
James Kavo ◆ Product Design

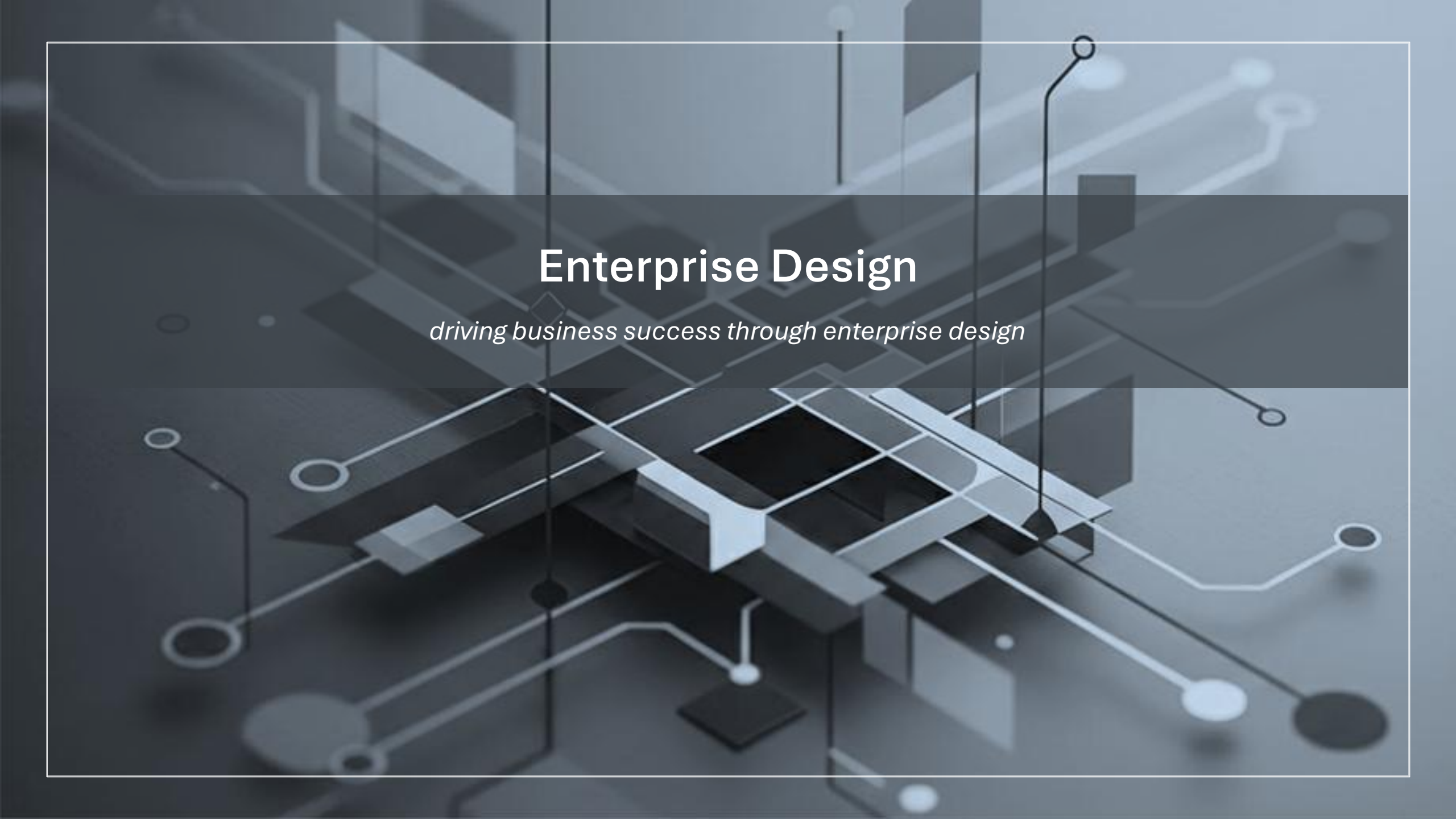
strategic thinking at enterprise scale

Approaches that work for standalone products often fail when applied to enterprise initiatives. The traditional model of adding a unit of value for a unit of work doesn't scale, so the focus shifts to taking an architectural approach that elevates the platform.

- **Enterprise Leadership (NYL)** | Orchestrating the digital experience for **12,000+ financial professionals** by ensuring cohesive workflows and architectural coherence across a massive, distributed environment.
- **Global Platform Strategy (Pega)** | Architecting the foundational patterns and presentation layers **used by Fortune 100 leaders** to build mission-critical enterprise applications.
- **Scalable Approach (NYL)** | Enabling the rapid launch of unlimited guided conversations to achieve **30x long-term value for ~1.5x the development effort**.

professional values and leadership

YES	NO
Enterprise Software	Brochureware
Continuous Improvement	Rote Process
Compounding Value	Linear Effort
Outcome-Oriented	Output-Driven
Empowerment	Enforcement



Enterprise Design

driving business success through enterprise design

Salesforce Evolution | Generative AI

Role: Product Manager, UX Lead & Business Analyst

Problem

A Salesforce CRM with 10 years of debt woven into custom-built pages — inconsistent validation, behavior that didn't match the data model, and unpleasant surprises at every turn. To make matters worse, the opaque functionality was mostly undocumented, and what existed was manually authored and invariably out of date.

Constraint

Hundreds of fields, each with bespoke validation behavior, conditional logic, and integration dependencies — across a platform that limits options for solutions. No keypress validation; formula fields inject immutable messages; “automatic” alignment that doesn’t keep things aligned. Every deviation had to be designed around explicitly while simultaneously serving as product manager, UX lead, and business analyst — three roles that normally belong to three people.

Leverage Point

Rather than authoring the prototype, specs, stories, and tests as independent documents that would drift, the work centered on a single field reference model defining every field's type, location, renderer, validation behavior, conditional logic, and integration source. The prototype is rendered from the model. Stories reference source material rather than creating copies of the information. The model is a single source of truth for all project deliverables.

Outcomes

- A single field reference model became the authoritative specification for hundreds of fields across two record types.
- Cross-artifact drift eliminated: prototype, stories, and test basis all derive from one source.
- Stories meet all standards and guidelines without requiring manual authoring and maintenance.
- Platform constraints identified and designed around up front — every deviation from out-of-the-box behavior made explicit rather than discovered in QA.
- Reusable patterns emerged from the constraint work: metadata-driven menus and anchor-field progressive disclosure — configuration, not code.

Why this matters

The conventional approach produces documents correct on the day they're written and out-of-date a week later. Having a single source of truth maintains correctness by construction: change the model, and the prototype, the user stories, and the test cases change with it. The architecture is the human contribution. The execution is delegated.

