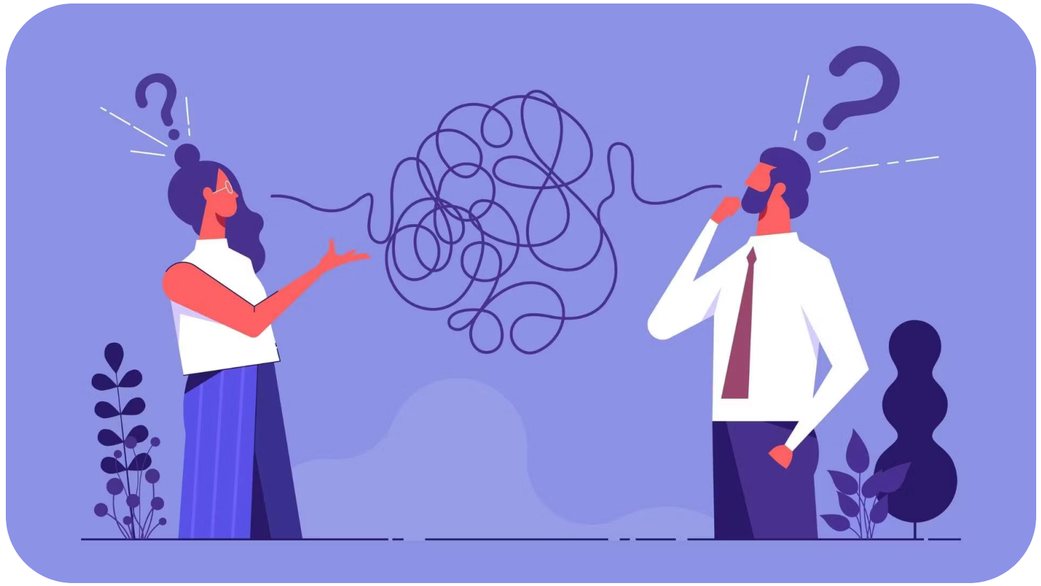
VAANTALK

KEY TO SUCCESS

PROTOTYPING & COMMUNICATION

HAVE YOU EVER BEEN HERE?



YOU SPENT DAYS BUILDING A FEATURE, ONLY TO DISCOVER THAT THE STAKEHOLDER MEANT SOMETHING SLIGHTLY DIFFERENT.

- Why? Because the verbal or even written specs were interpreted in different ways.

YOU LEFT A MEETING THINKING THE TEAM *WAS* ALIGNED...

- But later found out that people had very different understandings of the approach or technical nuances.

YOU IMPLEMENTED A SOLUTION YOU THOUGHT *WAS* SMART—

- Only to learn it didn't account for a critical edge case, because it wasn't visible or discussed early on.

THESE SITUATIONS ARE COMMON

IN MOST CASES, THE PROBLEM ISN’T TECHNICAL. IT’S COMMUNICATION.
THAT’S WHERE PROTOTYPING — ESPECIALLY VISUAL PROTOTYPING —
BECOMES A SECRET WEAPON.

Why visual Prototyping Matters

1. Increases the ways you can get your point across

In a typical IT workflow, most communication happens through verbal discussions or written documents — specs, tickets, messages.

But one powerful layer is often missing: visual communication.

Prototypes fill that gap. They make abstract ideas visible. Whether it’s a flowchart, a wireframe, or a simple diagram, visual prototyping helps align thinking faster and more effectively than words alone.

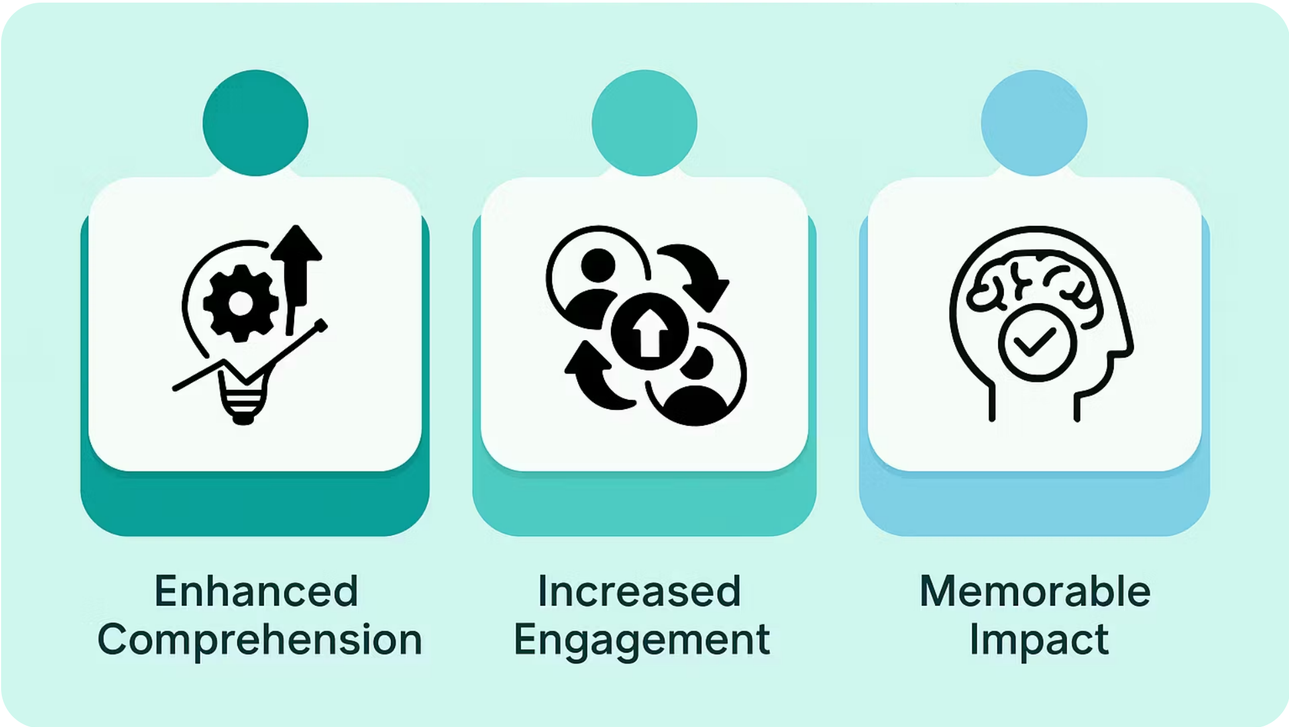
Why visual Prototyping Matters

1. Increases the ways you can get your point across

In a typical IT workflow, most communication happens through verbal discussions or written documents — specs, tickets, messages.

