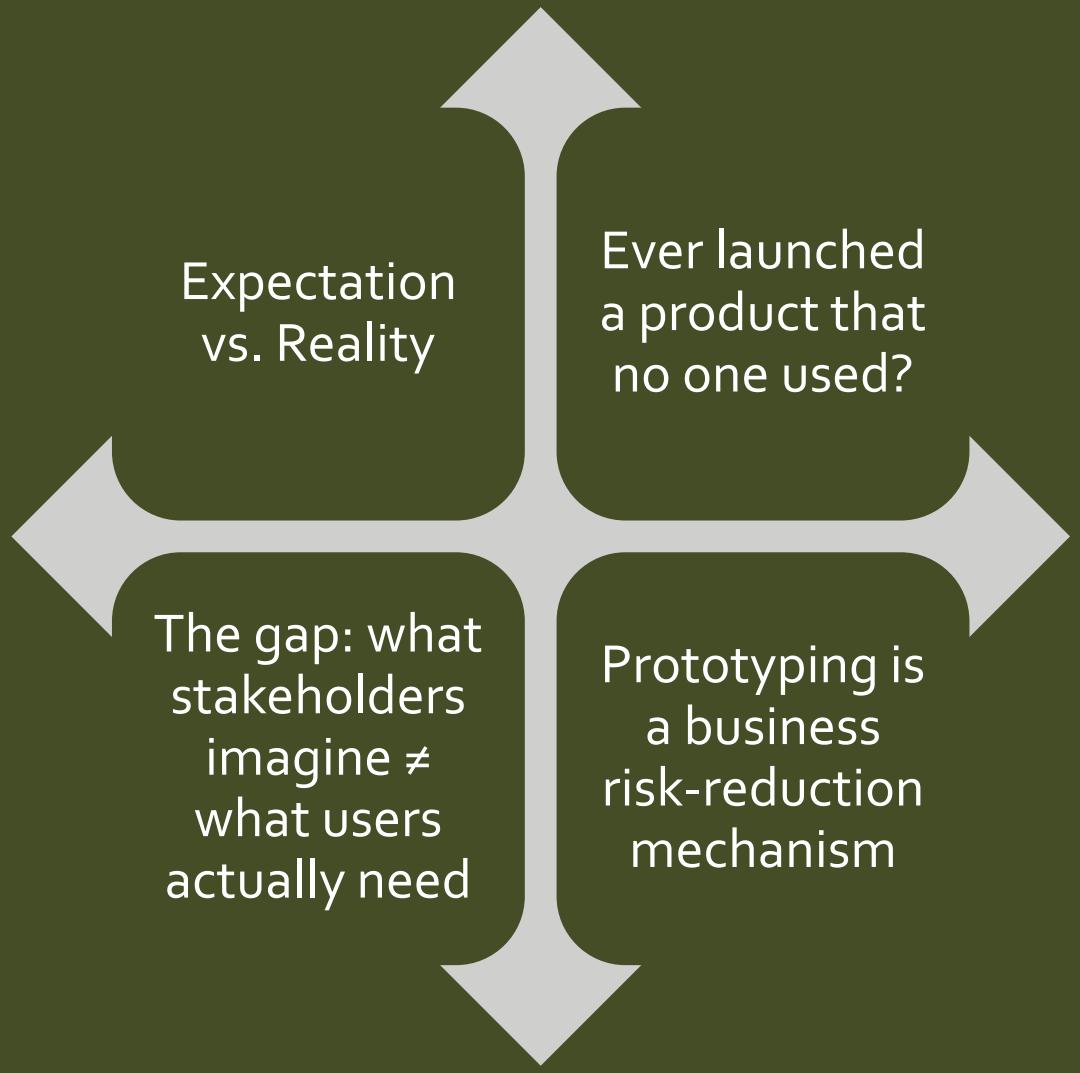
Prototyping: Reducing the Gap Between Expectation and Reality

MIS3506 – Spring 2026

Lavin

The Problem


Expectation vs. Reality
Ever launched a product that no one used?
The gap: what stakeholders imagine ≠ what users actually need
Prototyping is a business risk-reduction mechanism

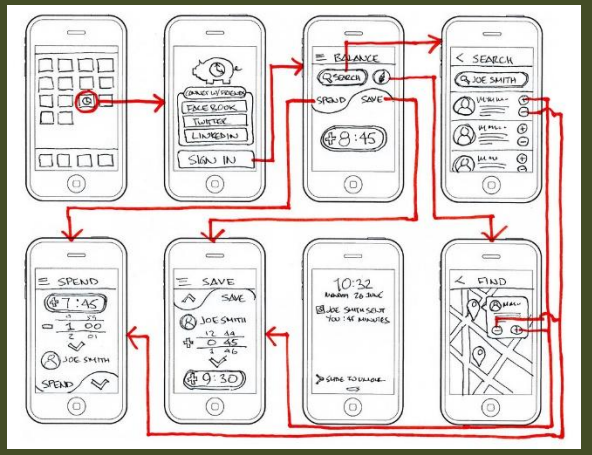
What is Prototyping?