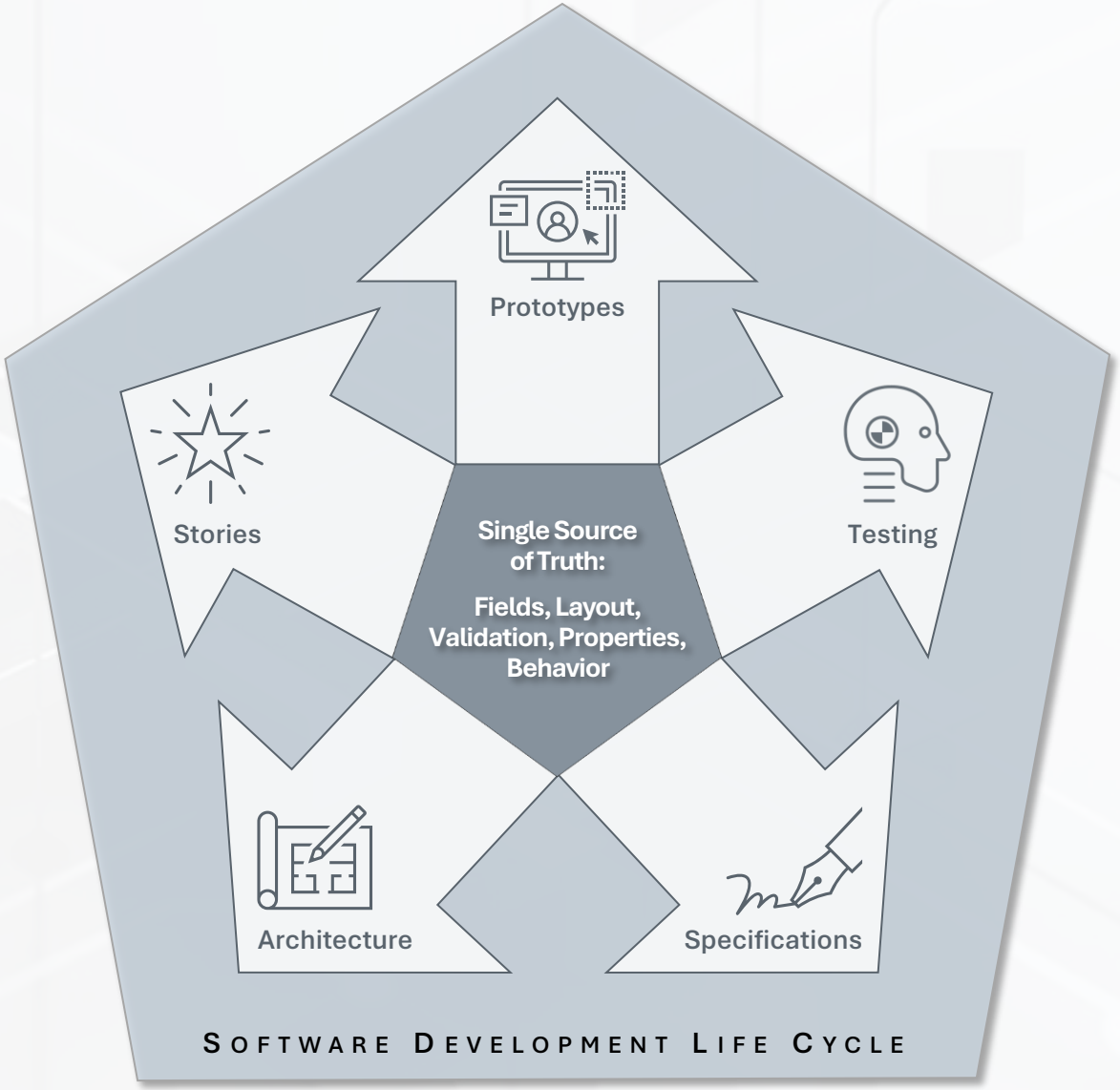
What to notice

- **The model is the deliverable**
  The prototype is one output among several; all derived from the same source.
- **Define once, derive many**
  Field specs live in one place; stories and tests reference rather than duplicate.
- **Constraints became patterns**
  Platform limitations designed around up front rather than discovered in development or testing.

Case
Salesforce Evolution | Generative AI

Company
New York Life

Role
Product Manager, UX Lead & BA



WHY THIS MATTERS

The field reference model doesn't just specify behavior — it ***surfaces*** it. Hovering any field reveals its full specification without leaving the prototype: data type, validation rules, accepted values, and testing notes.

WHAT TO NOTICE

- There's a single source of truth
- Field definitions are available without leaving the design to view a different artifact
- An unadulterated view of the screen is available by toggling off the information icons

CASE

Salesforce Evolution | Generative AI

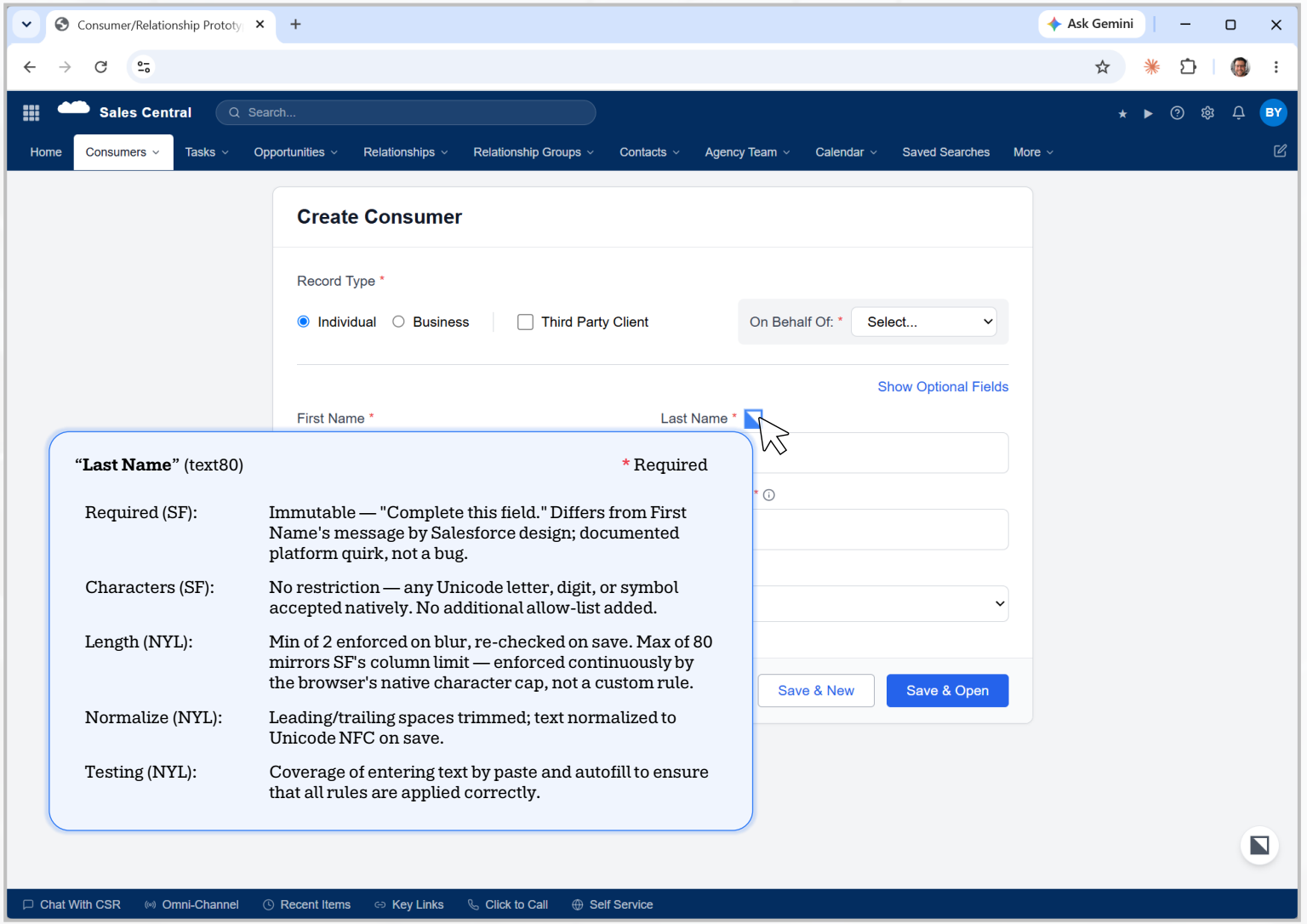
COMPANY

New York Life

ROLE

Product Manager, UX Lead & BA

1 + 1 = 10: Prototype generated from the Specification. Specification details available in the Prototype.



Periodic Review (as a Guided Conversation)

Role: Product Manager & Design Lead

Problem

Periodic Review (PR) conversations focus on revisiting existing policies and holdings, addressing life changes, and refining decisions over time. The product, however, was designed specifically for initial sales appointments and assumed that workflow. When agents conducted a Periodic Review, the system did not adapt. It continued to reflect the logic of selling new business — presenting screens and sequences optimized for prospecting rather than deepening a client relationship. Instead of guiding the conversation, the system assumed a different one, forcing agents to work around the misfit navigation in real time and discouraging the use of best practices.

Constraint

Periodic Review needed to ship. The initial direction was to build a dedicated PR experience, which would have duplicated core capabilities and created a parallel track inside the product — introducing defined appointment boundaries and closing behaviors that would exist only inside PR and not when conducting other conversations. We needed a solution that **satisfied PR requirements while strengthening the overall product architecture**.

Leverage Point

Rather than building a PR-specific interface, we invested in the Conversation Framework as a reusable orchestration layer — introducing a formal appointment container with defined start and finish states and standardizing end-of-conversation behaviors (action items, referrals, scheduling, summary) in a way that applied across conversation types.

Outcomes

- Achieved **30x long-term value for ~1.5x the initial effort**
- Delivered Periodic Review functionality **on time** without creating a Periodic-Review-only universe
- More than **90% of new capabilities reusable** across conversation types
- Productized **best practices** for closing an appointment
- Created authoring capability for all kinds of future guided conversations

Why this matters

Before this framework, process logic lived in the agent's head, not the system. This manual navigation created an **implicit process burden** and capped organizational agility. By shifting that logic to a **reusable orchestration layer**, we decoupled intent from functionality—enabling the system to lead the conversation without rebuilding the core.

What to notice

- **Orchestration over customization**
  Using a logic layer to guide the user while keeping the core system intact.
- **Help without hindering**
  Providing direction on main workflows and encouraging best practices without restricting the user's expertise.
- **Design as a force multiplier**
  Focusing effort on the core system logic rather than individual page layouts to achieve compounding value.

