

Data Express

A guided data-cleaning and analysis tool, built for small and medium companies whose data work still happens in Excel

Prepared September 2026 · Core application and MCP/natural-language layer built and functional; a small public sample is embedded separately on the portfolio site

1. Executive Summary

What it's for. Most small and medium companies still run their day-to-day data — sales, inventory, HR records — through spreadsheets, and most of the people who touch those spreadsheets are not trained data analysts. Cleaning a messy export, handling missing values correctly, and building a chart that actually supports a decision takes judgment a non-specialist doesn't necessarily have and can't easily justify hiring for on a routine basis. Data Express exists to close that gap: give a non-analyst a guided path to the same quality of result a specialist would produce, at comparable speed, without needing to become one.

What it is. Data Express is a data-cleaning and visualization application Luis designed specifically for that audience — companies whose data work still happens in Excel, run by people whose job title isn't data analyst. Rather than a blank workspace that assumes expertise, it walks the user through a fixed, sensible sequence of steps — inspect the data, handle what's missing or wrong, convert types, clean values, build charts — so the quality of the outcome depends on the tool's design rather than the user's training. A newer natural-language layer sits on top of the same engine, letting someone describe what they want in plain English instead of operating the guided flow by hand.

What it does. In its guided form, a user uploads a spreadsheet and is led step by step to a clean, analysis-ready dataset and a set of charts, with every change logged so nothing happens silently or irreversibly. Through its natural-language layer, the same result can be reached by describing the goal directly — "open this file, fix the missing values, chart revenue by quarter" — which, because that layer is built on an open, tool-callable interface, extends to being driven by another automated system as well, not just by a person typing a request. A small, fixed-dataset sample of this natural-language interface is also embedded on Luis's portfolio site as a free preview, so a visitor can see the underlying idea work — though it deliberately shows only a narrow slice of what the full application does.

2. Technical Summary

What it does, technically. Data Express separates its data-transform logic from every interface that reaches it. One core module implements loading, null-handling, type conversion, find-and-replace, chart generation, and export against a pandas DataFrame, and every surface of the project — the guided web application, the natural-language tool layer, the small public sample — calls those same functions rather than reimplementing them, so behavior stays identical across all of them by construction.

How it does it. The primary application is a Streamlit application (Python) built around pandas for transforms and Plotly for charts, organized as a fixed sequence a non-specialist can follow without guessing what to do next. The same core logic is exposed as seven tools over the Model Context Protocol (MCP) — list files, open a file, get a column-level overview, handle nulls, build a chart, save the cleaned dataset, reset — which any MCP-capable client can call directly; a bridge to a locally-run, free open-source model (via Ollama) already lets someone drive the full pipeline with plain-English instructions and no cloud dependency. A small FastAPI service separately reuses four of those same seven tools behind Google's Gemini model as a public, fixed-dataset sample of the same natural-language interface.

How it was built. The core transform logic was written once and reused deliberately: the MCP tool layer, the Ollama bridge, and the public sample all wrap the same functions the guided application calls directly, rather than keeping parallel implementations in sync by hand. The guided, step-by-step structure of the main application was itself a design choice aimed at the target user — someone without formal data-analysis training — rather than a simplified version of a more "advanced" tool; the natural-language layer was added on top of that same foundation once the core logic existed, extending who (or what) can drive it, not changing how it works underneath.

3. Business-Scope Documentation

The problem it addresses

Small and medium companies rarely have a dedicated data analyst on staff, but they generate the same kinds of messy, real-world data as larger ones: inconsistent exports, missing values, columns that need reshaping before anyone can act on them. The usual options are limited — hire or contract a specialist for routine cleanup work, have a non-specialist attempt it by hand with inconsistent quality and no record of what changed, or skip the analysis altogether. Data Express is built to remove that trade-off: a guided flow that gets a non-specialist to specialist-quality output, at specialist speed, without specialist headcount.

Designed for a specific user, not a generic one

The guided, step-by-step structure is the product's central design decision, not an interface preference. Each stage of the flow presents only the decisions relevant to that stage — what to do about missing values, how to convert a column's type, which chart fits the data — so a user doesn't need to already know the right sequence of operations or the right terminology to reach a correct result. This is what makes the tool usable by the actual target audience: office staff, business owners, and operational employees who handle spreadsheets as part of a broader job, not as their specialty.

