Module 2: Architectural Planning and System Design, Lesson 2: Generating Product Requirement Documents

Hands-On: R-I-C-E in Action

Overview

This demo proves three claims from the lecture in real time:

- The R-I-C-E framework produces structured, complete PRDs — not vague bullet lists
- AI scales user story generation across multiple personas in a single prompt
- Gherkin criteria, non-functional requirements, and edge cases come from prompt structure, not memory

You will generate a full Product Requirements Document for TaskFlow — our AI-powered task prioritization feature — using exactly three prompts in the same chat session. The student lab extends this to a feature of the student's own choosing.

Before You Start

- Open your AI tool of choice: Claude.ai, ChatGPT, or Gemini
- Have a plain-text editor open (VS Code, Obsidian, or Notepad) to capture output
- Keep the R-I-C-E checklist visible on screen or a second monitor
- Do NOT start a new chat between prompts — context continuity is the point

---

> **Remember**
> Every PRD prompt follows R-I-C-E. We will prove it by building TaskFlow's PRD from scratch — skeleton first, acceptance criteria second, edge cases third — all in one session. Watch the context compound.

During Hands-On Keep Visible if Possible

```
1. ROLE         — tell AI who it is (senior PM, domain-specific)
2. INSTRUCTION  — what sections, how many items, what format
3. CONTEXT      — product, users, tech stack, compliance
4. EXAMPLES     — one user story, one acceptance criterion
```

Prompt 1 — PRD Skeleton with Personas and User Stories

What to Learn

Here's R-I-C-E in action. Tell the AI exactly who it is, what sections to produce, what reality to work in, and what format to output — and most importantly, tell it what NOT to generate yet. No acceptance criteria. No edge cases. Just the skeleton.

Paste this into the AI chat:

```
ROLE: You are a senior product manager and software architect with
10+ years of experience shipping B2B SaaS products. You write PRDs
that engineering teams can act on without requiring clarification.

INSTRUCTION: Generate a PRD for the feature described below. Include:
  1. Executive summary (2-3 sentences)
  2. Three distinct user personas — name, role, goal, pain point,
     and technical proficiency
  3. 15 user stories in "As a [persona], I want [action], so that
     [outcome]" format — distributed across all three personas
  4. Functional requirements grouped by feature area
  5. Non-functional requirements (performance, security, compliance,
     accessibility)
  6. Success metrics (3-5, each quantitative)

CONTEXT:
- Product: TaskFlow — a project management SaaS with 10,000 MAU
- Feature: AI-powered task prioritization that reorders a user's
  task list based on deadlines, dependencies, and team workload
- Stack: React frontend, Node.js backend, PostgreSQL
- Existing auth: REST API with JWT
- Compliance: GDPR with EU data residency; Article 22 applies
  (automated decisions require a human-override path)
- User types: project managers, individual contributors, team leads

EXAMPLES:
- User-story format:
  "As a project manager overseeing 5+ teams, I want to see why the
   AI ranked a task as high priority, so that I can trust the
   system and explain rankings to my team."
- Non-functional-requirement format:
  "Ranking API must respond within 500ms at p95 under normal load
   (up to 1,000 concurrent users)."

BEHAVIOR: Do NOT generate acceptance criteria yet. Do NOT generate
an out-of-scope section yet. Those come in follow-up prompts.
```

What you should get

> A PRD skeleton with three named personas, 15 user stories distributed across them, functional requirements grouped by area (ranking logic, manual override, audit log, UI rationale), and non-functional requirements with specific numbers. No acceptance criteria, no out-of-scope — those are the next two prompts.

You should ask:
"Why defer acceptance criteria to a second prompt?"

Answer: Context compounding. The AI now has all 15 stories in its working memory. The next prompt produces criteria that reference real personas and real story IDs — not a generic template. This is the session-continuity payoff.

Prompt 2 — Gherkin Acceptance Criteria for Every Story

> **Remember**
> Same chat window. The 15 stories are already in context — I don't repeat them. I just reference them and ask for Gherkin. Watch the criteria line up with specific personas and specific story numbers because the AI still remembers them.

