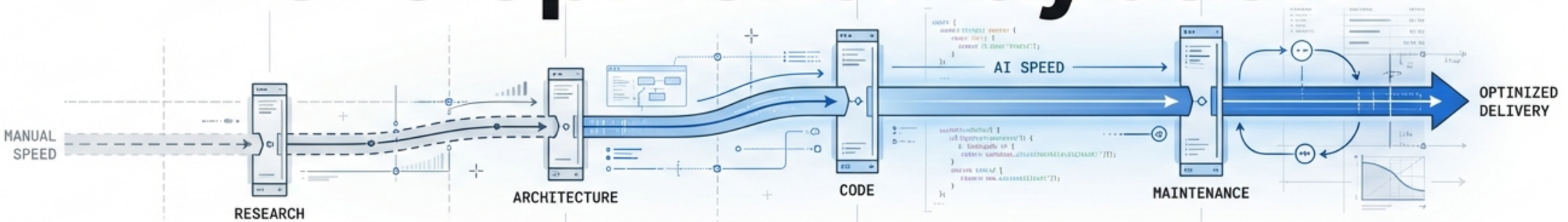
The AI-Accelerated Development Playbook



A tactical guide to achieving 50% time reduction across the SDLC.

Case Study: Workflow of the Tech Lead at Ecodeliver.

50%

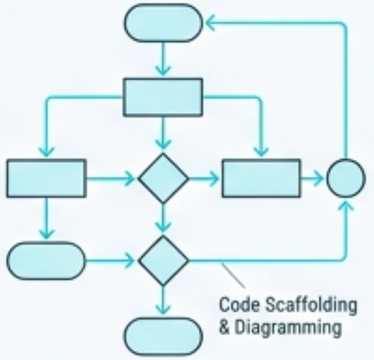
Reduction in Development Time

Real-world data from Ecodeliver startup operations. By integrating AI tools, the developer role shifts from “writer” to “architect”.

Smart Research

Synthesizing options vs. hunting for links.

Visual Architecture



Code-based diagramming and scaffolding.

Context-Aware Coding

Instant navigation of legacy codebases.

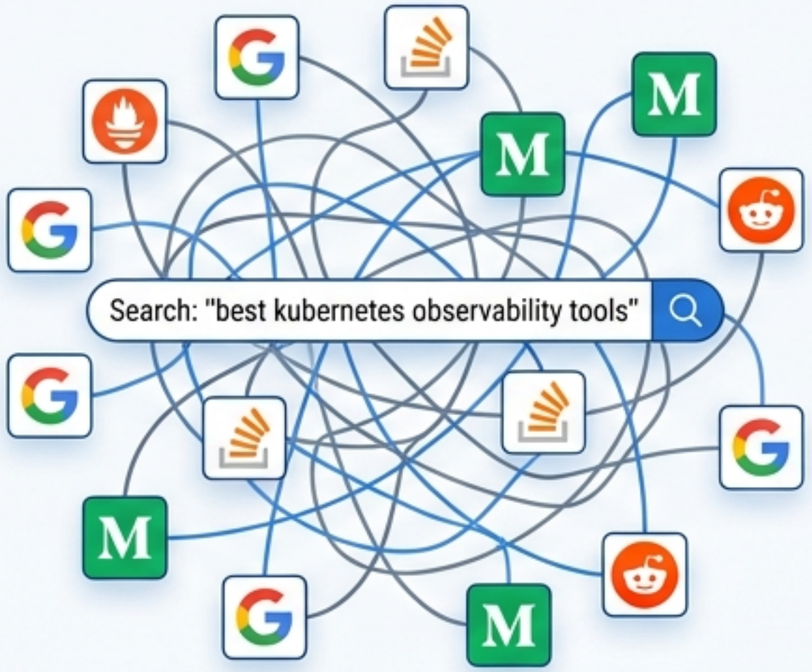
Automated Maintenance

Delegating testing and commenting chores.

“Phase 1: Smart Research & Tech Selection” in Inter Tight

Moving from 20 browser tabs to one synthesized analysis.

The Old Way



Scattered Reading: Multiple tabs open, fragmented context, contradictory info.

