# 📄 Delivery Reflection – Summary View

Prepared by: [Your Name]

Role: Lead Developer

Audience: Planning & Management Group

Date: 2025-07-16

## 👋 Purpose

As a fresh pair of eyes on the team, I’ve been gathering insights from recent 1:1s, team observations, and delivery experience. This short summary highlights a few recurring themes and small, low-risk opportunities that might help explain why delivery sometimes moves slower than expected.  
  
This isn’t a proposal to overhaul our process — just a friendly contribution toward our shared delivery goals. These are observations, not absolutes, and can be challenged or corrected where needed.

## 🚧 Key Observations

* • Stories are front-loaded and overly rigid

Stories are often created in bulk after analysis and handed to the team with many dependencies. This creates critical paths that make parallel delivery difficult. Stories seem to reach the backlog only once UCD and business analysis are “complete.” → Is our Definition of Ready too strict?

* • Delivery is slowed by shared backend/API

All four apps depend on the same backend, and while work is ongoing to deduplicate this, there’s still no clear API ownership. One team’s changes can unintentionally affect others.

* • Blockers surface too late

Teams don’t always escalate blockers early — sometimes due to unclear ownership or confidence. Having a Lead Developer role should help support earlier intervention.

* • Documentation is too heavy and disconnected

Developers prefer simple, in-repo docs (e.g., markdown) over outdated or scattered Confluence pages. Anything technical or architecture-related should live closer to the codebase.

* • Team capability is uneven, and knowledge is at risk

Some critical knowledge is held by contractors who may leave later this year. Newer team members can be hesitant to ask for help or escalate issues.

* • Team divides are subtle but real

There’s a friendly but real divide between Team1/2 and Team3. The naming of Slack channels may reinforce this separation, even if people remain collaborative.

* • Too many Slack channels

It’s hard to track conversations across feature-specific and team-specific channels. → Could we try including all devs in feature channels to improve visibility and “osmosis learning”?

## ✅ What’s Working Well

* • UI design walkthroughs – create excellent shared understanding
* • Spikes – effective at identifying unknowns early
* • UI team collaboration – strong use of Slack to solve problems
* • Code quality – newer UI apps are praised for their clarity and decisions
* • UI tickets – visual, clear, and easier to pick up than backend ones

## 🔄 Role of a Lead Developer

As I grow into the Lead Developer role, my aim is to help bridge the gap between teams, tech, and delivery. That means reinforcing shared understanding, surfacing blockers early, and supporting confidence and momentum across the team — without adding unnecessary layers of process.

## 💡 Opportunities to Explore (Small, Measurable Changes)

|  |  |
| --- | --- |
| Opportunity | Why It Might Help |
| Lightweight Definition of Ready | Avoids front-loading and encourages earlier dev input |
| API stewardship / ownership | Reduces cross-app breakages and builds trust |
| “Contract stories” for UI ↔ API | Allows FE/BE to work in parallel against shared contract |
| Review story structure | Reduce critical paths and unblock parallel work |
| Smarter documentation | Bring docs closer to the code for clarity and relevance |
| Pairing on key stories | Upskills team, reduces single-point reliance |
| Earlier blocker visibility | Helps leadership plan support more proactively |
| Channel rationalisation | Reduce Slack noise and improve cross-team awareness |

## 🧪 Case Study Timeline

📍 Real project example — used to reflect on perception vs timeline

|  |  |
| --- | --- |
| Stage | Date |
| Confluence Page Created | Feb 26 |
| Requirements Gathering | Feb 26 – Mar 24 |
| Screen Flow Diagram Complete | May 13 |
| Slack Channel Created | June 6 |
| UX & Policy Finalisation | ~July 2 |
| User Stories Written | July 16 |

## 🧱 User Story Options (Exploratory)

These are exploratory — not proposals, but possible models to reflect on.

* Option 1: Vertical Slices

Pros:  
- Testable units of value  
- FE and BE work on the same story  
- Stories should complete within a sprint

Cons:  
- Requires mindset and workflow shift across teams  
- Difficult without full cross-functional buy-in

* Option 2: Contract Story Model

Pros:  
- Less disruptive to existing workflow  
- FE and BE can deliver independently to the same contract  
- Final contract story integrates and verifies both ends

Cons:  
- Still requires coordination and ownership to maintain contract integrity

## 🤝 Final Thought

This is just one view, shared with humility and curiosity. I’m excited to work with you all to explore a few of these ideas in a low-risk, collaborative way — and see what helps us move with more confidence and consistency.  
  
Thanks for reading,  
[Your Name]