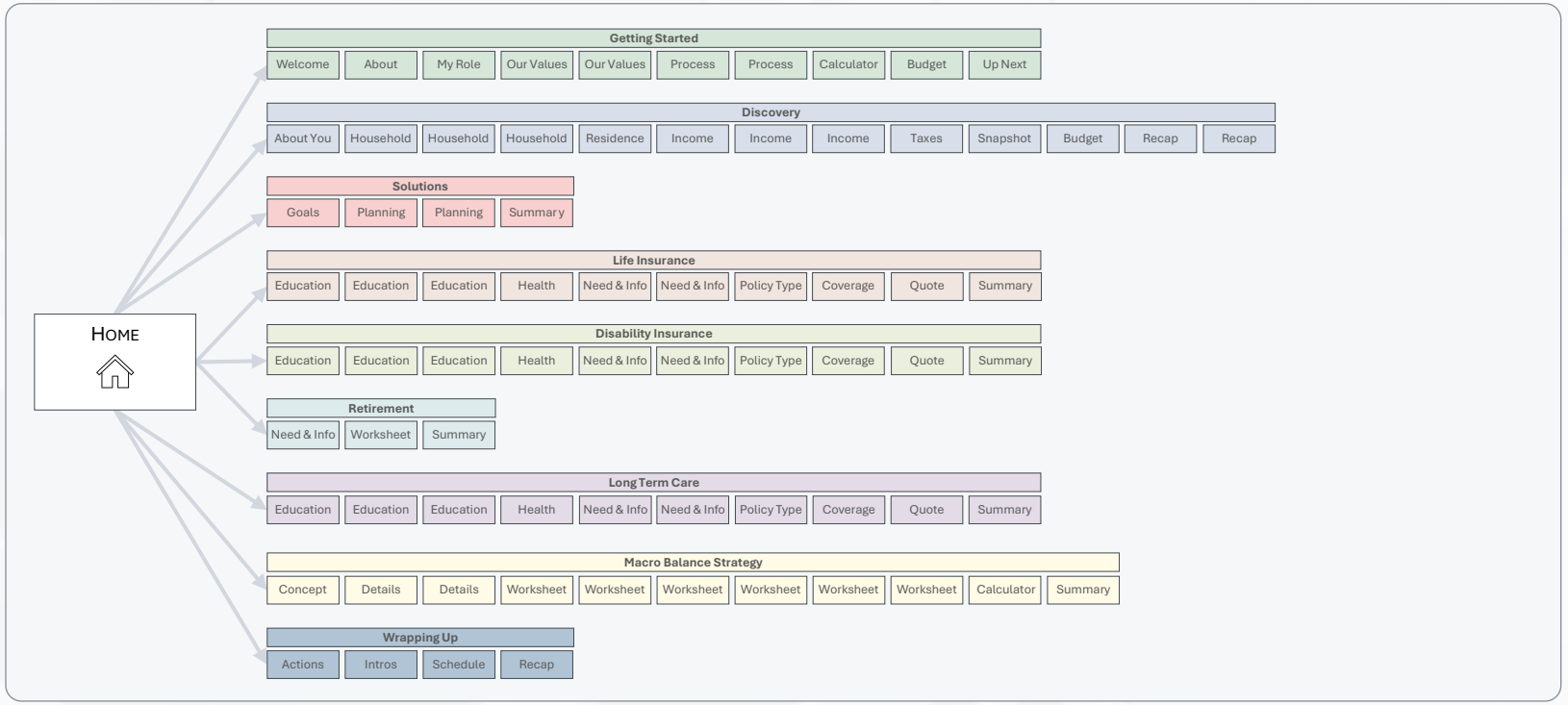
◇ ◇ ◇ ◇ ◇ ◇ ◇ ◇ ◇ ◇ ◇

Case
Periodic Review (as a Guided Conversation)

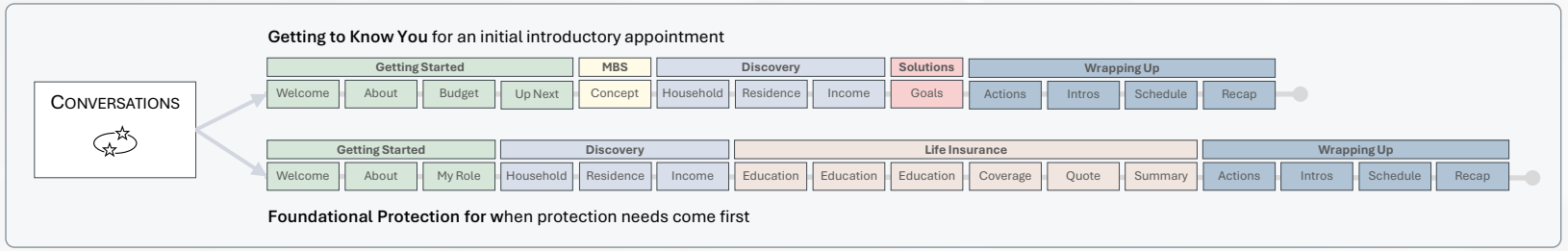
Company
New York Life

Role
Product Manager & Design Lead

Before: Manual navigation across isolated functional silos



After: Goal-driven conversations via a declarative framework



Why this matters

By standardizing the Definition and Orchestration, we shifted the UI from a manual authoring task to **output derived from the system's logic**. This architecture delivered **30x long-term value**, enabling new business lines to be published via configuration rather than custom engineering.

What to notice

- The definition is simple
- The orchestration is reusable
- The navigation is generated

Case
Periodic Review (as a Guided Conversation)

Company
New York Life

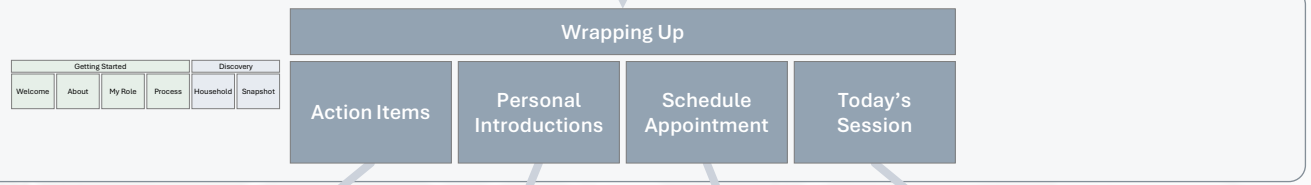
Role
Product Manager & Design Lead

Definition

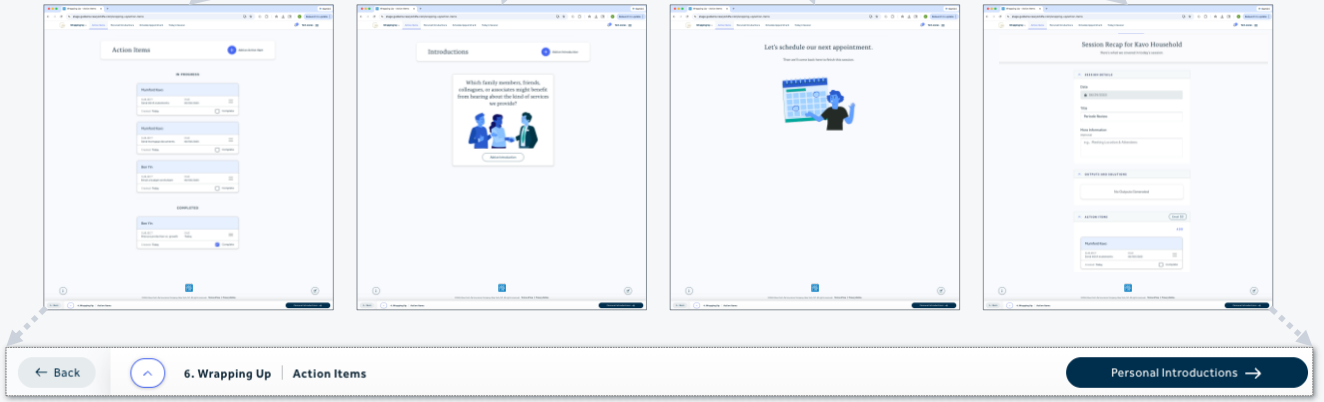
Conversation Topic: Label = "**Wrapping Up**"

Item 1:	Module = WrapUp	Page = /WrapUp/ActionItems	PageRules = None	Label = "**Action Items**"
Item 2:	Module = WrapUp	Page = /WrapUp/PersonalInto	PageRules = None	Label = "**Personal Introductions**"
Item 3:	Module = WrapUp	Page = /WrapUp/ScheduleAppt	PageRules = None	Label = "**Schedule Appointment**"
Item 4:	Module = WrapUp	Page = /WrapUp/TodaySession	PageRules = None	Label = "**Today's Session**"

Orchestration



Guided Conversation



Event Model and Best Practices

Role: Design Lead & Architect

Problem

Enterprise UI authors struggled to build consistent, high-quality interfaces. Each component was implemented with its own behavior and event logic, making outcomes unpredictable, best practices hard to apply, and upgrades risky for customers with critical production applications.

Constraint

The platform was already widely deployed, so **existing customer applications could not break**, and improvements had to be introduced incrementally while active development continued across multiple teams.

Leverage Point

Rather than improving individual components in isolation, **the initiative focused on system-level infrastructure.** Event handling was moved out of components and into a shared model, allowing components to invoke behavior without owning it. A UI Gallery of production-grade examples made the model concrete and discoverable, ensuring that the easiest way to build was also the correct one. Together, these shifts redirected effort from local fixes to reusable patterns that scaled across the platform.

Outcomes

- Time to design and build new pages **reduced by over 70%**
- Best practices applied consistently without enforcement
- Upgrades became **safer and more predictable**
- Teams internalized the model and made **aligned decisions independently** over time

Why this matters

Siloed functionality forces users to relearn configuration and behavior for each unique component. Centralizing behavior ensures platform-wide consistency and optimized execution—accelerating user mastery and lowering the "cognitive tax" of enterprise-scale work.