But one powerful layer is often missing: visual communication.

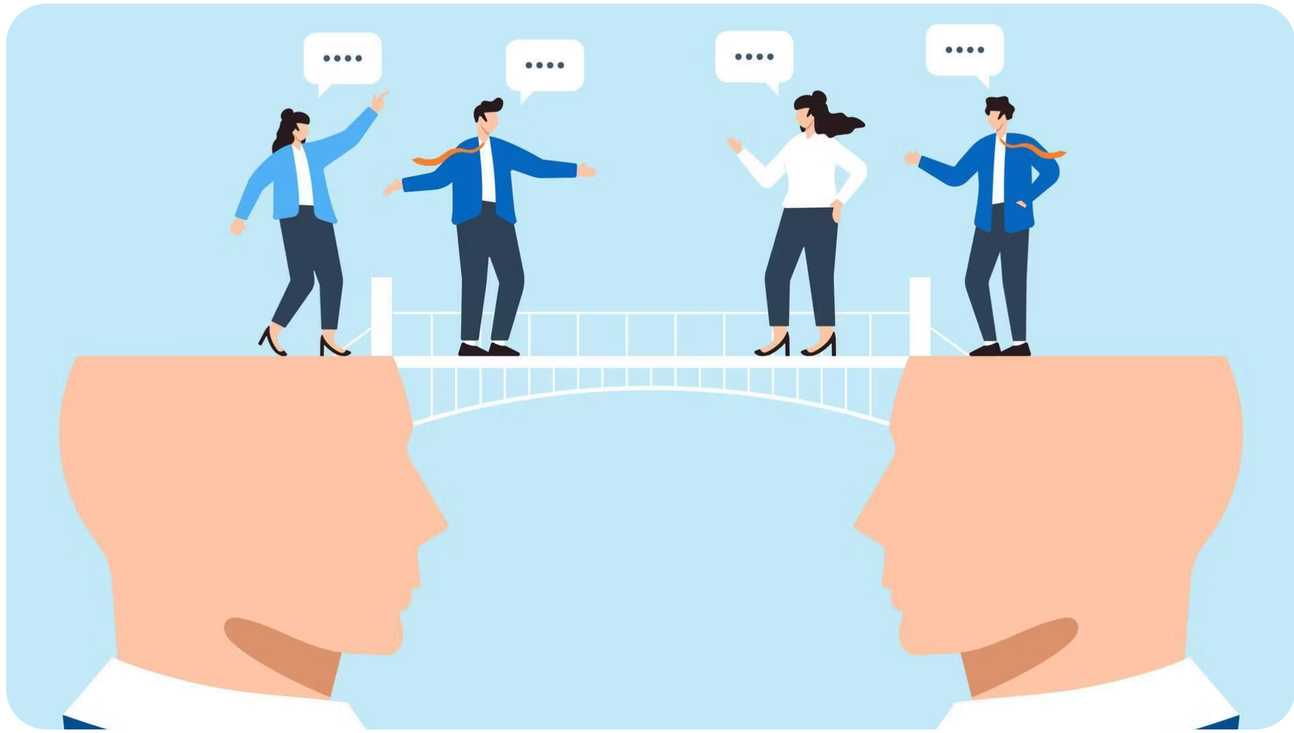
Prototypes fill that gap. They make abstract ideas visible. Whether it’s a flowchart, a wireframe, or a simple diagram, visual prototyping helps align thinking faster and more effectively than words alone.



Why visual Prototyping Matters

2. Bridges the gap between tech and non-tech

PMs, designers, and engineers often speak different ‘languages.’ A visual prototype is a shared language — it keeps everyone on the same page