The AI Workflow

GPT-4 output

Observability Tools for Kubernetes Analysis

	Tool Name	Pros	Cons	Fit for TypeScript/ Node Stack
1	**Prometheus + Grafana**	Mature, highly customizable, robust community.	Complex setup, steep learning curve.	**Strong:** Native libraries for Node.js.
2	**Datadog**	Turnkey solution, excellent APM & log management.	High cost at scale, vendor lock-in.	**Excellent:** Comprehensive TS/Node agents.
3	**New Relic**	Full-stack visibility, good anomaly detection.	Can be overwhelming, pricing model varies.	**Good:** Weli-supported agents.
4	**Jaeger**	Open-source distributed tracing standard.	Requires separate storage/UI, more effort.	**Solid:** Essential for microservices tracing.

Outcome: A “shades of grey” analysis tailored to specific architecture.

Decision

Pro-Tip: Prompt Strategy: "Given our stack is TypeScript and Node, compare these tools specifically for our architecture."

Phase 2: Architecture & Visualization

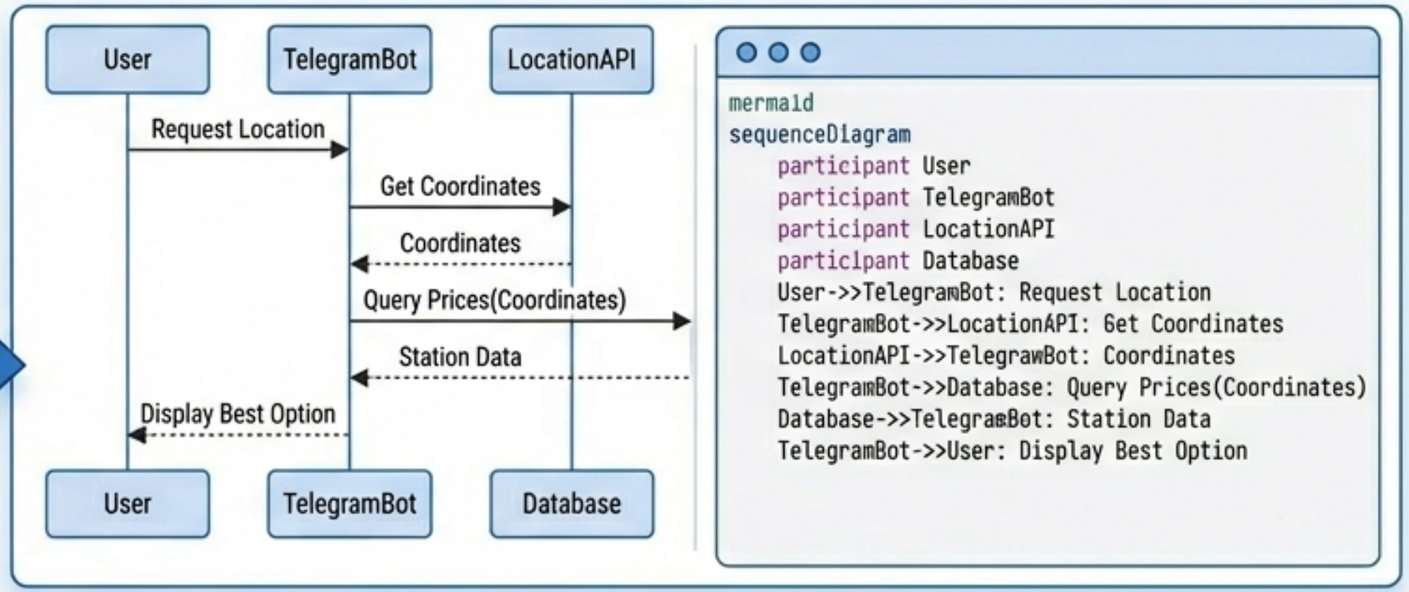
Diagrams as Code using GPT-4 + Mermaid.js.

Natural Language Prompt

Create a sequence diagram for a Telegram bot that finds the cheapest gas stations nearby.