What to notice

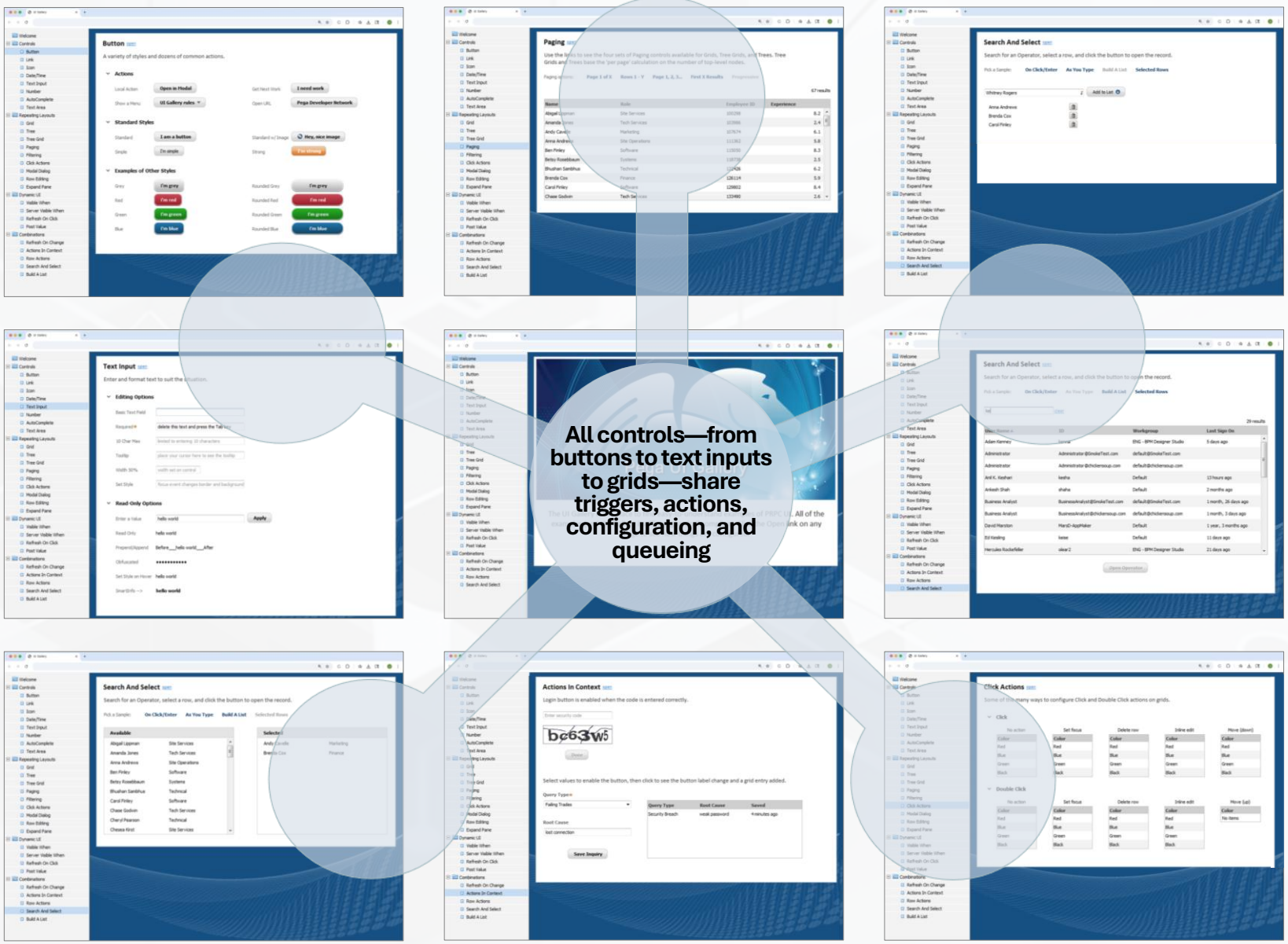
- **Predictable Interaction**
  Moving events out of individual components ensures platform-wide consistency for configuration and behavior.
- **Intelligent Execution**
  Logic to sequence or parallelize events optimizes performance and prevents collisions.
- **Decoupled Logic**
  Dependencies managed externally to prevent design drift, speed development and testing, and simplify system-wide updates.

◇ ◇ ◇ ◇ ◇ ◇ ◇ ◇ ◇ ◇ ◇

Case
Event Model and Best Practices

Company
Pega

Role
Design Lead & Architect



Backlog Foundry: Epic Stories from Scrap

Role: Product & Design Lead

Problem

Agile standards for epics, stories, and **related artifacts are fragmented** across working agreements, templates, informal conventions, and institutional knowledge, making consistent application slow, error-prone, and increasingly expensive at scale—especially as artifact volume grows.

Constraint

The organization relies on shared Agile standards, but inputs are messy, incomplete, and high-volume. Teams cannot be expected to remember or consistently interpret distributed rules, and post-hoc human review does not scale. A solution must enforce standards without relying on discipline, training, or manual cleanup.

Leverage Point

Rather than enforcing standards through review, Backlog Foundry **applies best practices at artifact generation time**. Canonical artifact definitions, structured templates, and embedded working agreements allow epics, stories, and related artifacts to be treated as derived objects, with standards applied deterministically and automatically regardless of input quality or source.

Intended Outcomes

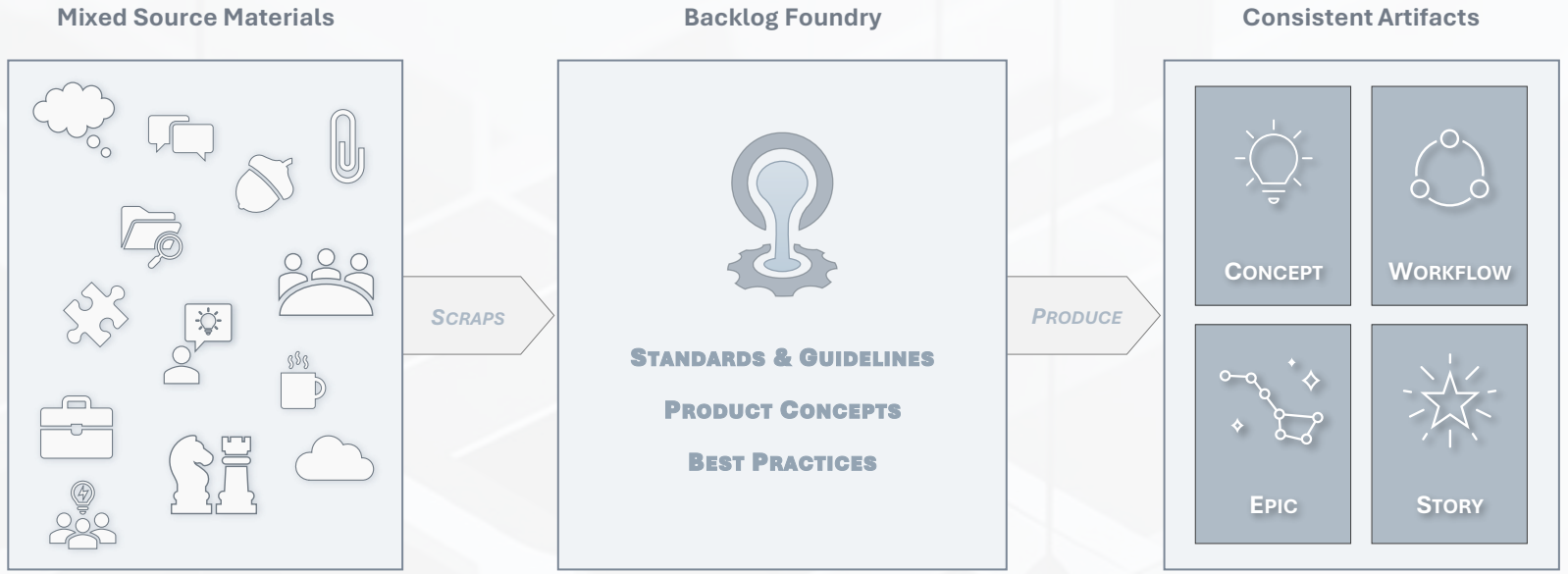
- Agile artifacts **start compliant** by default, reducing downstream rework and review cost
- Large volumes of epics and stories can be **generated safely without degrading quality**
- **Consistency improves** across teams and over time without additional process overhead
- AI-assisted generation becomes viable without increasing compliance risk

Why this matters

Before this framework, Agile standards lived in documents and institutional memory, making compliance dependent on recall, interpretation, and manual review. At scale , this approach breaks down, increasing cost and inconsistency.

What to notice

- Standards are enforced at **generation time**, not through post-hoc review
- Rules, definitions, and assumptions are **centralized and applied deterministically** across projects
- Agile artifacts are treated as **derived objects**, not authored documents



The time to produce agile artifacts **decreased by over 50%** while greater consistency improved the predictability and speed of development.

Case
Backlog Foundry

Company
New York Life

Role
Product & Design Lead

Why this matters

Teams lose enormous time reconciling backlog status across tools, formats, and teams.

When backlog artifacts are generated from a shared decision system, consistency and visibility emerge automatically. Teams spend less time explaining work, while leaders gain a reliable, comparable view across projects without custom reporting or manual reconciliation.