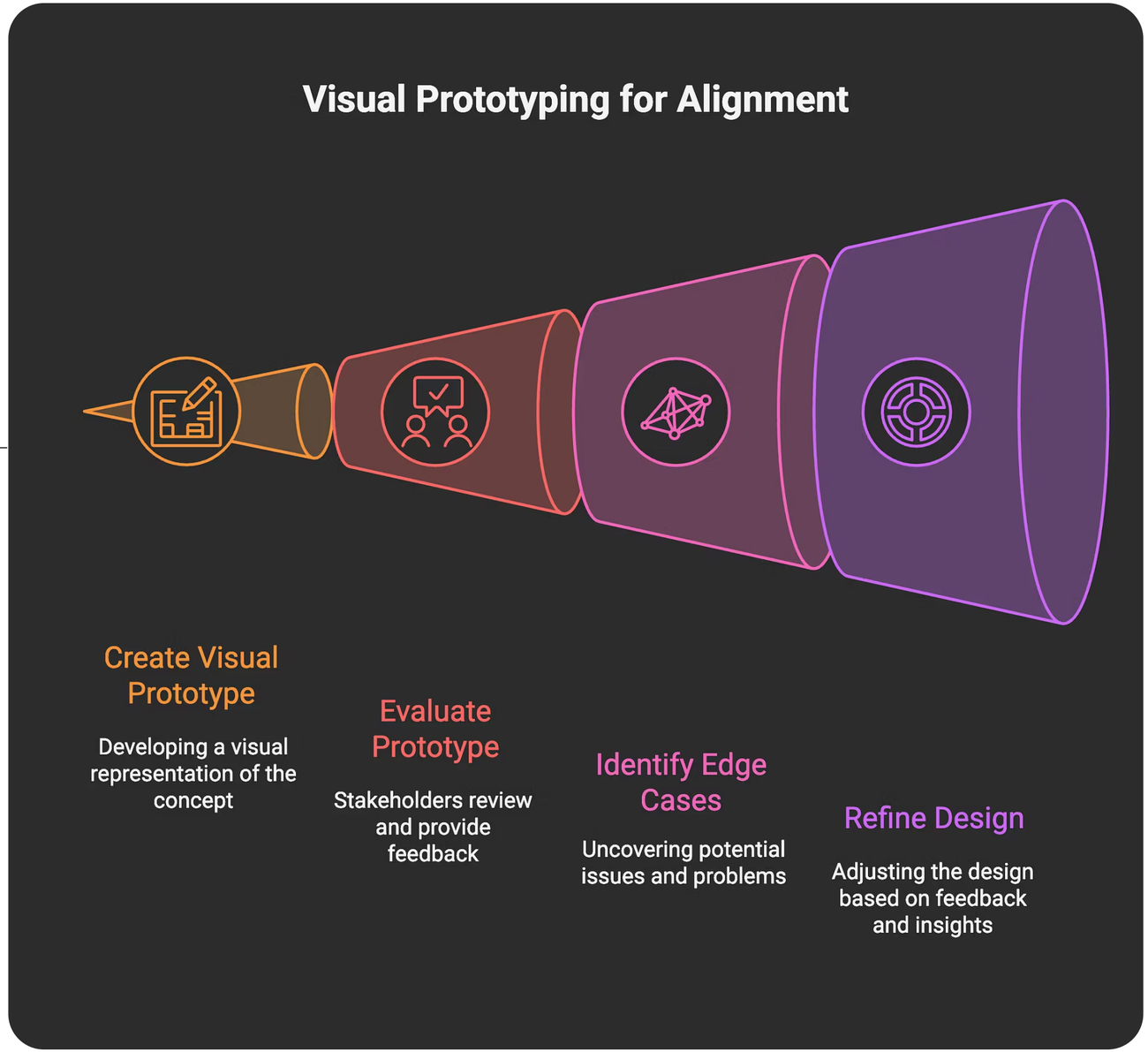
Why visual Prototyping Matters

3. Helps plan smart, align early, and avoid costly rework

Clean architecture is great — but it doesn't help if we're building the wrong thing.

Prototyping helps validate the idea early, while it's still cheap to change. A quick diagram or flow can spark alignment, uncover edge cases, and reduce rework before a single line of code is written.

Changing a box in Miro takes seconds. Refactoring the core of an implemented feature takes hours — or days. That's why early prototyping isn't just helpful — it's efficient.



Real Stories

1. Refactoring Miscommunication

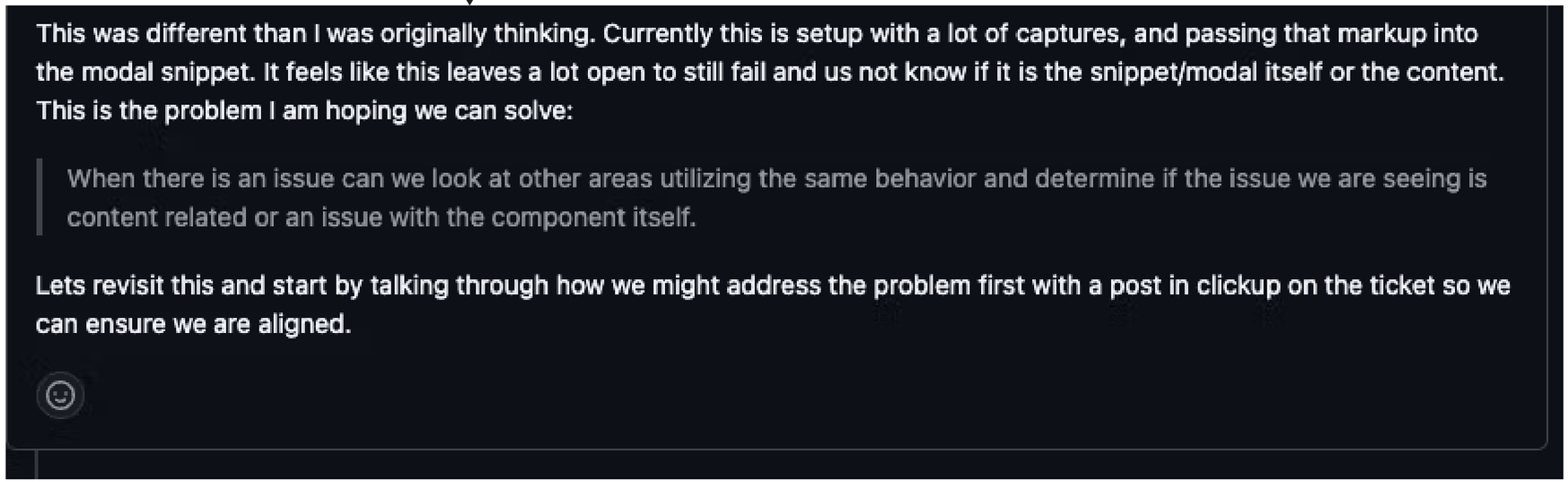
1

I was assigned a ticket where I received only general information about the refactoring goal, without a clear explanation of the overall vision or how it should be resolved.

We have a lot of different modals and popups throughout the project where each implements one standard (or lack of) accessibility and behavior. I believe we can group most of them to use the same logic and same accessibility helping tools to improve the general experience. Also improving the visual experience having the same effects for the background overlay, opacity animation, etc

2

After I had spent a significant amount of time working on it, at the final stage I received a message showing that my approach was not aligned with the original expectations.