GPT-4 Generation

Generated Mermaid.js Diagram & Code




```
mermaid
sequenceDiagram
    participant User
    participant TelegramBot
    participant LocationAPI
    participant Database
    User->>TelegramBot: Request Location
    TelegramBot->>LocationAPI: Get Coordinates
    LocationAPI->>TelegramBot: Coordinates
    TelegramBot->>Database: Query Prices(Coordinates)
    Database->>TelegramBot: Station Data
    TelegramBot->>User: Display Best Option
```

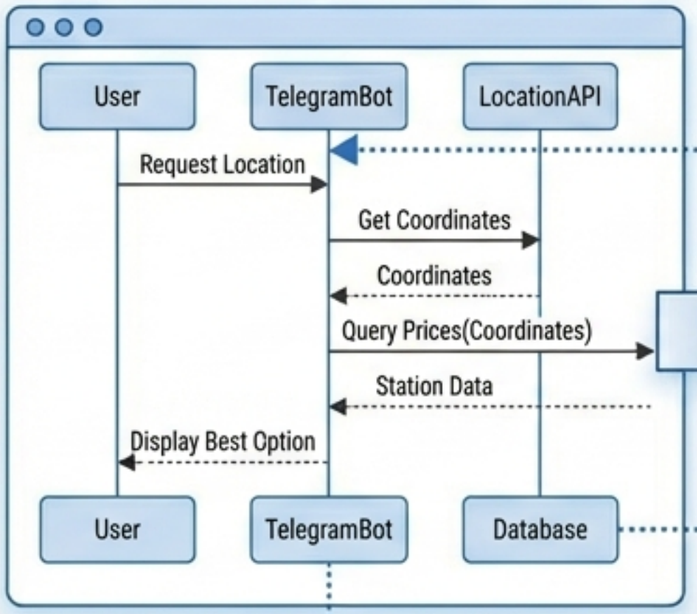
Key Benefit

Why it works: Rapid iteration. Change the logic by changing the text prompts. No manual drawing required.

Phase 3: Project Scaffolding

From Diagram to Directory.

Previous Diagram (Mermaid.js)



AI Context Injection

AI Processing

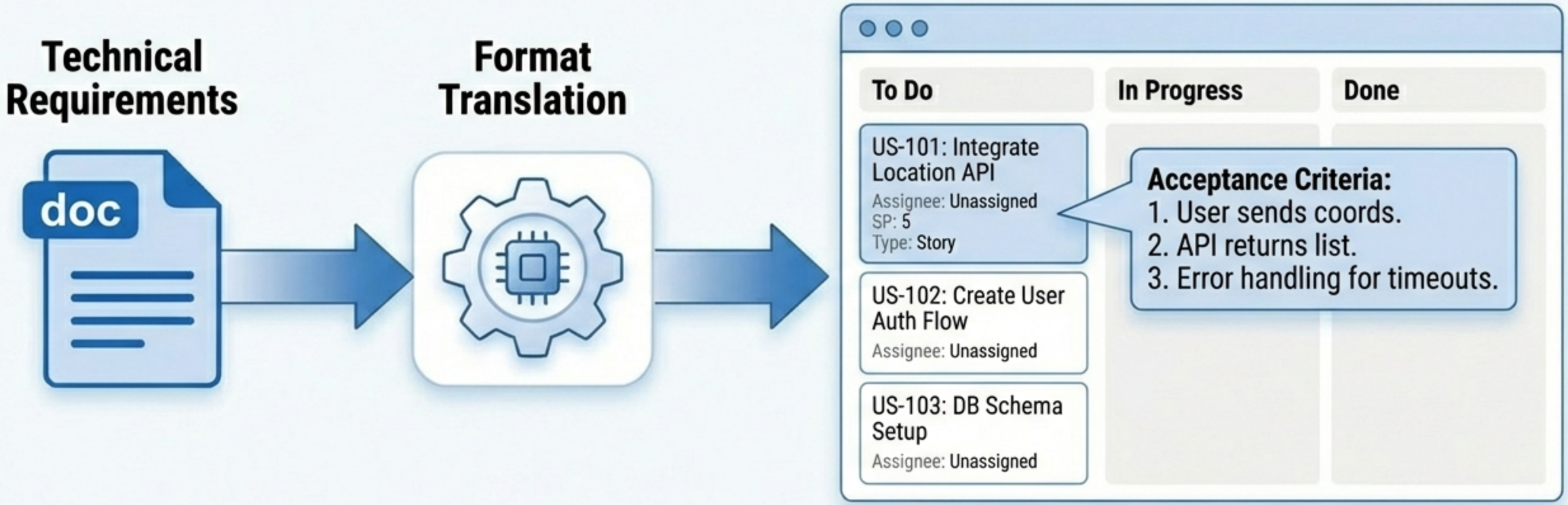
File Tree

- .
  - /src
    - /config
    - /bot
      - handlers.ts
    - /services
      - gasStationService.ts
    - /models
  - package.json
  - README.md

The Workflow: Feed the diagram back into the AI to request a directory structure. Automatically generates config files and hierarchy based on best practices.