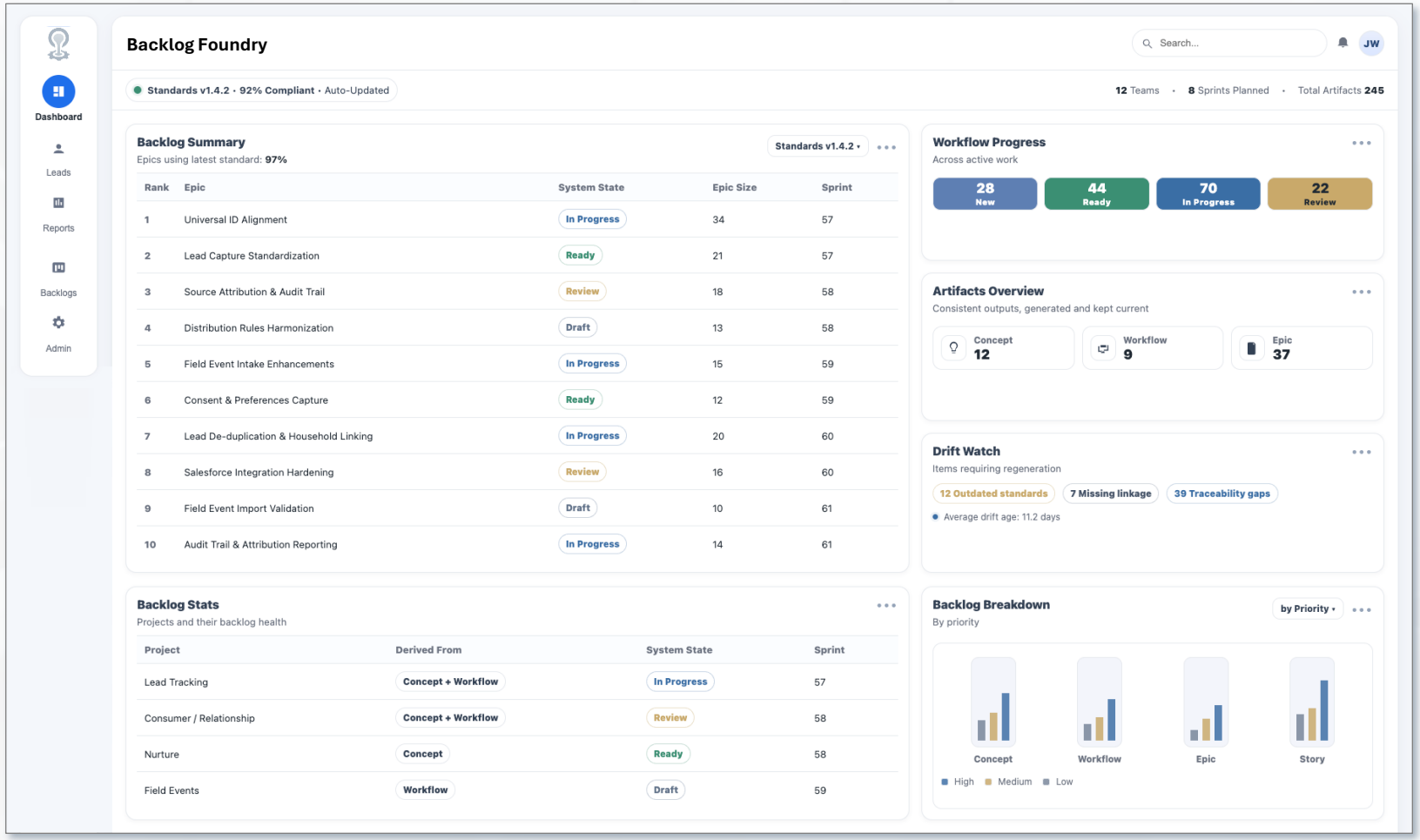
What to notice

- Status reflects derived artifacts, not manual updates
- Teams spend less time producing documentation and more time building things

Case
Backlog Foundry

Company
New York Life

Role
Product & Design Lead



McKESSON Unified Platform Architecture for Healthcare

Role: Design Architecture (Founding UX Hire)

Problem

McKesson's clinical and care-management products were **built independently by domain**—Disease Management, Case Management, Utilization Management, Providers, and Patients/Members—resulting in fragmented workflows, siloed data, and no shared view of the patient/member. Users were forced to move between systems to complete a single task, while the organization paid for multiple full-stack teams to inconsistently solve the same problems.

Constraint

This could not be solved by redesigning individual products within the existing structure. Teams, technologies, and workflows would reproduce the same fragmentation. The effort required a **shared platform and interaction framework** that could support radically different usage models—sequential, exploratory, and highly unstructured—while operating in a regulated, enterprise healthcare environment with embedded clinical SMEs but no traditional PM or BA resources.

Leverage Point

Established a **centralized architecture and design framework in parallel with the technical architecture**—defining canonical interaction models, shared components, and authoring patterns that could generate multiple domain-specific applications from a common foundation. Design functioned as the **connective tissue between clinical expertise and software engineering**, translating real-world practice into system-level decisions.

Outcomes

- A **unified platform vision** aligned across domains and disciplines
- A shared interaction framework capable of supporting **diverse clinical workflows**
- Reduced duplication of effort across products and teams
- A **foundation for consistent**, scalable application development across the enterprise

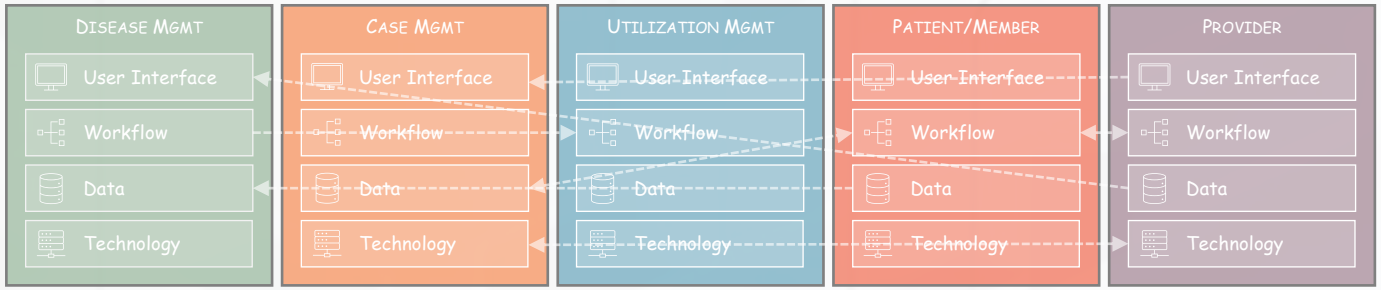
Applications share an enterprise foundation without losing the ability to provide domain-tailored experiences

CASE
Unified Platform Architecture for Healthcare

COMPANY
McKesson Health Services

ROLE
Design Architecture (Founding UX Hire)

Before: Multiple products, independent logic, no shared data layer

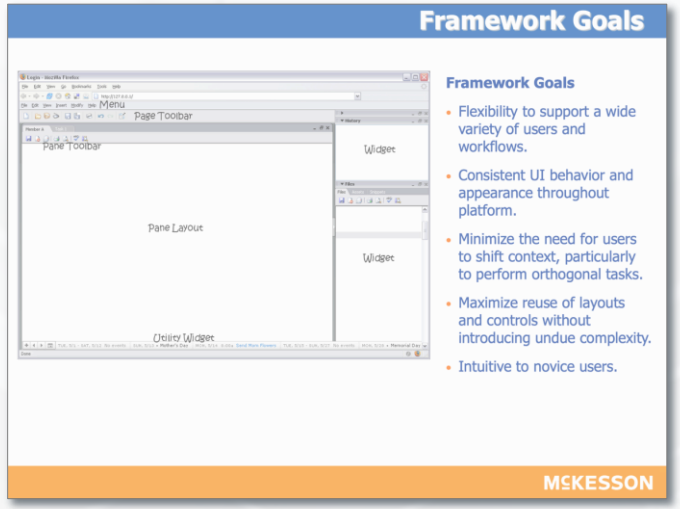


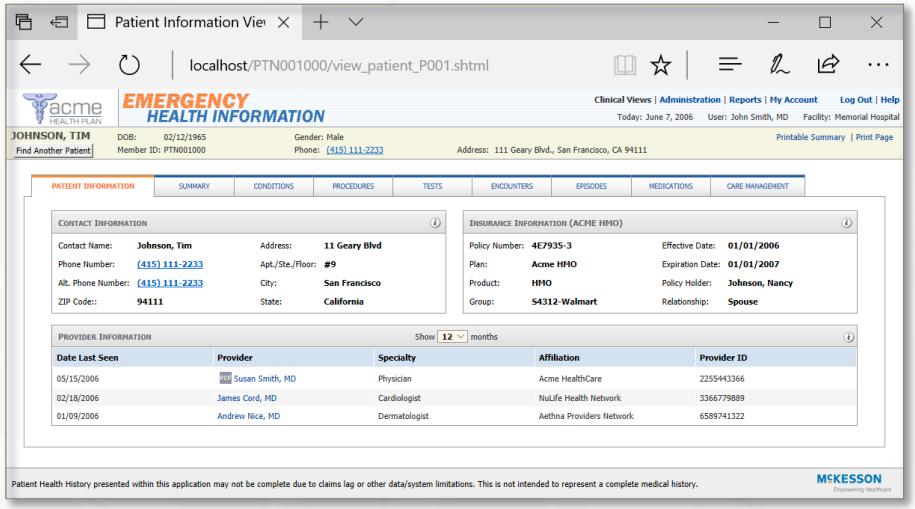
New Framework: Centralized authoring and interaction platform