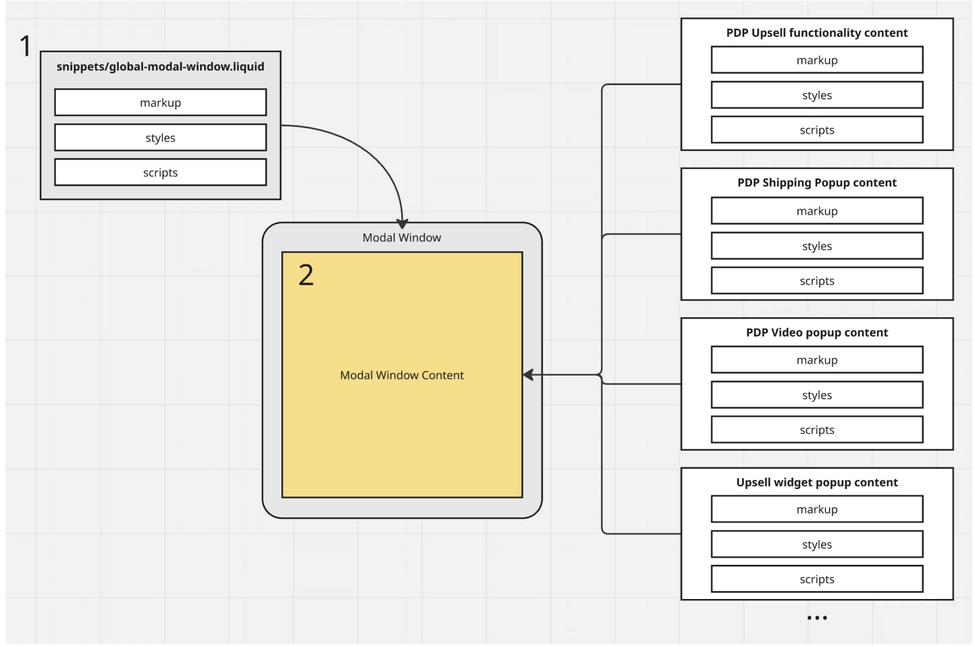
Real Stories

1. Refactoring Miscommunication

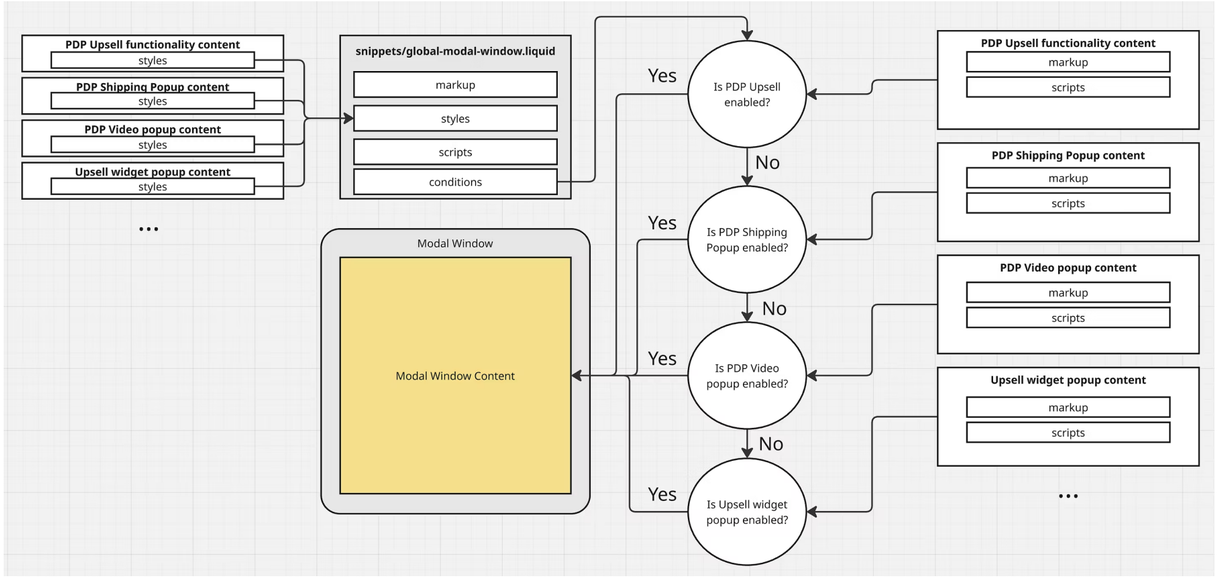
3

Before starting the discussion, I decided to prepare flowcharts presenting different approaches that could be implemented.

