0171	System Too Lean (Bank 1)	32	98.2
P0420	Catalyst System Efficiency Below Threshold (Bank 1)	25	85.0
P0300	Random/Multiple Cylinder Misfire Detected	18	62.4
P0128	Coolant Thermostat (Coolant Temp Below Thermostat Regulating Temp)	10	35.6

The Upgrade

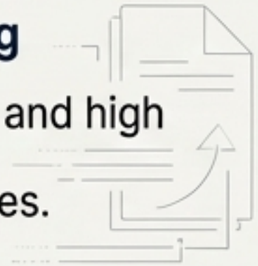
****Before***: LLM analysis was based on high-level summaries.
****After****: We now feed actual table data (up to 50 rows) directly into the LLM prompt.

> ****Example Insight*:** "Prioritize investigation of DTC P0301 due to the highest event count (47) and a total runtime of 123.5 seconds."

The Flywheel in Motion: A Timeline of Audits, Fixes, and Optimizations

Performance & Scalability

500MB+ File Handling



Problem: UI choppiness and high RAM usage with high RAM usage with large files.

Solution: Implemented memory mapping (**~80% RAM reduction**), progressive background loading, and frontend debouncing, making UI **5-10x faster**.

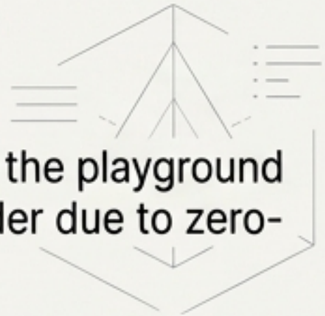
Robustness & Reliability

LangChain Caching

Problem: Sequential LLM chains chachoɞ. Sequential LLM chains caused high latency (25-50s) for complex queries.

Solution: Implemented TTL-based caching for chain results, reducing response time for repeat queries to **< 1 second**.

3D Plot Visibility

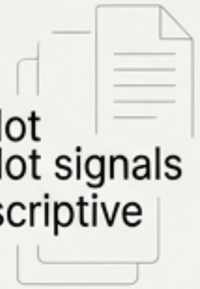


Problem: 3D plots in the playground section failed to render due to zero-size containers.

Solution: Added pre-render dimension checks and enhanced scene configuration to ensure visibility.

Core Functionality

CSV/XLS Plotting



Problem: Users could not plot chachoo. Users could not plot signals from CSV/XLS files with descriptive column names.

Solution: Implemented a multi-level fuzzy matching strategy to resolve signal names correctly.

PDF Report Accuracy

Problem: Section-specific PDF reports incorrectly included content from all sections.

Solution: Refactored PDF generator logic to ensure strict section scoping.

The Fortress: Engineered for Production Demands

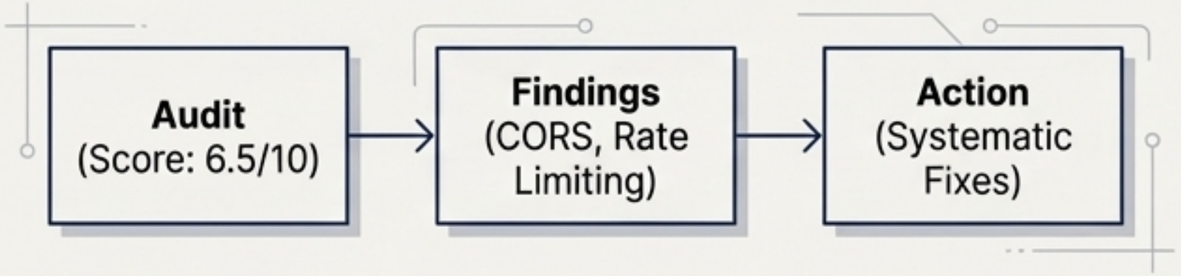
DiagAI is architected with the security, performance, and resilience required for real-world deployment. We proactively identify and mitigate risks before they impact users.

- **Proactive Security:** A continuous process of auditing, identifying vulnerabilities, and applying targeted fixes.
- **Code Quality & Stability:** Rigorous code reviews and robust error handling to ensure reliability.
- **Deployment & Operations:** A well-defined, documented deployment process with a clear understanding of production environments.

A Proactive Stance on Security and Code Quality

From Security Audit to Hardened Endpoints

The Process



The Result

- **Rate Limiting Applied**: All report endpoints are now protected (e.g., 30 reqs/60s for reports, 10 reqs/60s for PDF exports).
- **CORS Configured**: API access restricted to allowed origins via `ALLOWED_ORIGINS` environment variable.
- **Enhanced Path Validation**: Protected against path traversal attacks.

Rigorous Code Reviews for Maximum Stability

The Process

A comprehensive, CodeRabbit-style review of core OEM sections.

Critical Issues Fixed

- **MDF File Handle Leaks**: Replaced manual file operations with context managers (`with open(...)) to prevent resource leaks.