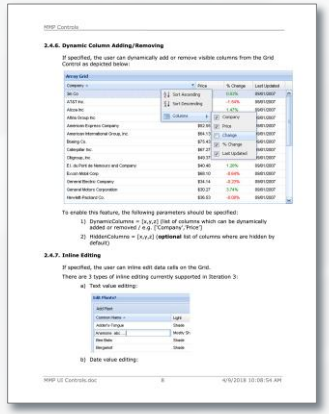
After: Purpose-designed applications using common patterns, workflow, and data

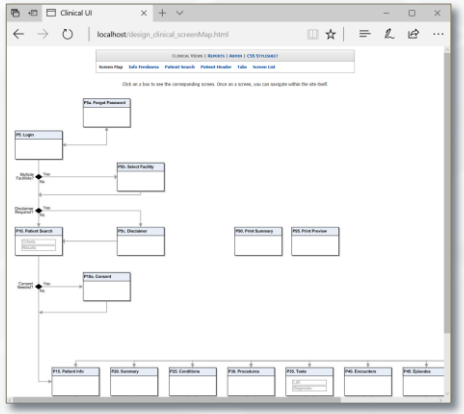
DISEASE MGMT	CASE MGMT	UTILIZATION MGMT	PATIENT/MEMBER	PROVIDER
Patient/Member Electronic Health Record				
Reporting and Administration				
Application Experience		App Experience		Application Experience
Authored Workflow				Authored Workflow
Shared Data Model				
Shared Technology Stack				

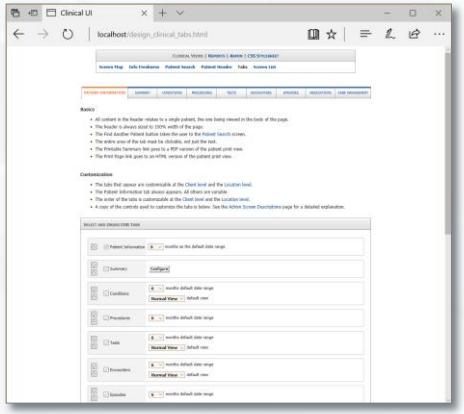
A single design system supported diverse clinical workflows

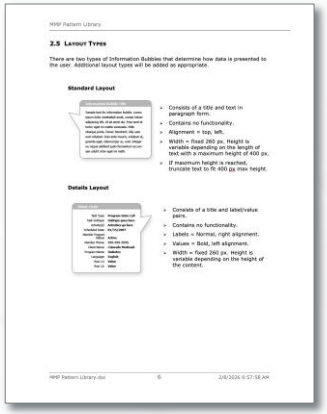












◇ ◇ ◇ ◇ ◇ ◇ ◇ ◇ ◇ ◇ ◇

CASE
Unified Platform Architecture for Healthcare

COMPANY
McKesson Health Services

ROLE
Design Architecture (Founding UX Hire)

How Product Work Gets Done

Role: Co-Ideator & Co-Author

Problem

At New York Life, product, UX, and technology teams were moving forward with **incomplete or misaligned problem definitions and solution concepts**. Research, design, and delivery often proceeded before there was a shared understanding of what was being built, for whom, and why. This led to late-stage rework, repeated debates, and avoidable friction between disciplines — despite good intent and strong individual contributors.

Constraint

This was not a tooling problem and could not be solved by adding more documentation or meetings. The organization needed to **move quickly while coordinating across users, business, and technology**. Any solution had to preserve agility while preventing teams from advancing on flawed assumptions — without introducing heavyweight governance or slowing delivery.

Leverage Point

Introduce explicit decision gates at the moments where misalignment is most costly:

- Problem definition, and
- Solution validation.

By formalizing a shared Product Concept artifact before design begins—and a shared validated prototype before delivery — teams are guided to **align on intent, constraints, and expectations** before moving forward. Progress is gated on shared understanding, not timelines or phase completion.

Outcomes

- Teams do not advance until users, business, and technology **agree on what is being proposed**
- Misalignment is **surfaced early**, when it is cheapest to resolve
- Design and delivery operate on stable conceptual models rather than assumptions
- Significant reduction in downstream rework and late-stage debate
- Faster delivery by eliminating false starts and churn

Why this matters

When teams move forward without shared understanding, misalignment is discovered late — after design and delivery work is already in motion. This drives rework, repeated debates, and unnecessary friction between disciplines. Introducing explicit alignment gates prevents teams from committing to flawed assumptions and makes progress safer, faster, and more predictable.