1. Centralized Modal Logic, Flexible Content

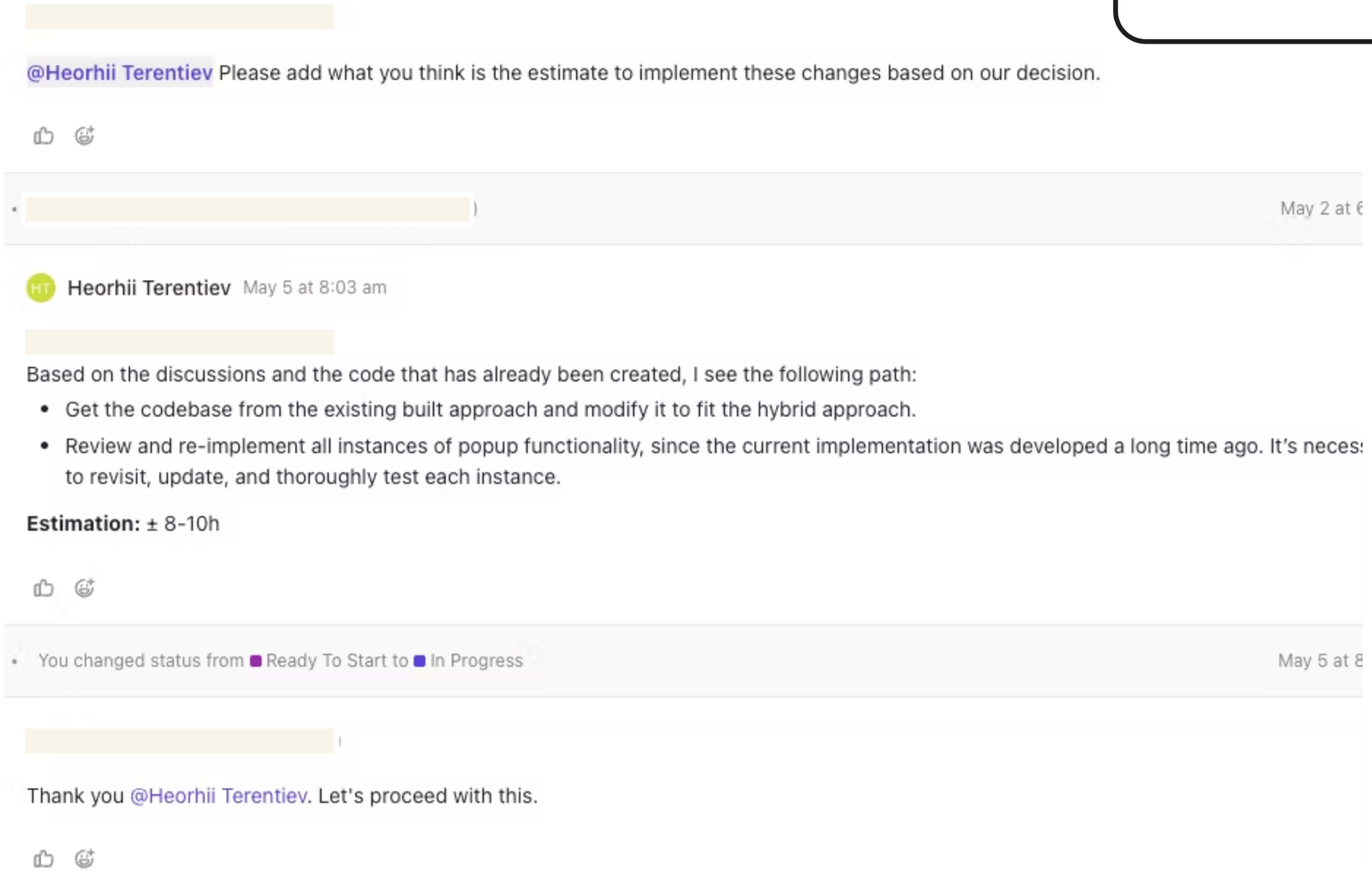


2. Global Modal with Conditional-Based Content Markup



Real Stories

1. Refactoring Miscommunication



4

As a result, we agreed to implement the third approach, which was a mix of the two previously discussed approaches, and we estimated the work accordingly.

5

However, this new decision required an additional **8–10 hours of work**. This was a case where extra time was spent due to misaligned development.

Conclusion

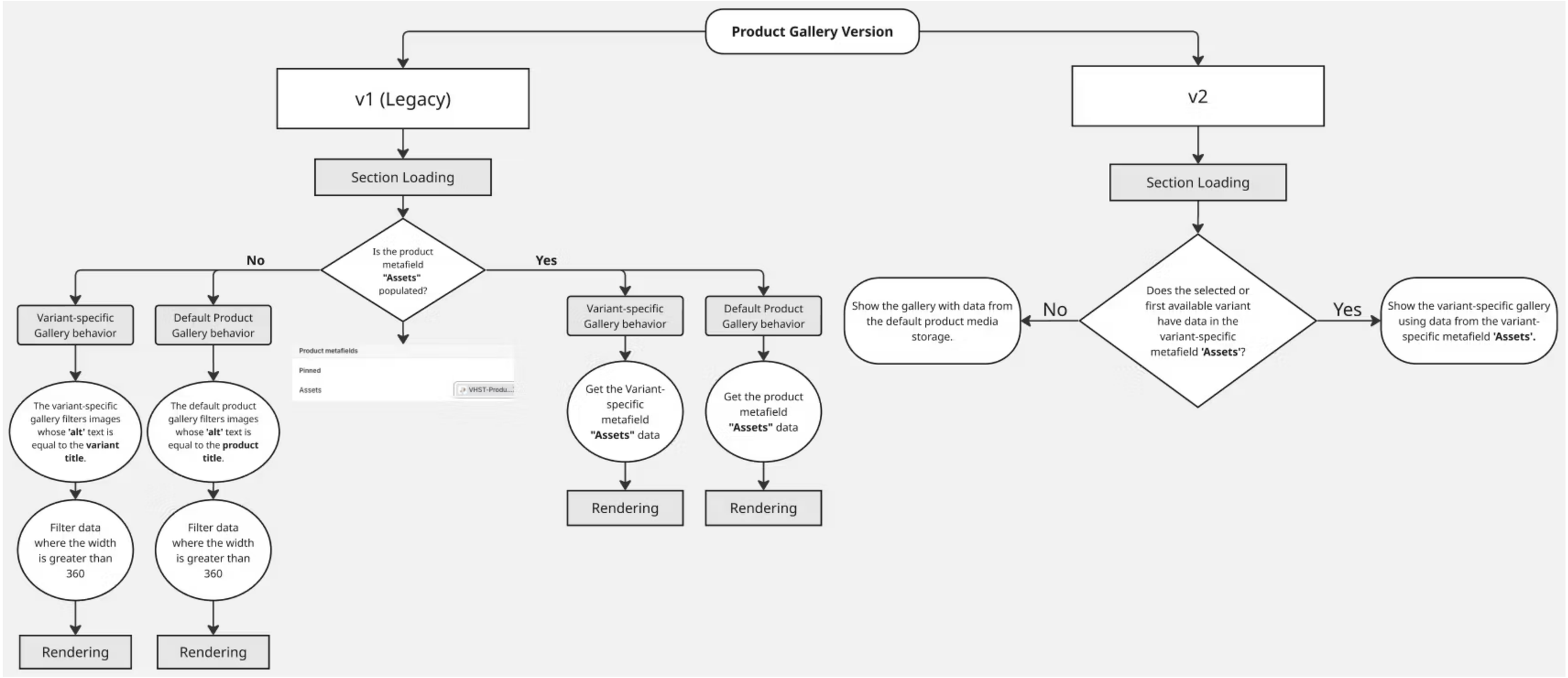
A properly aligned approach is not just a “nice to have” — it is an efficient way to develop features and avoid unnecessary rework.