Phase 4: Project Management & Agile

Automating the administrative burden with Jira integration.



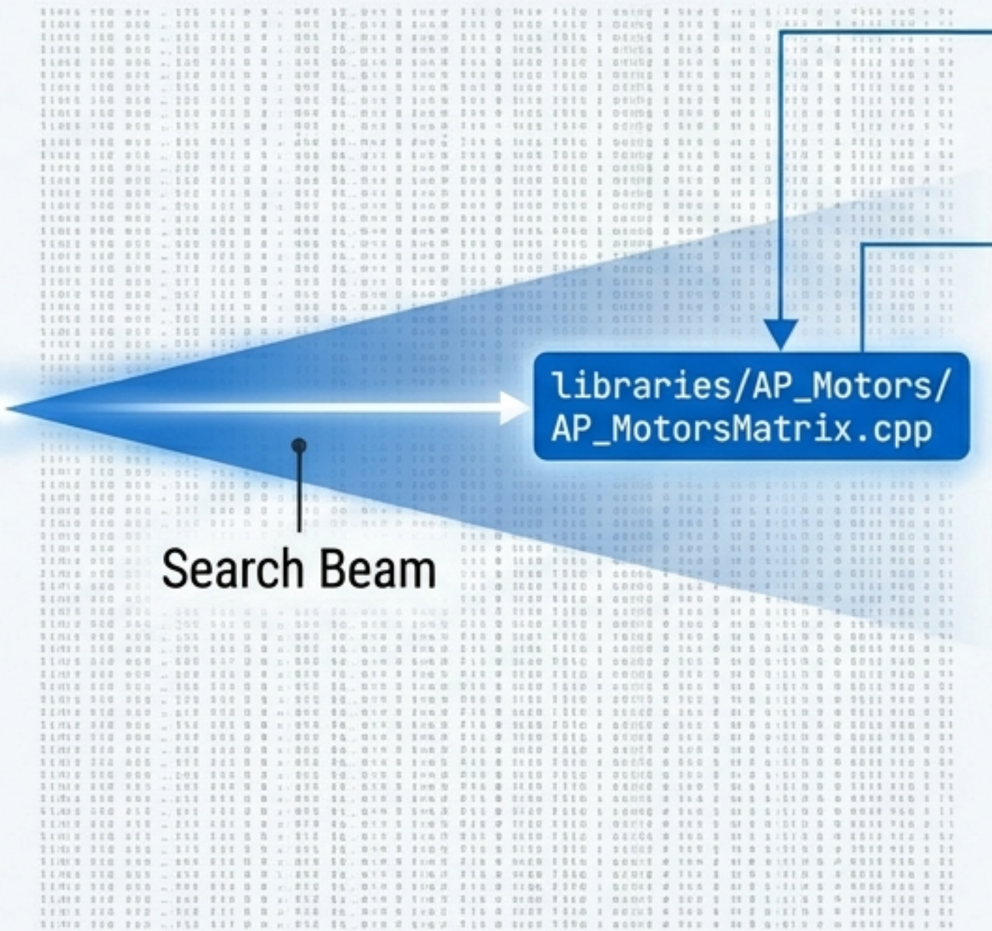
Context Injection: Provide the AI with previous tasks to match the team's writing style. The Tech Lead focuses on REQUIREMENTS, the AI handles FORMATTING.

Phase 5: Navigating Large Codebases

Case Study: ArduPilot (17,000+ files).

The Challenge: Adding a custom “motor mixer” to a massive legacy C++ drone autopilot repo.

The Old Way: Manually hunting through folders like an elephant in a china shop.



