AI Automation / Workflow Systems

Map how workflows depend on each other — shared data, shared tools, sequencing requirements — prevents cascading failures across systems.
Difficulty: Advanced
Model: GPT-4 / Claude / Gemini
Use Case: System Architecture, Risk Assessment
Updated: May 2026
Why This Prompt Exists
Workflows don’t exist in isolation. They share databases, APIs, queues, and file systems. When one breaks, others cascade — and no one knows why.

You get:

- mystery outages — workflow fails, but the error is in a different system
- cascading failures — one broken workflow takes down five others
- hidden dependencies — no one knows that Workflow B relies on Workflow A
- change anxiety — “if we touch this, what else will break?”
- no visibility into the blast radius of a failure

But dependencies can be mapped:

- data dependencies: Workflow B reads data that Workflow A writes
- tool dependencies: multiple workflows share the same API or database
- sequence dependencies: Workflow B must run after Workflow A
- resource dependencies: workflows compete for limited connections
- time dependencies: Workflow B expects fresh data from Workflow A

Without dependency mapping, you fly blind.

This prompt maps system-wide workflow dependencies.

The Prompt

```
Assume the role of a system architect who maps workflow dependencies.

Your task is to identify how workflows depend on each other across a
system.

Generate:

1. WORKFLOW INVENTORY (from WS-01)
   - List of workflows and their core functions

2. DEPENDENCY TYPES

| Type | Description | Example |
|------|-------------|---------|
| Data | Workflow B reads data created by Workflow A | Lead Scoring
reads leads from Lead Capture |
| Sequence | Workflow B must run after Workflow A | Routing runs after
Scoring completes |
| Tool | Workflows share the same API/database | Multiple workflows
write to Salesforce |
| Resource | Workflows compete for limited capacity | Five workflows
share 10 database connections |
| Time | Workflow expects fresh data within time window | Reporting
runs after daily batch completes |
```

3. DEPENDENCY MATRIX

Source Workflow	Target Workflow	Dependency Type	Criticality	Failure Impact
WF-001	WF-002	Data	High	WF-002 fails
WF-002	WF-003	Sequence	High	Chain breaks
WF-001, WF-004	Salesforce API	Tool	Medium	Rate limits

4. CRITICAL PATH IDENTIFICATION
   - Longest dependency chain: [WF-001 → WF-002 → WF-003 → WF-005]
   - Single points of failure: [workflows or tools that many depend on]

5. CIRCULAR DEPENDENCY DETECTION
   - Any circular dependencies? (A → B → A)
   - Risk level: [High / Medium / None]

6. FAILURE CASCADE ANALYSIS

If This Fails	Which Workflows Fail	Estimated Blast Radius
WF-001	WF-002, WF-003, WF-005	60% of system
Salesforce API	All workflows using Salesforce	80% of system

7. RECOMMENDATIONS
   - Break circular dependencies: [suggestions]
   - Add redundancy for single points of failure: [suggestions]

```
   - Isolate critical paths: [suggestions]

INPUTS:

Workflow inventory (from WS-01):
[PASTE LIST OF WORKFLOWS]

Shared resources:
[E.G., "Salesforce API, Snowflake database, S3 bucket, Redis cache"]

Known sequencing requirements:
[E.G., "Lead scoring must finish before routing starts"]

Historical failures (for cascade analysis):
[E.G., "When WF-001 failed last month, three downstream workflows also
failed"]

RULES:
- Every data write has potential readers (track them)
- Shared tools are single points of failure (document them)
- Circular dependencies are design flaws (fix them)
- Sequence dependencies create critical paths (optimize them)
- Blast radius analysis informs recovery priorities
- Review dependencies quarterly (they change as systems evolve)
```

How To Use It

- Every data write has potential readers — track them explicitly.
- Shared tools are single points of failure — document and monitor them.
- Circular dependencies are design flaws — break them immediately.

- Sequence dependencies create critical paths — optimize those workflows first.
- Blast radius analysis informs recovery priorities — know what to fix first.
- Review dependencies quarterly — they change as systems evolve.

Example Input

Workflow inventory:
"WF-001: Lead Capture (Typeform → Salesforce). WF-002: Lead Scoring (Salesforce → calculate score). WF-003: Lead Routing (Salesforce → assign rep). WF-004: Daily Reporting (Snowflake → email). WF-005: Customer Sync (Salesforce → Snowflake)."

Shared resources:
"Salesforce API (all workflows), Snowflake database (WF-004, WF-005), Email service (WF-004)"

Known sequencing requirements:
"WF-002 must run after WF-001. WF-003 runs after WF-002. WF-005 runs after any Salesforce update."

Historical failures:
"When Salesforce API rate limit exceeded, WF-001, WF-002, WF-003, and WF-005 all failed simultaneously."

Why It Works
Most teams discover dependencies during outages — "why did that break? oh, because it relied on that." That's the worst time to learn.

This framework improves outcomes by forcing:

- dependency type classification (data, sequence, tool, resource, time)
- dependency matrix creation (explicit relationships)
- critical path identification (what drives total system time)

- circular dependency detection (design flaws that cause deadlocks)
- failure cascade analysis (blast radius of each failure)

Failure modes this prevents:

- Mystery outages — no one knows why Workflow B failed (it needed Workflow A)
- Cascading failures — one broken workflow takes down the entire system
- Hidden single points of failure — shared API with no redundancy
- Circular dependencies — Workflow A waits for B, B waits for A (deadlock)

This improves on: Tribal knowledge about dependencies ("I think these are related"). Explicit mapping reveals the actual graph.

Related to: WS-01 (Documenter) for workflow inventory; WS-04 (Optimizer) for fixing bottlenecks.

Build Better AI Systems

Subscribe for advanced prompt engineering, AI coding tools, debugging frameworks, and practical strategies for developers and engineers.

Carefully engineered prompts for people doing real work.

Share this:

- Share on Facebook (Opens in new window) Facebook
- Share on X (Opens in new window) X

See also Workflow Documenter