Part 6: Case Study Execution Guide

Why You Must Open the IDE Now

Up to now, you've been doing:

- Problem framing
- Framework design
- Experimental design

Section 6 is **evidence**, not theory.

You cannot credibly write:

- 6.2 Application of the Workflow Framework
- 6.3 Results and Comparative Analysis
- 6.4 Observations and Patterns

without **actually running the task**, because:

- You need real decision-gate triggers
- You need real hallucination examples
- You need real rubric scores
- You need real iteration counts

Anything else would be speculative — and reviewers can tell.

What "Open the IDE" Actually Means (Step-by-Step)

This is not "go build an app from scratch."

It is **controlled execution**, aligned to your framework.

Step 1: Set Up the Baseline Environment

- IDE: WebStorm (as specified)
- Language: JavaScript (ES6+)
- Testing: Jest
- Linting: ESLint (+ security plugins if available)
- Accessibility (optional but good): axe-core

Create:

- Empty project scaffold
- Mock REST API (per Appendix C)

Do **not** pre-write logic.

Step 2: Capture Stage 1 and Stage 2 Artifacts

Before invoking any AI:

- Write the **Task Specification**
- Write the **Prompt Template**

These are *data*, not prose:

- Save them verbatim
- Timestamp if possible

These will be cited in Section 6.2 and Appendix A.

Step 3: Run the Task With Each Tool

For **each tool** (Codex, Gemini CLI, Claude Code):

1. Start at Stage 1 (same task spec)
2. Move to Stage 2 (same prompt)
3. Generate output
4. Apply triage (Stage 3)
5. Run verification (Stage 4)
6. Log:
   - Failures
   - Iterations
   - Modifications
7. Integrate only if pass criteria met

Important:

Do not "help" the AI outside the workflow.

All refinement must be triggered by a decision gate.

Step 4: Record Everything

You should record:

- Raw AI outputs
- Lint/test results
- Security findings
- Iteration counts
- Which stage caused rejection or regression
- Final rubric scores

This becomes:

- Section 6.2 narrative
- Section 6.3 tables
- Section 6.4 observations

What You Should NOT Do

- Do **not** optimize the code manually to "make it work"
- Do **not** fix things unless the framework allows it
- Do **not** hide failures — failures are *data*

What Happens After Execution

Once execution is complete:

- Section 6 will write itself
- Section 3 will be laser-focused
- Your Discussion section will be honest and grounded

Recommended Next Action

Yes — open your IDE.

Start with:

Stage 1: Task Specification

If you want, next I can:

- Help you finalize the exact task spec text
- Review your prompt template before you run it
- Help you design a logging sheet so you don't miss data

Tell me which you want before you begin.