From Idea to Production SaaS in 53 Days

Building **VZZY.IO** with AI as Development Partner

The Challenge

Every organization struggles with the same problem: their strategic initiatives exist in disconnected spreadsheets, slide decks, and project management tools that force hierarchical thinking onto inherently networked relationships. Goals connect to multiple objectives. Projects serve several initiatives. Dependencies cross team boundaries in ways that traditional tools cannot represent.

We all experienced this firsthand. At a previous organization, we attempted to implement a SharePoint-based system for tracking strategic alignment. The technical implementation was sound; the forms worked, the data was captured, and the lists were organized. But adoption collapsed within months. The problem wasn't the tool. It was that **no one could see the relationships**. People could feel the organizational complexity, but they couldn't point at it and say, "That connection right there is why we're stuck."

This insight became the foundation for VZZY.IO: *the problem isn't capturing organizational data—it's revealing the relationships within it.*

The Approach: AI as Development Partner

We wanted to test a hypothesis: could AI serve as a genuine development partner for production-quality software, not just quick prototypes? Today's AI landscape is filled with AI-generated "Netflix clones" and task managers: impressive demos that would never survive contact with real users. I wanted to build something that required the "boring but critical" features: multi-tenant authentication, subscription management, role-based permissions, email integration, and data export capabilities that enterprise users demand.

The development workflow combined personal experience and expertise with Anthropic's Claude across multiple contexts: Claude Web for initial ideation and requirements discussion, and Claude Code (within VS Code) as the primary development partner for architecture decisions and implementation. Each context brought different strengths—conversational exploration for design decisions, deep codebase awareness for implementation.

One pivotal moment came during the discussion of visualization options. We had envisioned simple flowcharts to show connections between strategic items. Claude suggested ReactFlow, a React library for building node-based interfaces. This recommendation fundamentally shaped the product

direction, enabling an interactive canvas where users could drag nodes, draw connections, and literally see their organizational complexity take shape.

What We Built

VZZY.IO is an organizational visibility platform that helps teams visualize and manage strategic initiatives across flexible hierarchical layers. Unlike traditional project management tools that enforce rigid parent-child structures, VZZY allows any item to connect to any other item—reflecting how work actually flows in organizations.

Technical Scope

Metric	Value
Development Timeline	53 days (Oct 7 – Nov 29, 2025)
Commits to Production	300+ commits (~35/week average)
Database Architecture	25 tables across 38 migrations
Major Architecture Refactors	3 (unified layers, multi-solution, attributes)
Pre-built Templates	13 industry templates
Technology Stack	React 18, TypeScript, Node.js, PostgreSQL, ReactFlow