From guided flow to full automation

The natural-language layer built on the same engine changes who — or what — needs to be involved at all. An administrative employee can already describe a cleanup or charting task in plain English and have it carried out correctly without learning the guided flow's specific steps. Because that same interface is exposed as standard MCP tools rather than a one-off integration, it is equally reachable by another automated system calling those tools directly — which is what ultimately allows a data-preparation step to run inside a company's existing automated workflow, with no person operating either the guided UI or a chat interface at all.

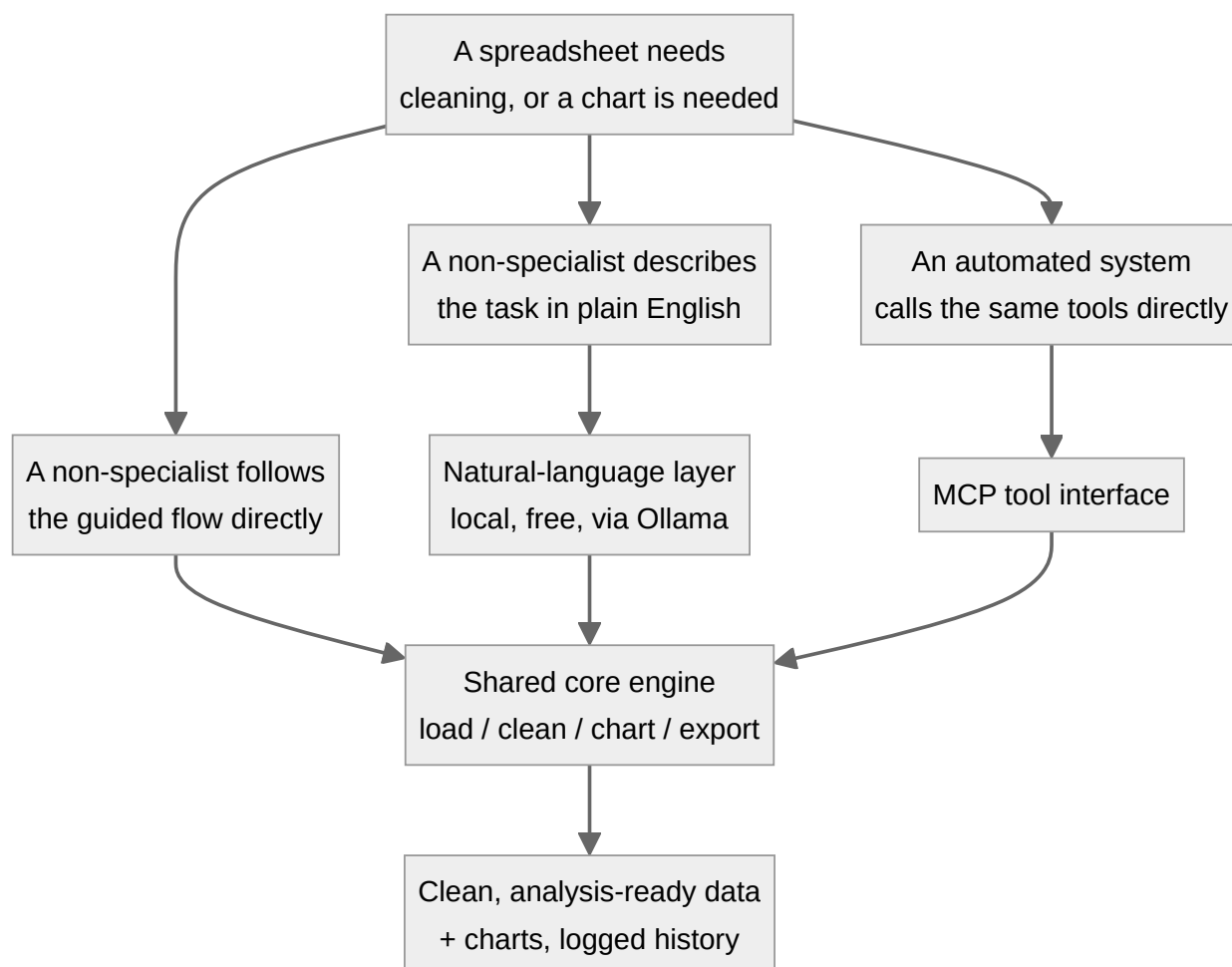


Fig. 1 — Three ways to reach the same result: guided by hand, guided by conversation, or fully automated.