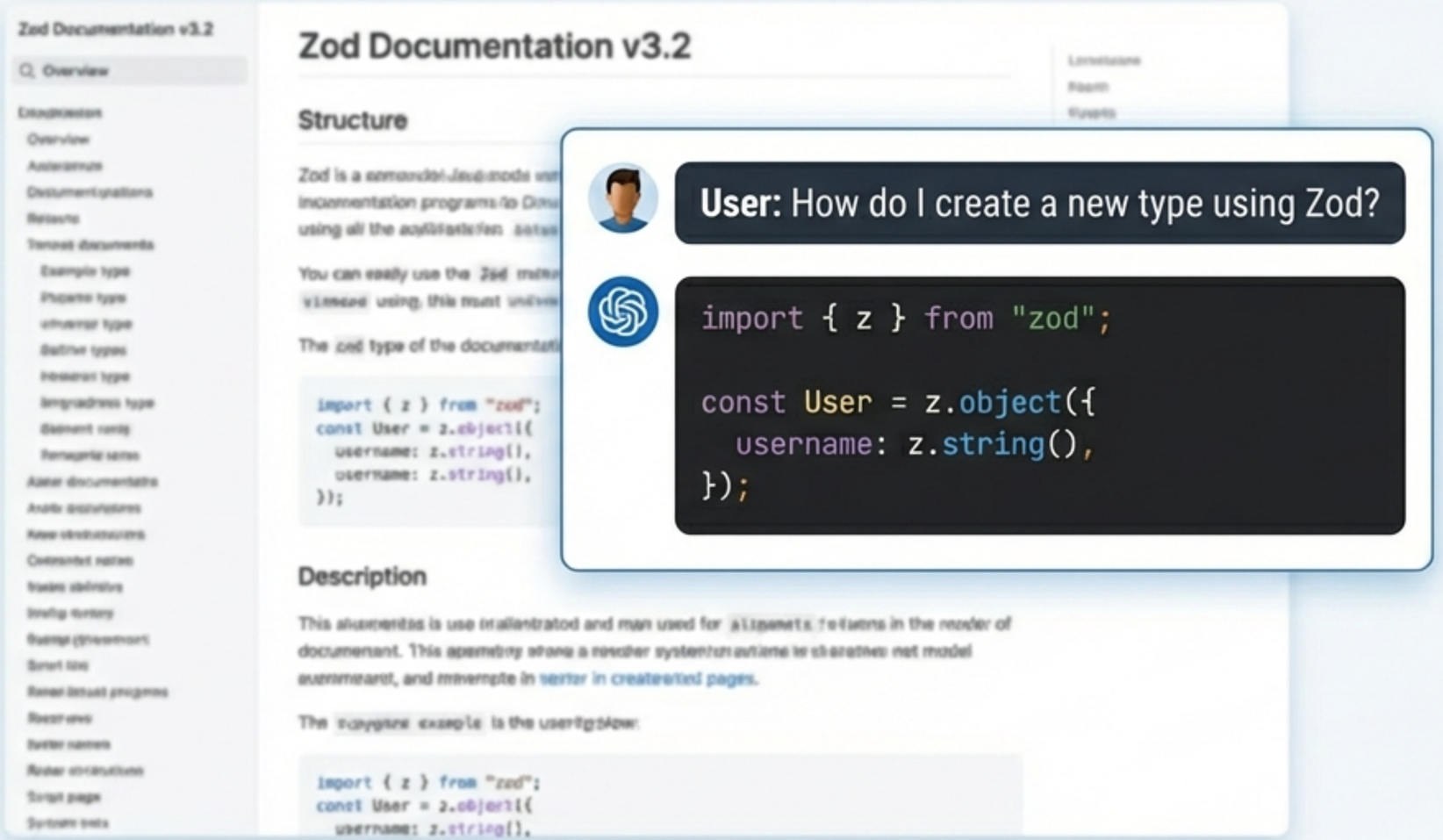
The Solution: Indexing with Cursor.

Prompt: “How do I add a custom motor mixer?”

Result: AI identifies the exact file and logical location instantly instantly.

Phase 6: Interactive Documentation

Shift from linear reading to active querying.



1. Paste URL/Docs.
2. Ask specific implementation question.
3. Get tailored snippet.

Benefit: Acts as an on-demand expert for specific libraries, bypassing the search for relevant sections.

Phase 7: Automated Code Commentary

Enforcing hygiene through style mimicry.