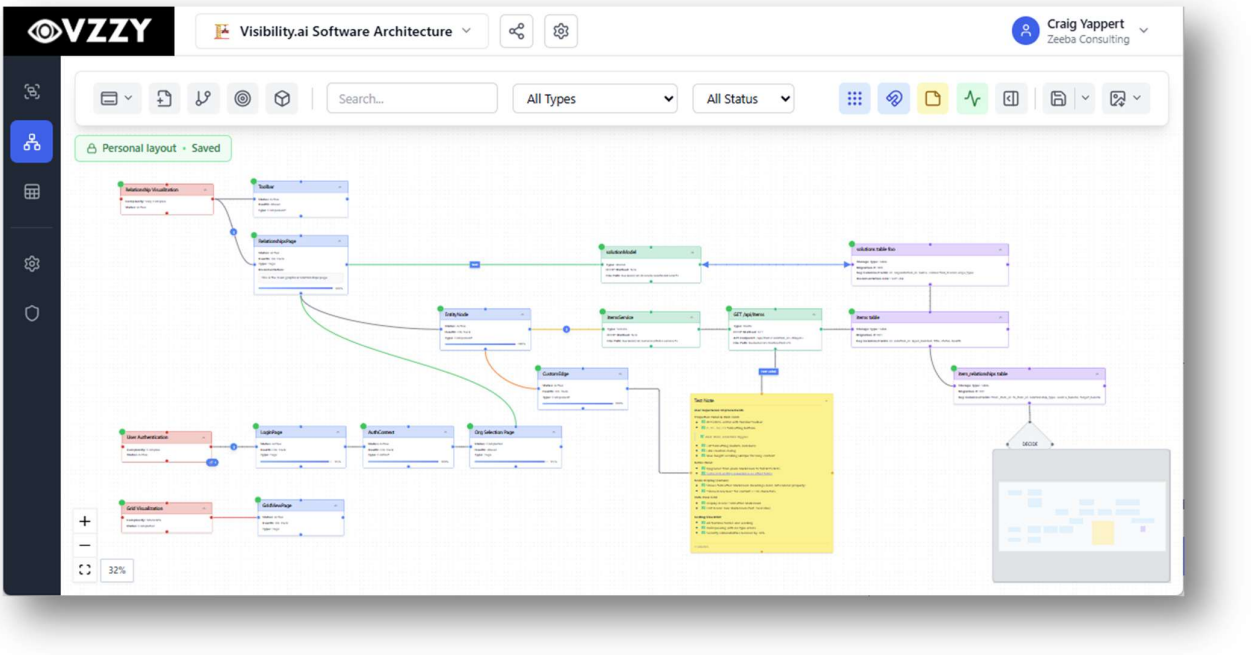


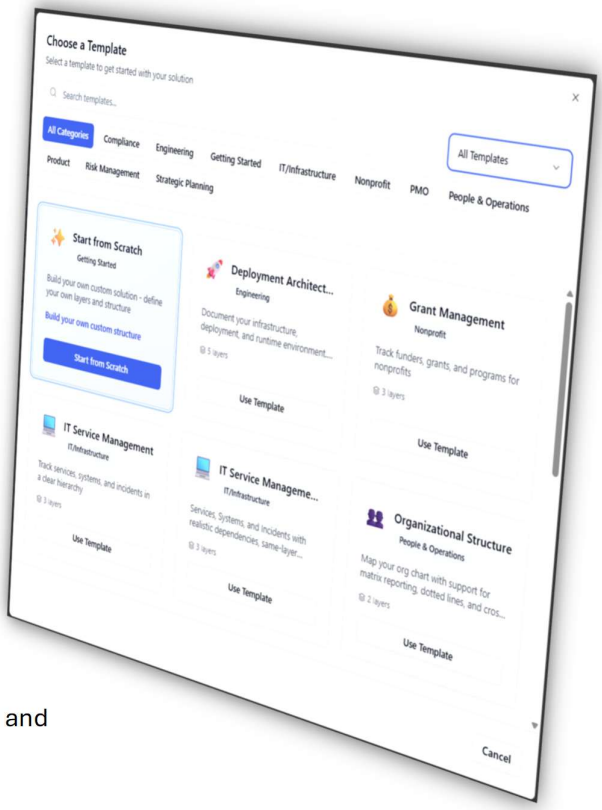
FIGURE 1: THE REACTFLOW-BASED RELATIONSHIP GRAPH REVEALS ORGANIZATIONAL DEPENDENCIES THAT SPREADSHEETS HIDE.

Key Capabilities

Visual Relationship Mapping: The ReactFlow canvas supports five custom node types (Entity, Note, Decision, Connector, Container), drag-to-connect relationship creation, auto-layout, and

personal layout persistence. Users can filter by layer, focus on connected nodes, and export for presentations and sharing.

Flexible Data Layer Architecture: Rather than enforcing a fixed hierarchy, VZZY supports 1-10 configurable layers per solution, plus a Layer 0 for annotations (notes, decisions, connectors). Each layer can have custom attributes with typed fields. This flexibility accommodates everything from simple three-layer goal tracking to complex strategic frameworks.



Multi-Solution Templates: The platform includes 13 pre-built templates spanning Strategic Planning, Software Architecture, Risk Registry, Grant Management, IT Service Management, and more. Organizations can create solutions from templates, customize them, and save their own configurations as organization templates.