- Definition
  - Building simplified versions of a product to test before full development
  - Tests functionality, usability, and value early
- Fidelity Spectrum
  - Low-fidelity: sketches, wireframes, clickable mockups
  - High-fidelity: near-real interactivity, realistic data flows
- In Data/Analytics Context
  - Dashboard wireframes, report mockups, simulated drilldowns, fake-data interactions

Wireframe vs. Prototype — Key Differences

- Wireframe
  - Static blueprint or skeletal layout
  - Shows structure, hierarchy, content placement
  - No interactivity or functionality
  - Low-fidelity: grayscale, boxes, placeholder text
  - Purpose: Align on layout and information architecture
  - Tools: Paper, Balsamiq, Sketch, Figma (static frames)

- Prototype
  - Interactive simulation of the product
  - Shows user flows, transitions, behaviors
  - Clickable and testable
  - Can be low to high-fidelity
  - Purpose: Test usability, validate interactions, gather feedback
  - Tools: Figma, Adobe XD, InVision, Axure



How Figma Supports Prototyping

Figma: End-to-End Prototyping Platform

Core Prototyping Features:

Interactive hotspots: Link frames to simulate navigation and user flows	Transitions & animations: Add motion between screens	Overlays & modals: Simulate pop-ups, tooltips, dropdowns	Conditional logic: Show/hide elements based on user actions	Device preview: Test on mobile, tablet, desktop in real-time

Figma: Collaboration & Testing

- Collaboration & Feedback:
  - Real-time co-editing with team members
  - Comment threads directly on prototype frames
  - Shareable prototype links with password protection
  - Version history to track iterations
- Testing & Handoff:
  - User testing mode: Observe clicks, paths, time-on-task
  - Developer handoff: Inspect CSS, export assets, view specs
  - Why Figma: Free tier, browser-based, industry standard

Why Prototyping Matters — Part 1

1. Test Functionality Early

Identify navigation, filter, data-flow issues before coding

Example: Finance dashboard—users can't find cash-flow drilldown

2. Ensure Usability

Validate information hierarchy, clarity, interaction ease

Example: Sales ops report—executives need only 3 KPIs above fold

3. Gather Targeted Feedback

Make abstract requirements concrete; align expectations

Example: 'Real-time' becomes 'daily refresh is fine' after walkthrough

Why Prototyping Matters — Part 2

4. Increase Development Assertiveness

Higher confidence = less rework

Example: Two prototype rounds → engineering hits acceptance on sprint 1

5. Reduce Cost and Cycle Time

Catch misfits early; avoid expensive fixes

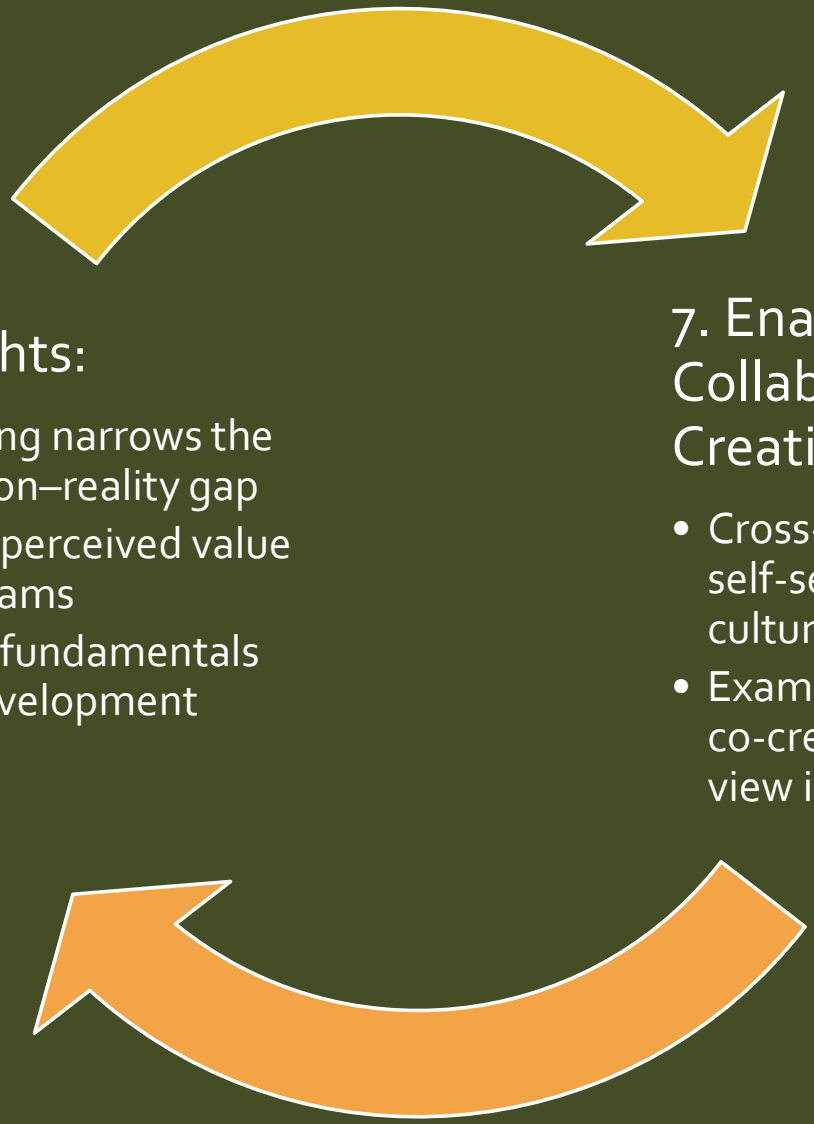
Example: Repositioning 3 charts takes minutes in prototype vs. redoing code