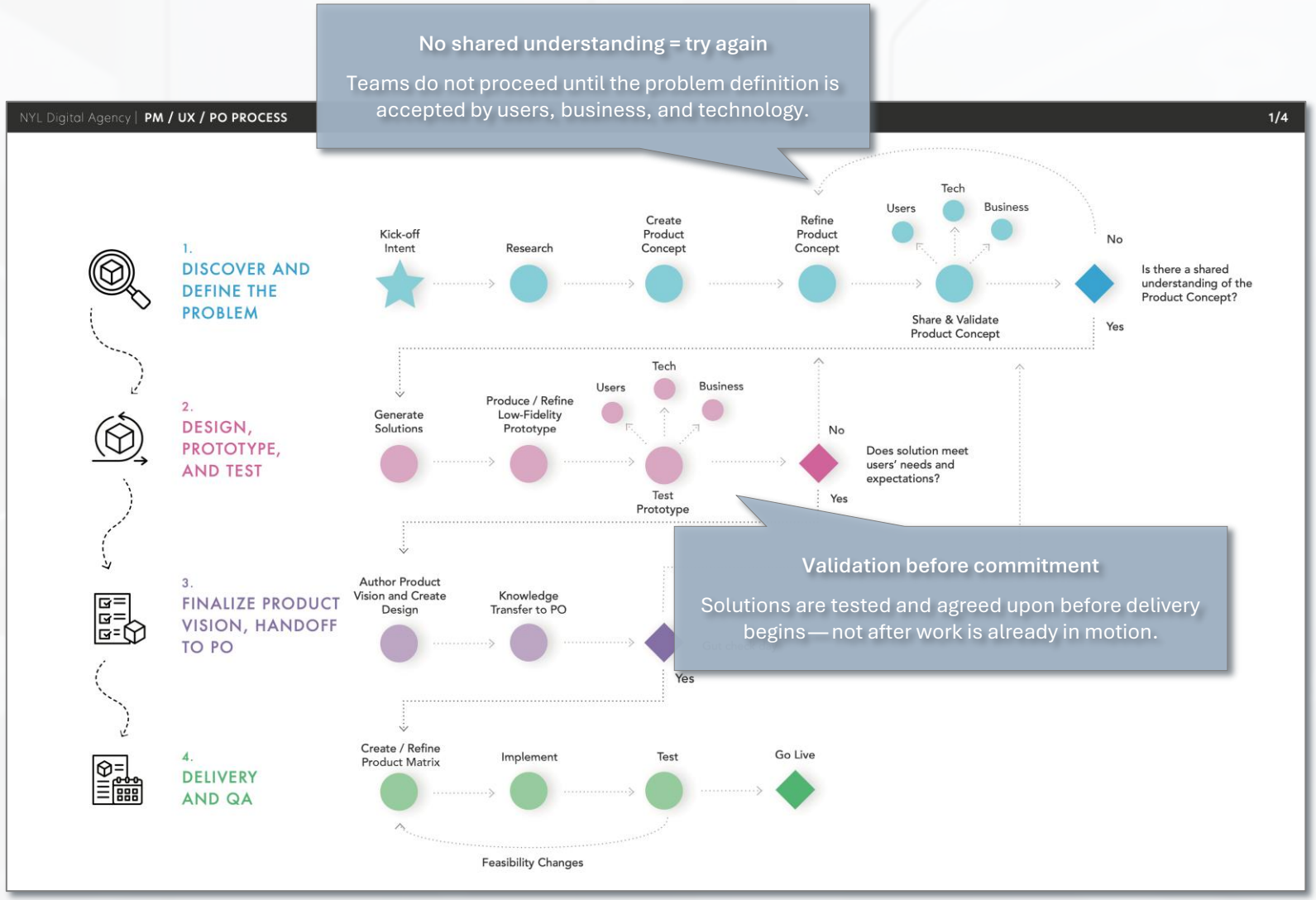
What to notice

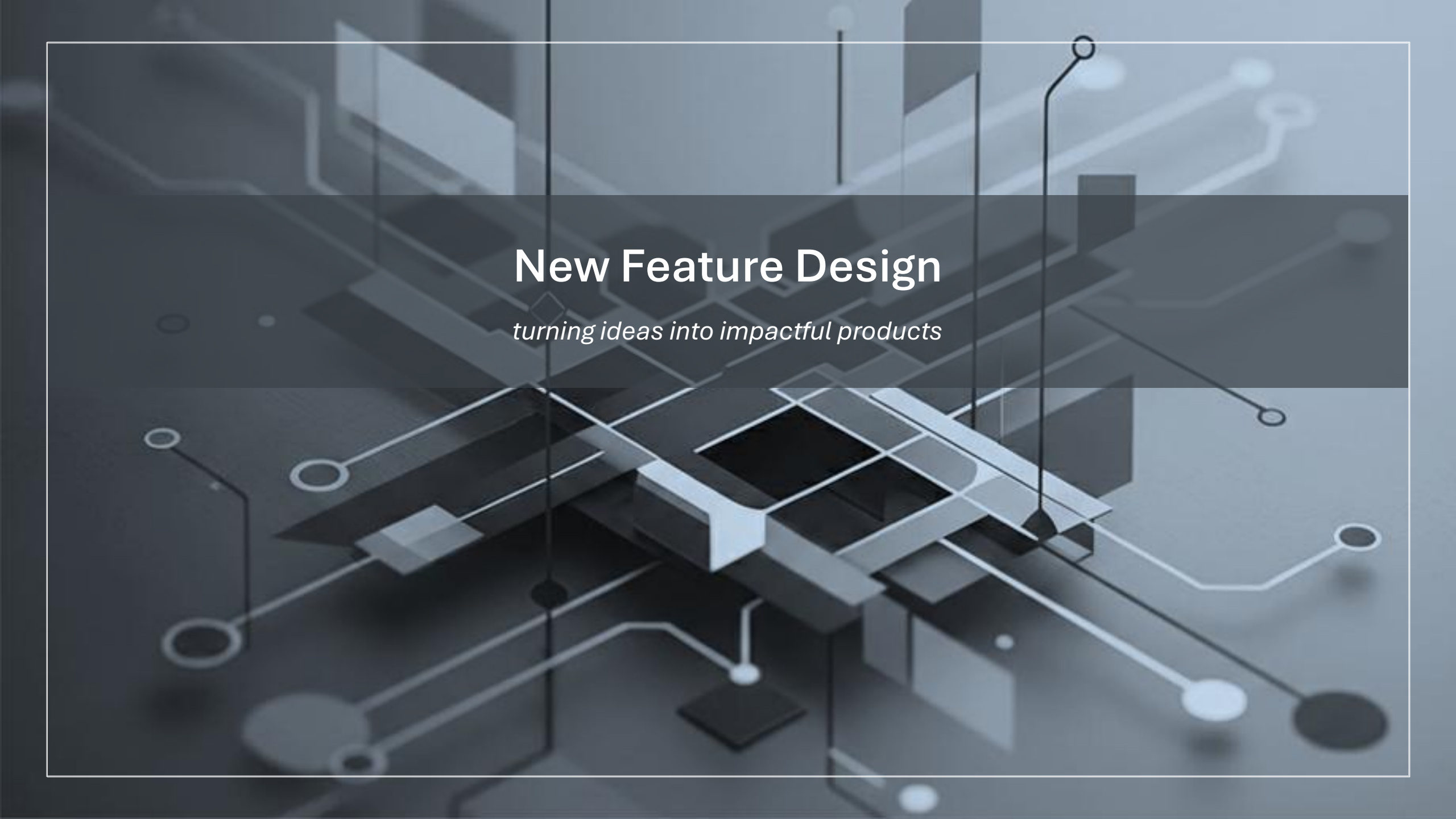
- Progress is gated by shared understanding, not phase completion
- Misalignment is surfaced early, when it is cheapest to resolve
- Expensive work starts only after assumptions are validated

Case
How Product Work Gets Done

Company
New York Life

Role
Co-Ideator & Co-Author





New Feature Design

turning ideas into impactful products

Lead Capture and Tracking

Role: Design Lead & Project Author

Problem

Lead capture capabilities evolved through **disconnected tools and workflows**, each optimized for a specific source or role. Different entry points encoded different rules, validation, and downstream behavior—resulting in manual work, inconsistent data, and limited visibility into what drives outcomes.

Principle

There are many valid ways to capture a name. Once captured, it should move through a **single, coherent lifecycle**—regardless of source.

What This Addresses

This work addresses fragmented lead capture and identity workflows by unifying how names enter, move through, and are acted on across the organization, **enabling consistent execution**, higher data quality, and end-to-end visibility from source to client.

Scope

- The work builds on evolving the Lead and Client foundation and demonstrates how a unified lifecycle can support **multiple capture patterns while staying extensible over time**.

WHY THIS MATTERS

Fragmented lead capture doesn't just slow people down — it prevents the organization from learning what works in practice. When entry points behave differently, effort is duplicated, data quality degrades, and outcomes can't be reliably compared.

A unified lifecycle maintains consistency: multiple capture methods, one system of logic, ownership, and analytics.

WHAT TO NOTICE

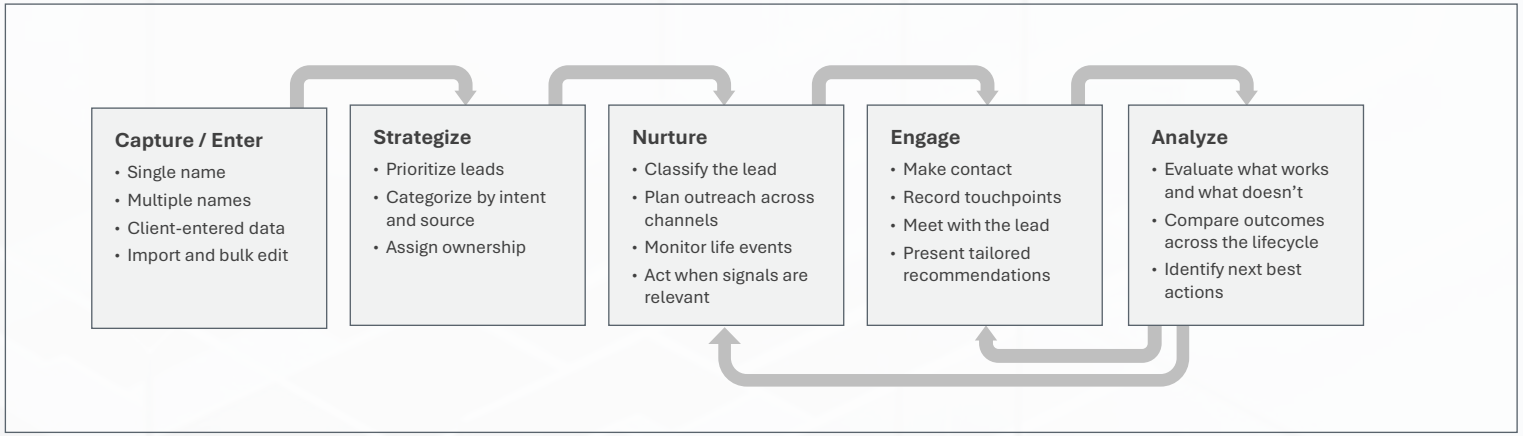
- Same lifecycle, different behavior
- The Before and After diagrams show the same stages — the difference is coherence, not scope

