Table of Contents

- 1. In-class micro-exercise: From Prompt → Spec → Web App (15–20 min)
  - 1.1. Setup (1 min)
  - 1.2. Part A: Identify the elements (5 min)
    - 1.2.1. Request A
    - 1.2.2. Request B
    - 1.2.3. Request C
    - 1.2.4. Request D
  - 1.3. Part B: Turn one request into a 5-line spec (5 min)
  - 1.4. Part C: Convert the spec into a web app plan (5 min)
  - 1.5. Optional (if time): Start implementing (5–10 min)
  - 1.6. Quick share-out (2 min)
- 2. Instructor wrap (30 seconds)

1. In-class micro-exercise: From Prompt → Spec → Web App (15–20 min)

1.1. Setup (1 min)

You will work in pairs. Your job is to:

1. identify the elements (Goal/Input/Output/Layout/Features) in short “messy” user requests,
2. rewrite them as a clean spec using the exact 5-line template,
3. turn ONE spec into a tiny web app plan (and optionally start coding if time).

1.2. Part A: Identify the elements (5 min)

Below are 4 short user requests. For each request:

- underline or label the parts that correspond to: Goal, Input, Output, Layout, Features
- if an element is missing, write **MISSING** next to it.

1.2.1. Request A

"I want a page that helps me pick a movie for tonight. I tell it my mood and how much time I have. It should suggest 3 movies. Put the suggestions in big cards and let me click one to see a short pitch."

1.2.2. Request B

"Make something to help me track my workouts. I enter exercise name, sets, reps, and weight. It should show a weekly summary and highlight personal records. Keep the form on the left and the charts on the right."

1.2.3. Request C

"I need a tiny tool to write better subject lines for emails. I paste my draft subject line and tell you the tone (formal/friendly). Return 5 improved options. Add a button to copy each one."

1.2.4. Request D

"Build a simple quiz page for my class. Students choose a topic and difficulty. The page generates 5 multiple-choice questions and shows the score at the end. Make it phone-friendly and add a ‘Try again’ button."

1.3. Part B: Turn one request into a 5-line spec (5 min)

Pick ONE of A–D and rewrite it using this exact template:

1. **Goal**:
2. **Input**:
3. **Output**:
4. **Layout**:
5. **Features**:

Rules:

- Each line must be one sentence.
- Inputs must be listed as fields (what the user types/chooses).
- Output must be testable (“I can tell if it worked”).
- Layout describes where things appear (left/right/top/bottom).
- Features are behaviors beyond the core input→output (history, copy, remove, save, etc.).

1.4. Part C: Convert the spec into a web app plan (5 min)

Using your rewritten spec, produce a short plan with 4 bullets:

- **UI components**: list the visible parts (form fields, buttons, panels, card area)
- **State**: list what variables the page must remember (e.g., messages[], currentInput)
- **Functions**: list 3–5 functions (e.g., generateMessage(), addToStack(), copyCard())
- **Events**: list what triggers what (button clicks, Enter key, remove card)

Deliverable: write this plan in your notes as if you were about to code it.

1.5. Optional (if time): Start implementing (5–10 min)

Create a single-file HTML app with:

- the inputs on the left,
- the output area on the right,
- one working button that produces an output.

You do NOT need perfect styling—just make the behavior work.

1.6. Quick share-out (2 min)

Each pair shares:

- one ambiguity you found (what you had to decide),
- one feature you added that wasn’t explicit (and why),
- one test you would use to verify the app works.

2. Instructor wrap (30 seconds)

Notice the pattern:

- A good prompt contains a spec.
- A good spec becomes UI + state + functions + events.
- Most “agentic” work is really: turning vague language into testable structure.

Author: Marcus Birkenkrahe

Created: 2026-01-31 Sat 16:25