Cross-Solution Linking: Items can be linked across different solutions, enabling a portfolio view of how strategic initiatives, technical architecture, and operational processes interconnect. This addresses a fundamental limitation of siloed tools.

Enterprise Features: Multi-organization support with role-based permissions (Owner, Admin, Editor, Viewer), email-based invitations, subscription tier management, Excel export with professional formatting, and shareable read-only views for external stakeholders.

The Development Compression Effect

The most surprising aspect of AI-assisted development wasn't the code generation—it was the **collapse of iteration cycles**. Traditional development estimates account for friction at every step: context switching, documentation research, debugging delays, overnight gaps, and hours lost to configuration issues. AI assistance eliminates most of this friction.

Throughout this project, initial estimates consistently proved conservative by 5-10x:

Feature	Initial Estimate	Actual Time
Share links with read-only guest views	1 week	6 hours
Layer expansion (7→10) + Layer 0 annotations	3 weeks	2 days
Multi-solution architecture	1 week	1 day
Dynamic attribute system	2 weeks	1.5 days

The acceleration doesn't come from AI writing code faster—it comes from eliminating the gaps between iterations. In traditional development, a feature might involve: implement, test, find an issue, context-switch to another task, return tomorrow, debug, and test again. With AI assistance, that same cycle becomes: implement, test, find issue, fix immediately, test, refine, all within a single working session.

Critically, this speed enabled *architectural quality rather than undermining it*. We refactored major systems three times—unifying the layer architecture, adding multi-solution support, and implementing the dynamic attribute system—because the cost of iteration was low enough to do it right rather than accumulate technical debt.

The Visualization Insight

The decision to center the product around ReactFlow visualization wasn't just a technical choice—it was a philosophical one. Traditional tools present organizational data in lists, tables, and hierarchical outlines. Users must *think* about relationships. With a visual graph, they can *see* relationships.

This shift makes certain organizational realities undeniable:

- The "strategic initiative" connected to nothing
- The compliance work is consuming resources from everything else
- The innovation goal with no supporting projects
- The single person who owns 40% of all critical dependencies

In the previous SharePoint implementation, stakeholders could deny these connections: "That's not really related," or "We handle that separately." With the visual graph, the lines are literally drawn. Denial becomes difficult when the mess has a shape.

Insights for Development Leaders

When AI Development Works

AI-assisted development excels when you maintain **architectural clarity** while delegating **implementation details**. The most productive sessions involved discussing design tradeoffs at a conceptual level, then letting the AI generate the code. The least productive involved trying to debug complex state management issues where context was fragmented across files.

This project succeeded because the requirements were precise, the architecture was well-documented, and there was a tight feedback loop between ideation, project management, and implementation. AI doesn't replace architectural thinking; it amplifies it.

Template-Driven Architecture as Accelerator

Building 13 templates in the first release wasn't feature bloat—it was a forcing function for flexible architecture. Each template stress-tested the layer configuration system in different ways: a 2-layer organizational chart versus a 7-layer strategic planning framework. By the time all templates worked correctly, the underlying system was genuinely flexible.

The "Show the Mess" Principle

Most enterprise tools try to hide complexity or force it into simplified structures. **VZZY** takes the opposite approach: show the complexity as it actually exists, then provide tools to work with it. This requires a different relationship with users; they must be ready to see their organizational reality. But for those who are, the clarity is immediate.

Conclusion

VZZY.IO demonstrates that AI-assisted development has matured beyond prototyping into a production-capable partnership. The 300+ commits, 20 database tables, and three major architectural refactors represent not just velocity but *sustained iteration toward quality* over 53 days—including multi-tenant authentication, subscription management, email integration, and enterprise export capabilities.

More importantly, the project validated a product thesis: organizational complexity tools fail not because they can't capture data, but because they can't reveal relationships. By centering the product on visual graph representation, **VZZY** provides something spreadsheets and hierarchical tools cannot—a picture of how work actually connects.

The platform is now in production at **vzzy.io**, serving organizations that need to **see** their strategic landscape rather than just list it.

Zahra Afrookhteh/Craig Yappert (Zeeba Consulting) • November 2025