Who benefits

AUDIENCE	VALUE
Small/medium business staff (non-analyst)	Specialist-quality cleaning and analysis without hiring a specialist
Business owner or manager	Faster, more consistent data preparation, with a visible history of every change made
Companies wanting automation	The same engine is callable directly by other systems via MCP, without a person operating any interface

The public sample, and what it isn't

A small version of the natural-language layer is built directly into Envoy's own interface as a free preview: a fixed sample dataset, four safe operations, no file upload. It exists purely to let a visitor see the underlying idea work in under a minute. It is not a scaled-down version of the product, and it deliberately shows only a narrow slice of what the guided application and the full tool set actually do — see Section 5 for its own documentation.

4. Technical-Scope Documentation

Core application

The Streamlit app (`data-express-v2.1.py`) is a single-file Python application built around pandas and Plotly, organized into clear, ordered stages so a non-specialist always knows the next available action: load file → inspect data → delete/rename columns → convert types → handle missing/invalid values (the largest section, since it's the step most likely to be done inconsistently by hand) → find-and-replace → optional inline cell editing → save → build up to twelve charts → cleaning history. Every transform is recorded in an in-session history log, so a user — or later, an auditor — can see exactly what changed and why. It runs standalone via Streamlit or in Docker, on port 8501.

Natural-language and automation layer

A headless version of the same logic (`dataexpress_core.py`) backs an MCP server exposing seven tools over stdio: `list_files` , `open_file` , `get_overview` , `handle_nulls` , `create_chart` , `save_dataset` , `reset_data` . Because these are standard MCP tools, any MCP-capable client can drive the full pipeline — today that includes a bundled bridge to a locally-run, free open-source model through Ollama, so a plain-English instruction such as "open enrollments.csv, delete all null values, and chart enrollments per year" is carried out with every

tool call and result shown for visibility, at no cloud cost. The same interface is equally available to any other automated system, which is what ultimately lets the data-preparation step run without a person operating either the guided UI or a chat interface.

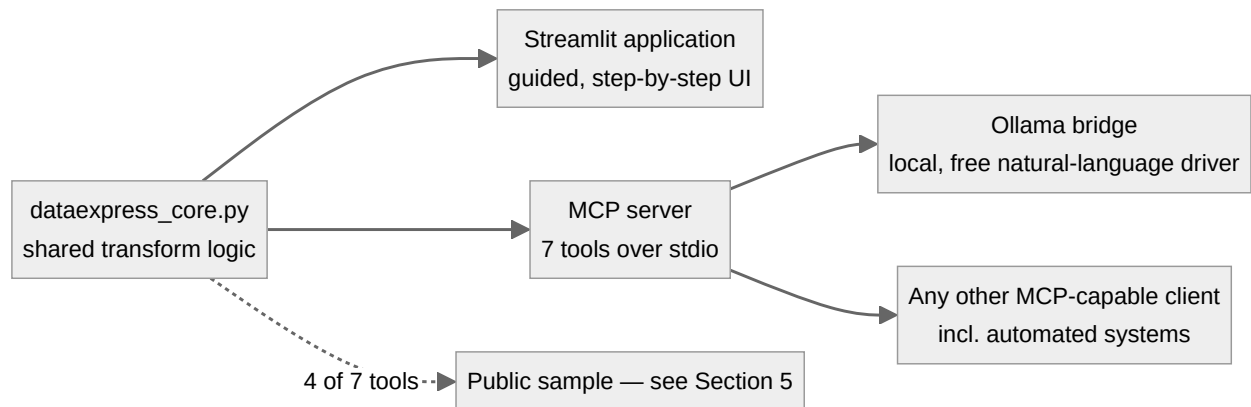


Fig. 2 — One shared engine behind the guided app, the natural-language/automation layer, and the public sample.

Public sample

A small FastAPI service reuses four of the seven tools behind Google's Gemini model as a public-facing sample, scoped to a fixed, in-memory, per-session dataset with no file upload and nothing written to disk. Its widget is built as a native component directly inside Envoy's own React frontend (not a separate embedded page), offering both a manual mode — direct calls to the tool functions, no model involved — and an agent mode using the same Gemini function-calling flow, reached through Envoy's own server proxying to this backend. It is a narrow demonstration piece, not a preview of the product's real capability — full documentation is in Section 5.

5. Demo

Executive summary

The chat widget on Luis's portfolio site is a small, free sample of Data Express's natural-language interface — not a scaled-down version of the product. It runs against a single fixed dataset and exposes four safe operations, purely so a visitor can see, in under a minute, how the natural-language layer works in principle. It does not reflect the guided application's full step-by-step workflow, its handling of a company's own real data, or the fuller natural-language and automation capability the same engine already supports elsewhere.