Real Stories

2. Beauty brand – Product Assets Migration

A one-time investigation and visualization made it easier to explain the overall picture of what's happening, helping to:

- Understand the current state of functionalities for all stakeholders
- Estimation & planning, and evaluation of the level of effort (LOE) for migration.
- Simplify the handover process

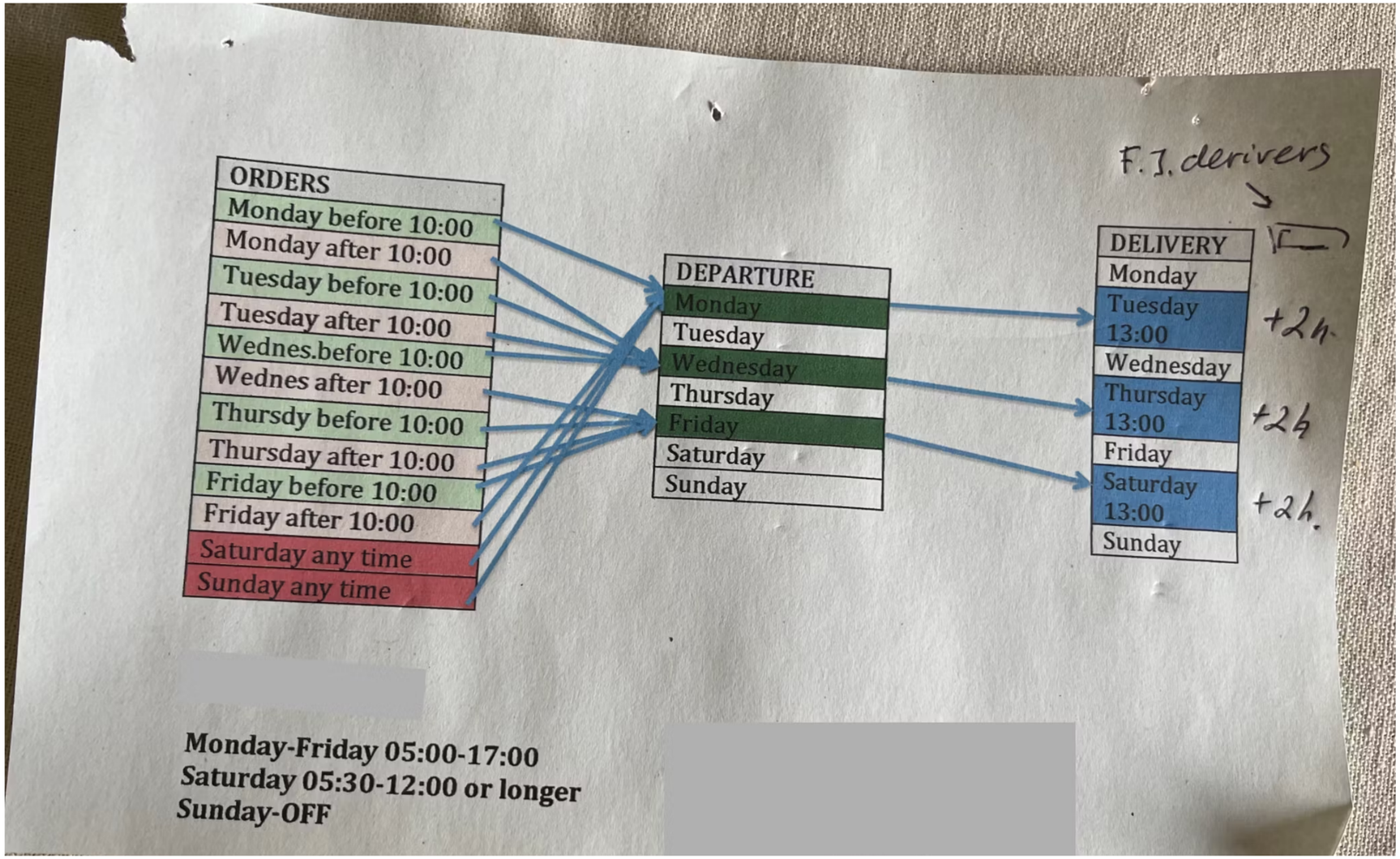


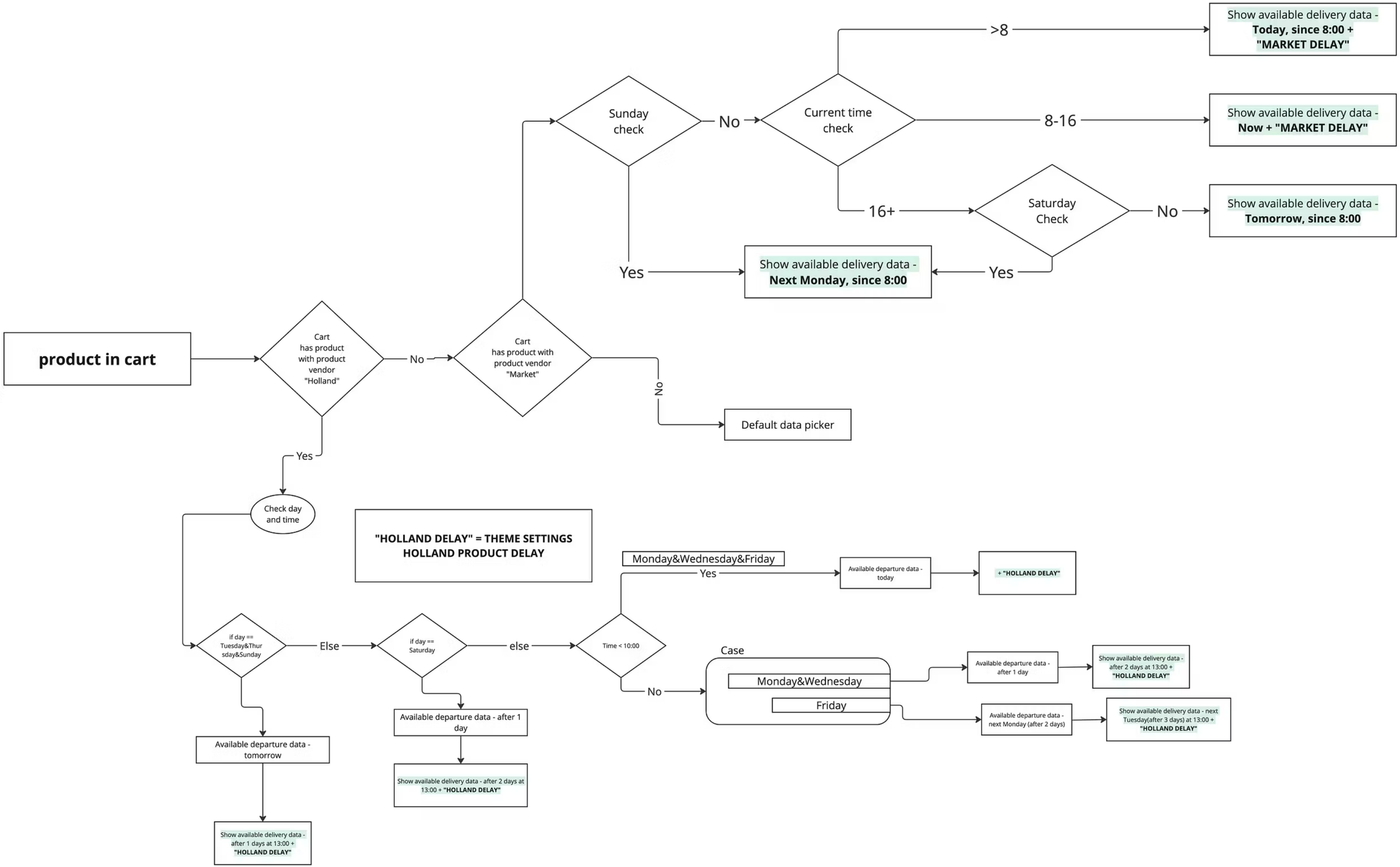
Real Stories

3. Set Up Custom Delivery System

Initial prototyping using flowcharts helps identify and think through all edge cases even before coding begins.

This approach helps structure the development process, allowing developers to focus on the necessary details at each stage of building. In the initial step, they think through the logic, and during coding, they can focus solely on the functionality of the code rather than its fundamental logic.