CASE
Lead Capture and Tracking

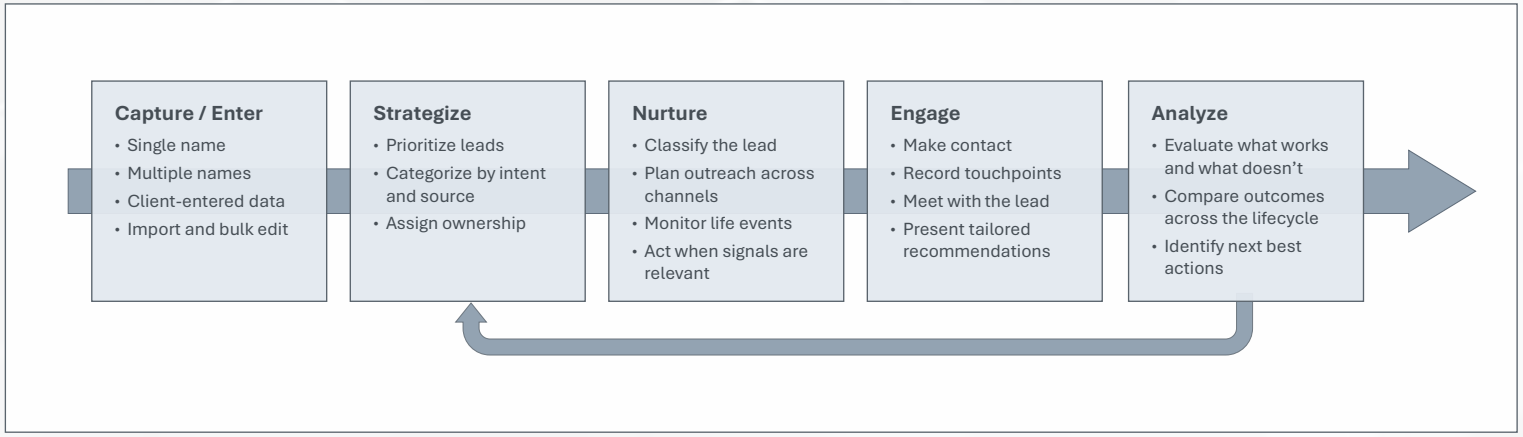
COMPANY
New York Life

ROLE
Design Lead & Project Author

Before: Iteration happens in isolated pockets



After: Learning feeds back into strategy



Lead Capture project was architected and realized through rapid iteration

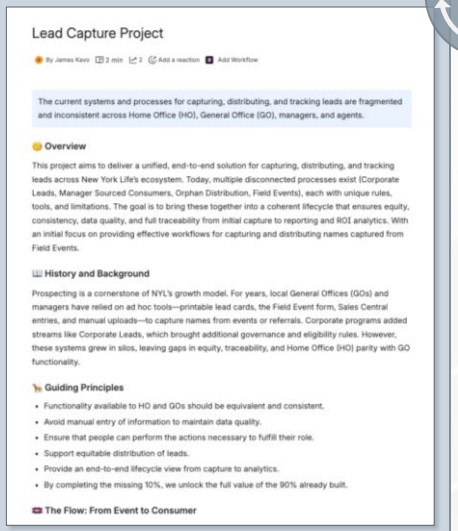
CASE
Lead Capture and Tracking

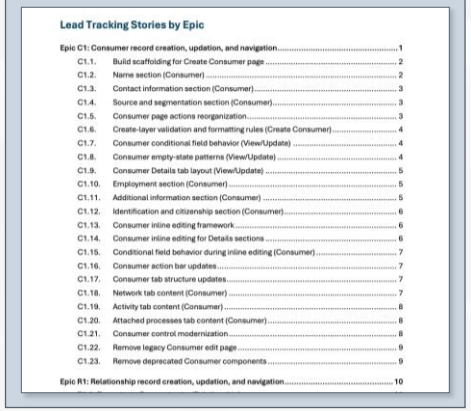
COMPANY
New York Life

ROLE
Design Lead & Project Author

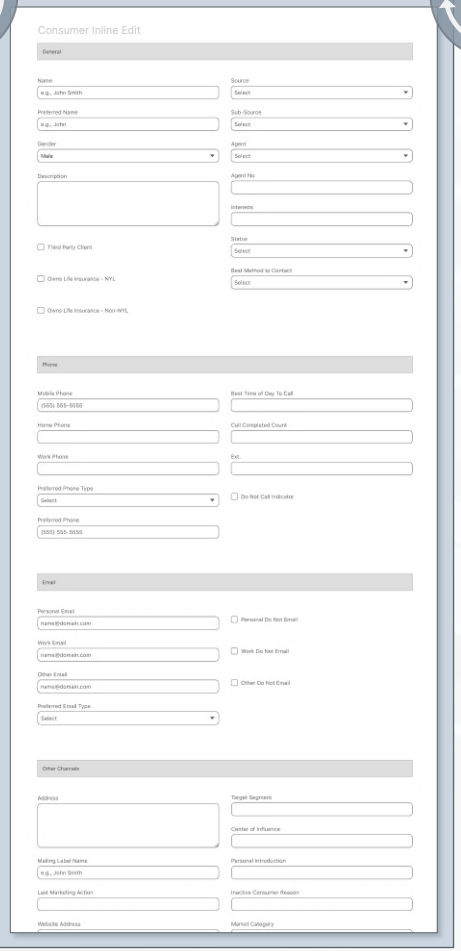
Rapid iteration with generative AI

CONCEPT

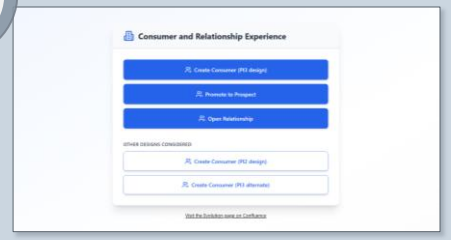


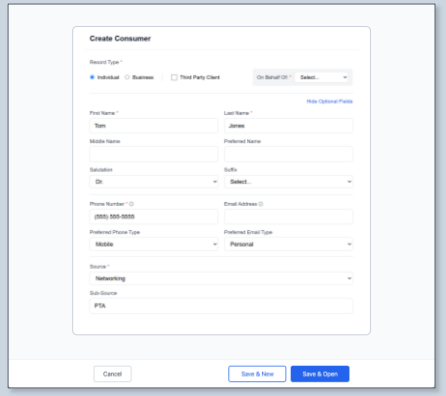


LAYOUT



PROTOTYPE





Making Follow-Up Actionable | Nurture List

Role: Design Lead / Product Manager

Problem

Agents don't fail to follow up because they lack data—they fail because follow-up isn't **organized into daily work**. Names worth contacting exist across leads, clients, events, and reminders, but without a practical way to act on them consistently, outreach depends on memory, personal notes, or external tools. As a result, **valuable opportunities rot on the vine** after initial identification.

Constraint

This was not a marketing automation or campaign problem. Those tools already existed and were optimized for criteria, channels, and batch execution—not for the day-to-day reality of agents making individual outreach decisions. Any solution had to **work for agents of all tenure**, integrate cleanly with compliance requirements, and **fit naturally into existing workflows** without adding process overhead or duplicating functionality.

Leverage Points

- Treat follow-up as first-class work rather than metadata buried in records
- **Centralize outreach** into a single, actionable list populated from across the platform
- Make future intent explicit through scheduling instead of relying on memory
- **Reuse existing platform capabilities** to reduce friction and speed adoption

Outcomes

- Agents have a clear, actionable daily plan for outreach
- Follow-up happens more **consistently without additional process** or tooling
- Less work is lost due to context switching or forgetfulness
- New and experienced agents are equally **supported in maintaining outreach momentum**

Why this matters

Nurture List turns fragmented names, reminders, and suggestions into a single, actionable to-do system that supports consistent outreach without creating new process overhead.

What to notice

- Existing platform capabilities are reused, lowering cost and adoption friction
- Scheduling makes future intent explicit, so outreach doesn't rely on memory
- Activity is reinforced through feedback loops, not manual tracking
- Suggestions nudge behavior without automating or removing agent judgment

Case

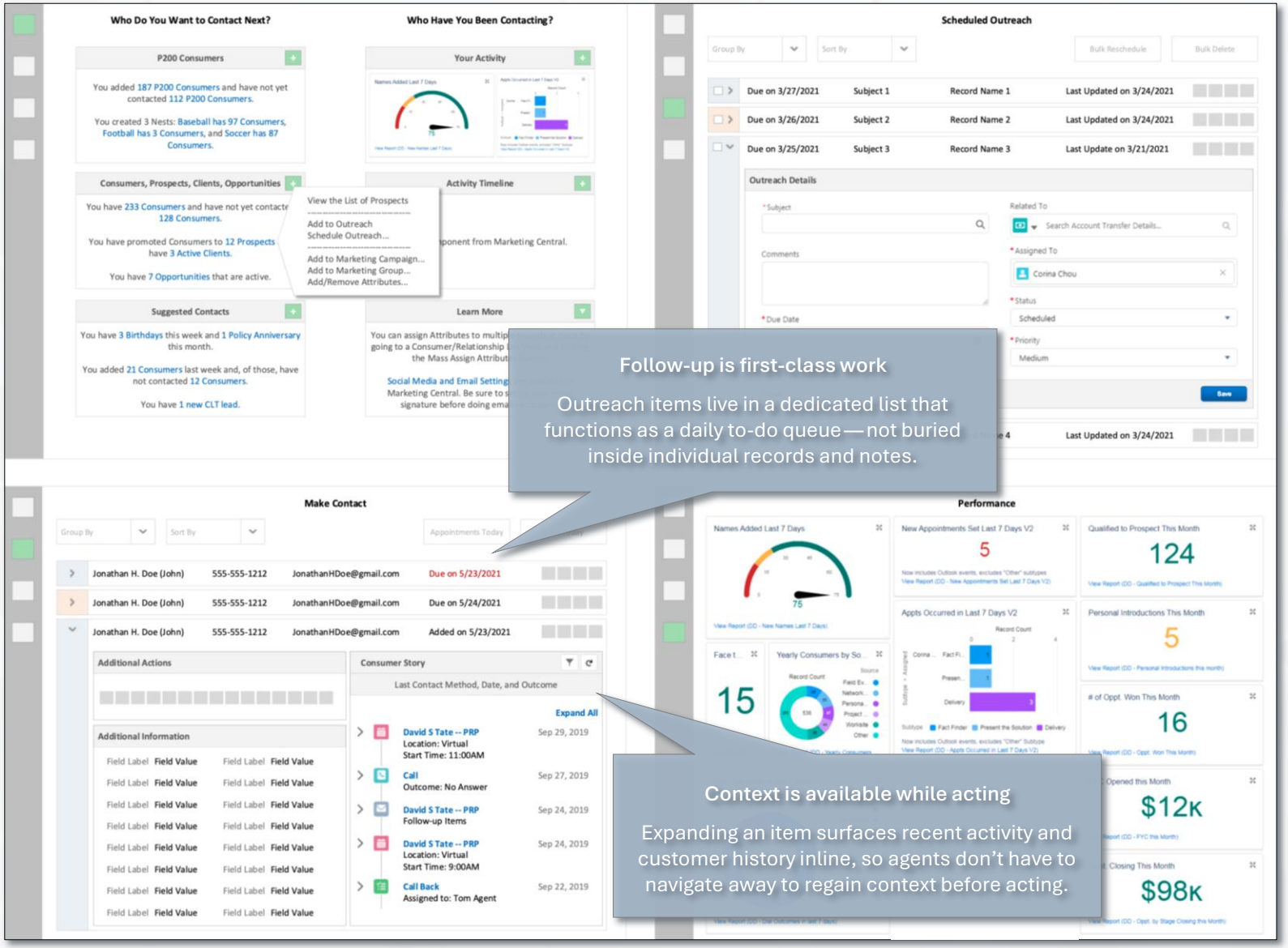
Making Follow-Up Actionable | Nurture List

Company

New York Life

Role

Design Lead / Product Manager



thank you

james@ **jameskavo.com** /resume /portfolio /linkedin