6. Improve Intuitiveness and Tool Fluidity

Test whether flows feel natural; ensure filters aren't overwhelming

Example: Marketing dashboard—prototype reveals need for autosuggest

Why Prototyping Matters — Part 3



Key Insights:

- Prototyping narrows the expectation–reality gap
- Increases perceived value of data teams
- Validates fundamentals before development begins

7. Enable Collaborative Creation

- Cross-team co-design; self-service analytics culture
- Example: Ops and Sales co-create shared pipeline view in prototype

Prototyping vs. Visual Library

- Visual Library
  - Reusable kit: charts, cards, filter patterns, design tokens
  - Speeds up consistent prototypes
- Prototyping
  - End-to-end process of bringing ideas to life for validation
- Relationship
  - Visual Library accelerates prototyping
  - Prototyping validates the right solution
- For Business Leaders: Invest in library; insist on prototyping phase

Success Strategy 1 — Start Right

1. Start with Outcomes and Questions

- What decisions will this enable? What questions must it answer?
- Who is the primary user?
- Write 3 'jobs to be done' before drawing anything

2. Choose the Right Fidelity

- Round 1: Low-fi to align on structure (paper, whiteboard)
- Round 2–3: Mid/high-fi clickable flows for key tasks
- Use sample data; clearly label 'sample data'

Success Strategy 2 — Test Smart

3. Prototype the Riskiest Assumptions First

- Examples: 'Users understand cohort analysis,' 'Executives will explore'
- Design small experiments: A/B layouts, task completion tests

4. Make Feedback Specific and Structured

- Script 5–7 realistic tasks
- Use rubric: clarity (A–F), findability, decision confidence, time-on-task

5. Validate Language and Mental Models

- Labels, filters, metrics must match user vocabulary
- Co-create metric dictionary in prototype (hover help, glossary)

Success Strategy 3 — Build for Reality

Instrument	Instrument Your Prototype (if possible) • Track clicks/paths to see drop-offs and confusion points
Design	Design for Performance Perceptions • Simulate loading states, empty states • Prototype default states to prevent 'blank dashboards'
Align	Align on Scope and Acceptance Criteria • Convert validated prototype into user stories • Freeze scope with 'prototype sign-off' before engineering

Success Strategy 4 — Scale and Communicate

9. Reuse with a Visual Library

- Standardize chart types, spacing, typography, interaction patterns
- Establish 'golden patterns' for common flows

10. Communicate the Iteration Plan

- Set expectations: 2–3 prototype rounds; time-boxed sessions
- Have a clear decision-maker to avoid design-by-committee

Facilitation Tips:

- Keep reviews to 30–45 mins; start with user/job; end with decisions

Discussion Questions

- For Each Scenario:
  - What assumption is riskiest?
  - What's the minimal prototype experiment to test it?
  - How would you structure user feedback sessions?

Key Takeaways

- 'Don't Skip the Prototyping Phase'
  - Prototyping occurs between understanding needs and development
  - More than one concept can be proposed
  - No harm if prototype is rejected or changed—that's the point!
  - Corrections happen before development begins
- Bottom Line:
  - Reduces expectation–reality gap; increases probability of success
  - Achieves this without spending resources on full development

Prototype Checklist

- Before You Build:
  - ☐ Problem framing: user, decisions, top 3 questions
  - ☐ Fidelity choice aligned to phase
  - ☐ Task script for testing (5–7 tasks)
  - ☐ Metric glossary and labeling review
  - ☐ Empty/loading/error states included
  - ☐ Instrumentation or observation plan
  - ☐ Acceptance criteria drafted and signed off
  - ☐ Reuse components from a visual library

Questions & Discussion

- Reflect:
  - Do you currently develop prototypes in your projects?
  - Or do you go straight to final product development?
  - What's one prototype experiment you could run this week?