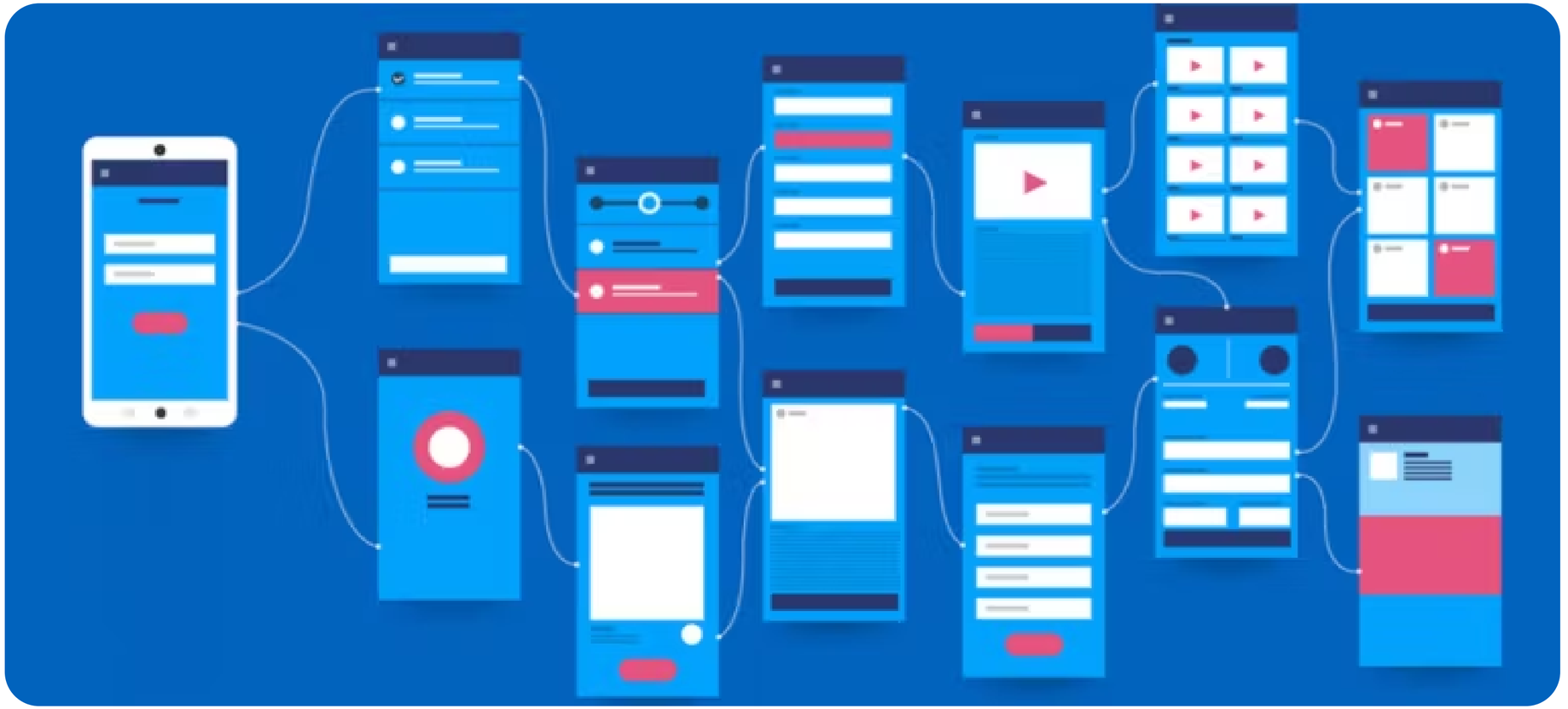


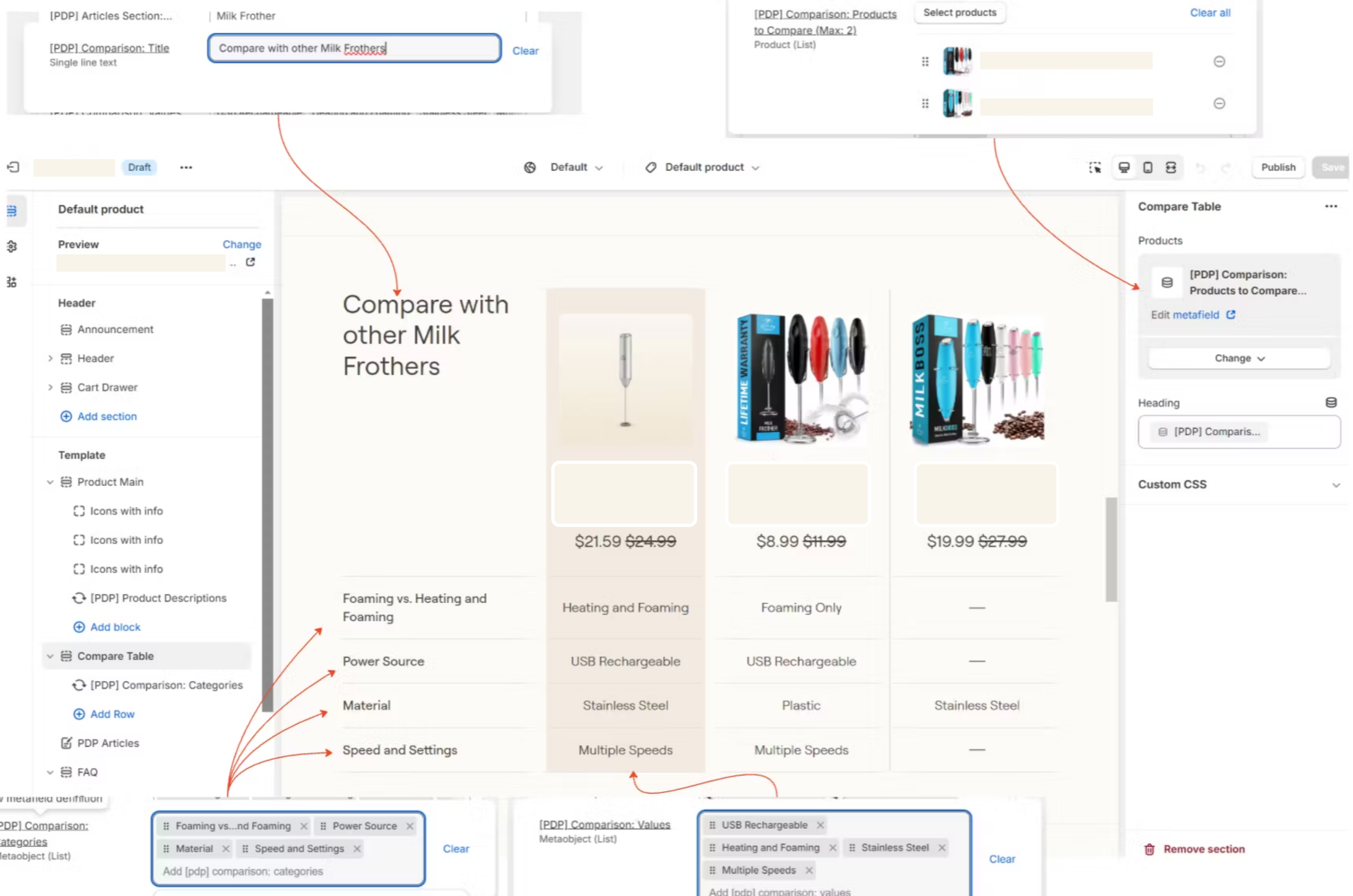

product in cart
Cart has product with product vendor "Holland"
No
Yes
Cart has product with product vendor "Market"
No
Default data picker
Sunday check
No
Yes
Current time check
>8
8-16
16+
Saturday Check
No
Yes
Show available delivery data - Today, since 8:00 + "MARKET DELAY"
Show available delivery data - Now + "MARKET DELAY"
Show available delivery data - Tomorrow, since 8:00
Show available delivery data - Next Monday, since 8:00
Check day and time
"HOLLAND DELAY" = THEME SETTINGS HOLLAND PRODUCT DELAY
if day == Tuesday&Thursday&Sunday
Else
Available departure data - tomorrow
Show available delivery data - after 1 days at 13:00 + "HOLLAND DELAY"
if day == Saturday
else
Available departure data - after 1 day
Show available delivery data - after 2 days at 13:00 + "HOLLAND DELAY"
Time < 10:00
Monday&Wednesday&Friday
Yes
Available departure data - today
+ "HOLLAND DELAY"
No
Case
Monday&Wednesday
Friday
Available departure data - after 1 day
Show available delivery data - after 2 days at 13:00 + "HOLLAND DELAY"
Available departure data - next Monday (after 2 days)
Show available delivery data - next Tuesday(after 3 days) at 13:00 + "HOLLAND DELAY"

Real Stories

4. Simple UI Guide

Even creating a simple visual guideline of an implemented feature helps others quickly understand the concept without having to dive into documentation or other detailed resources.




[PDP] Articles Section:...
Milk Frother
[PDP] Comparison: Title
Single line text
Compare with other Milk Frothers
Clear
[PDP] Comparison: Products to Compare (Max: 2)
Product (List)
Select products
Clear all
Draft
Default
Default product
Publish
Save
Default product
Preview
Change
Header
Announcement