Continue in the same chat session — do NOT start a new one:

```
For each of the 15 user stories generated above, produce Gherkin-
style acceptance criteria using GIVEN / WHEN / THEN / AND.

REQUIREMENTS:
  - Minimum 4 acceptance criteria per story (target 5-6)
  - Include at least one negative or failure case per story
    (e.g. what happens if the ranking service is down?)
  - Every criterion must be testable — use specific numbers,
    states, or UI elements. No vague terms like "quickly" or
    "appropriately."
  - Reference the persona name from the story in the GIVEN clause
    when relevant

FORMAT:
  Story [number]: [story text]
    GIVEN ...
    WHEN  ...
    THEN  ...
    AND   ...

BEHAVIOR: Output acceptance criteria only. Do not rewrite the
stories. Do not add commentary. This output will be copied directly
into a BDD test file in Module 3.
```

What you should get

15 stories, each with 4–6 Gherkin acceptance criteria. Persona names appear in GIVEN clauses. Numbers are specific (500ms, 3 seconds, 10MB). At least one failure path per story — because we asked for it explicitly.

Tip: These Gherkin criteria become the scaffolding for your BDD test suite in Module 3. One prompt session, two artifacts — the PRD and the test backlog.

Prompt 3 — Edge Cases and Out-of-Scope

What to learn

Still the same session. This is the prompt that earns its keep. Edge cases and out-of-scope are what most teams skip — and what causes the most production pain six months later. You're going to ask for them explicitly and by category.

Third prompt — still the same session:

```
Using the full PRD and acceptance criteria generated above, surface
missing requirements across six edge-case categories, then define
what is explicitly OUT of scope for v1.

EDGE-CASE CATEGORIES (2-3 requirements per category):
  1. Scale and load       — what happens at 10x normal traffic?
  2. Security boundaries  — cross-tenant data access, prompt
                            injection, auth-token misuse
  3. Failure modes        — ML model down, third-party timeout,
                            partial results
  4. Data integrity       — duplicate submissions, concurrent
                            edits, orphaned records
  5. Accessibility        — WCAG 2.1 AA minimum; screen-reader
                            support for AI-ranked lists
  6. Internationalization — time zones for deadlines, translated
                            rationale strings, RTL layouts

OUT-OF-SCOPE SECTION (minimum 5 items):
  Items explicitly deferred from v1. For each: one-sentence
  rationale and a target version (v1.1, v2, or "backlog").
  Examples to consider: native mobile app, team-wide AI settings,
  predictive deadline suggestions.

BEHAVIOR: Format as two clearly labeled sections. The out-of-scope
section must be phrased as "This feature does NOT include..." —
not as "future work." Phrasing matters; it sets expectations.
```

> **What you should get**
> 12–18 additional requirements across the six categories, plus 5–7 explicit out-of-scope items with rationale. Many of these are things nobody raised in the kickoff meeting — that is exactly the point. The AI surfaces them because the prompt forced the categories.

Save the output as v0.1 and commit it:

```
cp chat-output.md taskflow-prd-v0.1.md
git init && git add . && git commit -m "TaskFlow PRD v0.1 — AI draft"
```

At least one requirement will feel wrong or out of scope. This is intentional:

- The AI produced 30+ requirements in 90 seconds. You verify and own the prioritization.
- If a requirement doesn't fit, delete it. The cost of cutting from a draft is near zero.
- A PRD with too many requirements is a better problem than a PRD with too few.

Wrap-Up — R-I-C-E, Context Compounding, and the PRD as Code

> **Remember**
> Three prompts. One session. We produced a full PRD — three personas, 15 user stories, Gherkin criteria, non-functional requirements, edge cases, and an out-of-scope section — in roughly fifteen minutes. Each prompt built on the context of the last. That is R-I-C-E plus session continuity.

Key takeaways:

- R-I-C-E first, always — Role, Instruction, Context, Examples
- Defer acceptance criteria and edge cases to follow-up prompts — let context compound
- Gherkin criteria aren't just documentation; they seed your BDD test suite
- Edge cases and out-of-scope do not appear unless you ask — always ask, by category
- Version the PRD in Git as v0.1 — the AI wrote the draft; human judgment writes v1.0