BEFORE: Raw Code

```
void set_motor(int i, float val) { ... }
```

AFTER: Automated Commentary

```
/**
 * @brief Sets the output for a specific motor.
 * @param i    Motor index.
 * @param val  Output value (0.0 to 1.0).
 */
void set_motor(int i, float val) { ... }
```

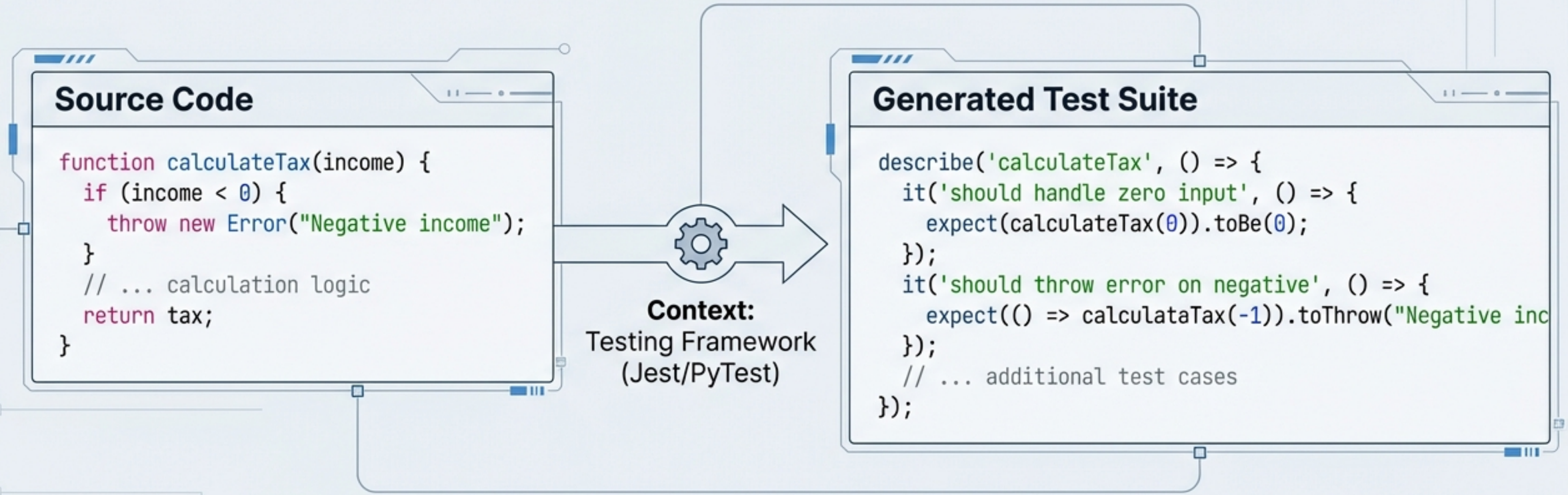
Engineering

The Strategy: "Comment this file using the same style as the rest of the project."

Result: Consistent, parse-ready documentation generated automatically. Zero human effort on chores.