Header
Cart Drawer
Add section
Template
Product Main
Icons with info
Icons with info
Icons with info
[PDP] Product Descriptions
Add block
Compare Table
[PDP] Comparison: Categories
Add Row
PDP Articles
FAQ
Compare with other Milk Frothers
$21.59 $24.99
$8.99 $11.99
$19.99 $27.99
Foaming vs. Heating and Foaming
Heating and Foaming
Foaming Only
—
Power Source
USB Rechargeable
USB Rechargeable
—
Material
Stainless Steel
Plastic
Stainless Steel
Speed and Settings
Multiple Speeds
Multiple Speeds
—
Compare Table
Products
[PDP] Comparison: Products to Compare...
Edit metafield
Change
Heading
[PDP] Comparis...
Custom CSS
Remove section
[PDP] Comparison: Categories
Metaobject (List)
Foaming vs...nd Foaming
Power Source
Material
Speed and Settings
Add [pdp] comparison: categories
Clear
[PDP] Comparison: Values
Metaobject (List)
USB Rechargeable
Heating and Foaming
Stainless Steel
Multiple Speeds
Add [pdp] comparison: values
Clear

Summary

When we think about prototyping, we often imagine polished UI mockups.
But it can be as simple as drawing boxes and arrows.

In Miro, we can map a feature flow, add notes for edge cases, and instantly create shared understanding.
The PM sees it. The designer sees it. The developer starts thinking about implementation approaches.

That 1–2 hour sketch can prevent days of misaligned development.

Let’s Compare

Advantages

- Makes abstract ideas concrete and easy to visualize
- Aligns decisions and reduces miscommunication about implementation approaches
- Helps identify edge cases early in the process
- Makes it easier to hand off projects, tasks, or features

vs.

Challenges

- Might seem like “extra work” at first
- Can be skipped under time pressure
- Requires time and tools that not all team members are familiar with