FRAMEWORK 4

Pre-Mortem Analysis

> **PURPOSE:** *Identify the risks in a decision before you commit to it by imagining it has already failed and working backward to find out why.*

Tentative decision: *One sentence*	**Facilitator:** *Enter name*	**Date:** *MM/DD/YYYY*	**Time horizon:** *6 months / 1 year*

HOW TO USE THIS FRAMEWORK

Make the tentative decision, fast-forward in time and imagine it failed, work backward to identify what went wrong, then adjust the decision or create mitigation plans.

Use it for major technical decisions, significant organizational changes, high-stakes projects, and any time optimism bias might be present. It surfaces concerns people hesitate to raise, exposes blind spots, and builds alignment on the challenges ahead.

STEP 1: SET THE SCENE

> "It's *[6 months / 1 year]* from now. We implemented *[decision]*. It was a disaster. What happened?"

STEP 2: INDIVIDUAL BRAINSTORM (5 MIN)

Everyone writes down reasons for failure independently before anyone speaks. This prevents groupthink.

STEP 2: INDIVIDUAL BRAINSTORM — NOTES

STEPS 3–4: SHARE, CATEGORIZE AND ASSESS LIKELIHOOD (15 MIN)

Risk theme	Failure scenario	Likelihood (Low / Med / High)
Technical		
People / team		
Market / customer		
Organizational / political		
Resource / timeline		

Risk theme	Failure scenario	Likelihood (Low / Med / High)

STEP 5: MITIGATION PLANS (HIGH AND MEDIUM RISKS)

Risk	Can we prevent it?	Can we detect it early?	Response if it happens

STEP 6: DECIDE

- ☐ Proceed, with the mitigation plans above
- ☐ Adjust the decision
- ☐ Don't proceed (risk too high)

STEP 6: DECIDE — NOTES

ACTION ITEMS

Owner	Action item	Due date

EXAMPLE: MIGRATE TO MICROSERVICES ARCHITECTURE

Failure scenarios: "We underestimated complexity and took 2x longer than planned." "The team didn't have distributed systems expertise." "Debugging across services became a nightmare." "A partial migration created more problems than it solved."

Mitigations: Add a 50% time buffer to estimates. Send two engineers to microservices training. Invest in observability tools first. Define clear success criteria for each service migration.

> **Pro Tip:** This is a teachable moment approach. Learn from imagined failures before they become real ones.