Phase 8: Unit Testing

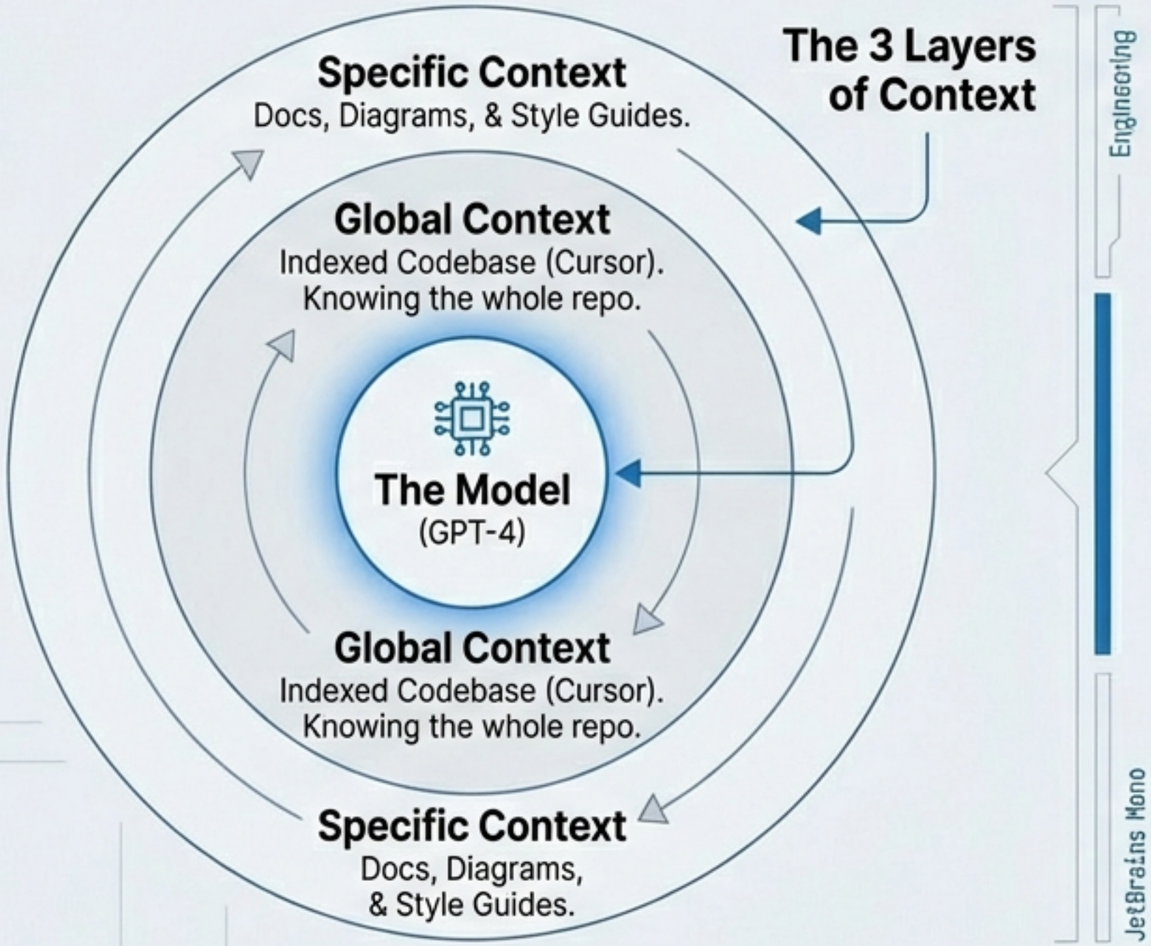
Logical derivation of test cases.



Why it works: Testing is derivative. If the AI 'sees' the code and understands the function, it can logically predict the necessary edge cases and assertions.

The Golden Rule: Context is King

The difference between generic code and production code.



Generic Prompt ⇒ Generic Code

Prompt + **Context** ⇒ `Production-Ready Code` ✓

The Acceleration Stack

The specific tools used in this case study.

LLM	IDE	Diagrams	Management
GPT-4	**Cursor**	**Mermaid.js**	**Jira**
Browsing & Reasoning.	Codebase Indexing.	Diagrams-as-Code.	Task Automation.

Languages Referenced: TypeScript, Node, Python, C++.

Total Impact

Summary of efficiency gains.

Research:
Hours -> Minutes.

Design: Manual Drawing
-> Instant Code.

Coding: Lost in Repo
-> Laser Navigation.

Maintenance: Neglected
-> Automated.

AI doesn't replace the decision; it accelerates the execution.

Where to Start

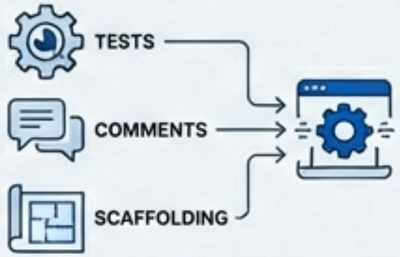
Three immediate actions to implement today.

1. Index Your Codebase.

Stop pasting snippets. Use tools that read the whole repository to gain context.

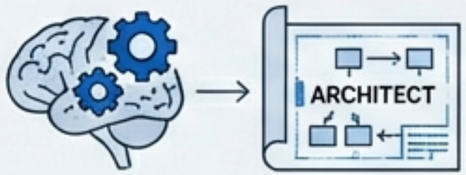
2. Offload the Mechanical.

Start by automating tests, comments, and scaffolding. These are low-risk, high-volume tasks.



3. Shift Your Mindset.

Stop writing every line. Start reviewing and orchestrating the architecture.



Source: LIDR - Carreras potenciadas por IA / Tech Lead @ Ecodeliver.