- **Thread-Safety**: Implemented a double-checked locking pattern for all singletons to prevent race conditions in concurrent environments.

- **Circular Dependencies**: Refactored imports to be lazy, ensuring architectural integrity.

Performance at Scale: Taming 500MB+ Data Files

Initial tests showed UI choppiness and high memory consumption when processing files over 500MB, making the application unstable for production use cases.

Backend (Memory)

Implemented `asammdf` memory mapping for files > 100MB. This allows the OS to handle data paging, avoiding loading the entire file into RAM.

Backend (Responsiveness)

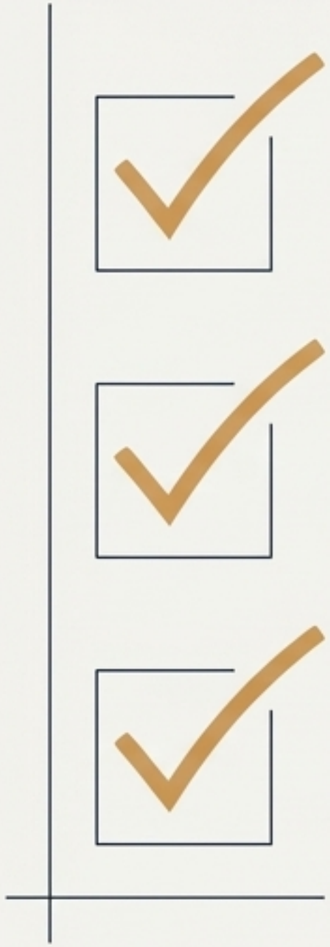
Optimized channel discovery to return a partial list immediately (<1s) while a background thread completes the full scan.

Frontend (UI)

Added a 150ms debounce to the channel search input and used `DocumentFragment` for batch DOM updates, eliminating rendering lag.

Metric	Before Optimization	After Optimization	Improvement
Memory Usage (500MB file)	~1-2 GB RAM	**200-400 MB RAM**	**~80% Reduction**
Initial UI Response Time	20-40 seconds	**< 5 seconds**	**> 5x Faster**
Channel Search Lag	Laggy, 200-300ms delay	**Smooth, <50ms response**	**Instantaneous**

Deployment Hardened: Verified, Documented, and Ready



☑ Comprehensive Error Handling

☑ Configuration Consistency

☑ Resilient Architecture

☑ PRODUCTION READY

A full deployment review confirmed stability, security, and operational readiness.

Key Readiness Indicators

- **Comprehensive Error Handling**: 470+ exception handlers are implemented across the codebase, ensuring graceful failure.
- **Configuration Consistency**: `render.yaml`, `Procfile`, and `runtime.txt` are synchronized for consistent builds.
- **Resilient Architecture**: Critical dependencies (like Supabase) have automatic fallbacks to local JSON storage, ensuring core functionality is always available.

Transparent Environmental Compatibility

What Works on Render

✅ **DeepSeek & Gemini APIs**: Fully compatible.
✅ **Supabase Database**: Fully compatible.

Known Limitations

Transparently documents that local-only services like LM Studio are not supported, but explains the system's robust fallback logic makes this non-critical.

DiagAI: The Senior Engineer in the Machine

Core Intelligence

Relentless Improvement

Production Readiness

- **We Proved its Intelligence:** DiagAI possesses the deep, multi-agent knowledge of a senior engineer, validated by a 100% success rate on complex diagnostic queries.
- **We Proved its Process:** It is built on a culture of relentless improvement, turning audits and user feedback into quantifiable gains in performance, quality, and accuracy.
- **We Proved its Readiness:** It is a production-hardened system—secure, scalable, and engineered to handle real-world data with exceptional performance (~80% less RAM, 5-10x faster UI).

DiagAI is not just a product; it is a testament to engineering excellence, ready for deployment.

Appendix: Technology & Architecture Stack

Backend & Core Application

Runtime:
Python 3.11.9

Web Framework:
Flask 2.3.3

WSGI Server:
Gunicorn

Configuration:
python-dotenv`

Data Processing & Analysis

Data Manipulation:
pandas (>=2.0.0)

Scientific Computing:
numpy, scipy

Measurement Data:
asammdf (>=7.3.0) for MF4/MDF files

CAN Database:
cantools (>=39.0.0) for DBC support

AI & Orchestration

LLM Orchestration:
LangChain (>=0.1.0)

Vector Database:
ChromaDB

Embeddings:
sentence-transformers

Cloud LLMs:
Google Gemini API, DeepSeek API

Database & Storage

Primary DB:
Supabase (Postgres)

Local Query Engine:
DuckDB

File Storage: render
Render Persistent Disks (recommended)

Frontend & Visualization

Charting Library:
Plotly (>=5.17.0)

Image Export:
Kaleido

Report Generation:
reportlab (for PDFs), python-pptx