Technical summary

Technically, the sample is a standalone FastAPI service that exposes four of Data Express's seven MCP tools to Google's Gemini model as function-calling tools, against an in-memory, session-scoped copy of one fixed dataset — no file upload, nothing written to disk. Rate limiting (20 messages per 10 minutes), a per-turn tool-call cap, a message-length limit, and a deterministic keyword-based fallback (used automatically when no API key is configured or the live call fails) keep it safe and testable independent of API cost. It reuses the same core transform functions as the guided application, so its behavior is accurate as far as it goes — it is simply narrow by design, not a preview of the product's real depth.

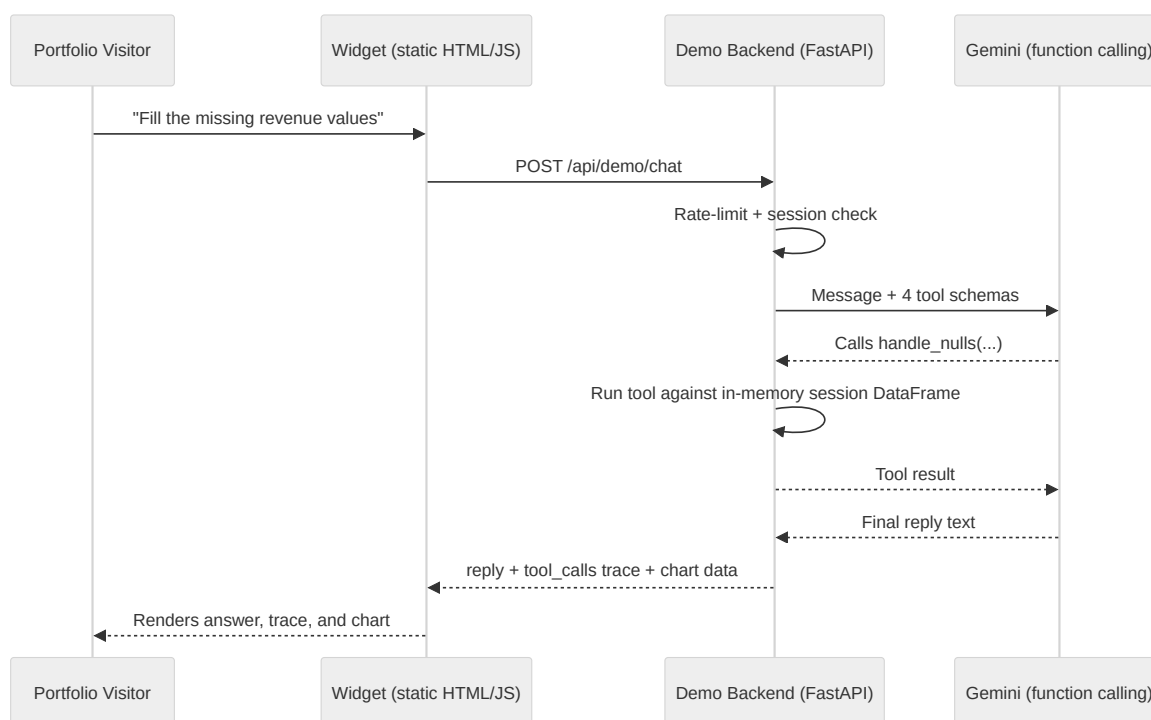


Fig. 3 — Demo request flow: visitor message to rendered result, all within the four-tool sample.

Business and technical scope

As a portfolio piece, the sample's job is to make the underlying idea tangible in a few seconds of interaction, not to represent the product. Limiting it to four operations on a fixed dataset is a deliberate scoping decision for a public, unauthenticated page — not a limitation of the underlying engine, which already exposes the complete seven-tool set to any MCP client, including the locally-run natural-language interface that already drives the guided application's full functionality today.

The sample's backend runs as its own container, joined to the external network the Envoy (CV-Website) project creates, so Envoy's Next.js server can proxy to it internally without opening any new public port. Its frontend is a native port of the standalone widget, built directly into Envoy's own React codebase with two modes side by side — manual (direct tool calls, no model) and agent (Gemini function calling) — both acting on the same in-memory session dataset, so a visitor

can see the manual-vs-agent distinction directly rather than being shown only the chat trick. It is fully built, functional, and embedded inside Envoy's own interface, not a pending integration.