Final project guide

Key takeaways

1. Build one consequential claim that a grader can rerun.
2. Independent evidence and controlled change matter more than breadth.
3. Design the repository, demo, and writeup as one evidence package.

Thirty course points follow proof, not volume

The final-project group is 2 + 23 + 5; every point is one percentage point of the course grade

2

Proposal

Four half-point commitments

23

Demo and code

Six evidence-scored axes

5

Writeup

A compact evidence index

A narrow, deeply verified tool can earn full credit; polish, size, technologies, lines, tokens, and prompt volume cannot substitute for evidence.

The deadline fixes the evidence boundary

Claims receive credit only when the grader can tie them to the pinned artifact or live defense

Primary grading evidence	Useful only as a pointer
Deadline repository commit and reachable history	A verbal account of what used to be in the repo
Behavior run live from that commit or a faithful local replay	Screenshots or narrated recordings with no rerunnable state
Committed tests, fixtures, specs, decisions, and reproducible reports	A terminal screenshot saying tests passed
Live answers that trace behavior through code and evidence	Chat transcripts, token counts, line counts, or commit counts

Generated code is graded exactly like human-typed code: credit follows verified engineering quality, not authorship volume.

Choose a claim a grader can rerun

One workflow, one independent challenge, and visible control of change

Runnable workflow

Start from a known state, run one consequential path, and expose an observable result.

Name the shortest safe command or interaction.

Independent challenge

Use an oracle, fixture, differential check, fault, held-back case, or user trace that can disagree with the code.

Choose a plausible wrong build the check rejects.

Controlled change

Keep two episodes that connect a bounded agent task to a diff, observation, and next human decision.

Preserve one correction, rejection, or scope cut.

One consequential workflow with strong evidence can outscore five shallow features.

The proposal makes four graded commitments

A few readable pages are enough; points live in the contract, not the polish

Points	Commitment	What full credit commits you to
0.50	Artifact and workflow	One runnable artifact, its user, the core workflow, and observable result
0.50	Validation and demo	Acceptance checks that reject a plausible wrong build, plus a short proof path
0.50	Human and agent ownership	Consequential human decisions, bounded agent tasks, and independent acceptance
0.50	Risk, fallback, and reuse	Main failures, smaller viable scope, dependencies, data, and prior-work attribution

Scope may change; silent drift may not

Record why the claim changed and what acceptance contract now applies

Choose one bounded workflow and its acceptance contract.

Record the failure, scope cut, or replacement decision.

Demo what the repository now promises; label gaps honestly.

Missing stretch goals do not hurt. An unacknowledged gap in the final core claim does.

Test count is not evidence quality

Design checks that can disagree with the implementation path

Evidence layer	Question it must answer	Strong examples
Automated checks	Would a plausible defect make this check fail?	Acceptance tests, mutation-sensitive cases, invariants, end-to-end smoke
Independent challenge	Does the expected answer come from a different path?	Oracle, differential model, fault injection, held-back input, user trace
Claim-to-evidence trace	Can a grader map each major claim to a current command or case?	A committed claim matrix with paths, commands, inputs, and report dates

Agent-written tests are useful, but tests that repeat the implementation’s assumptions are not independent proof.

Work backward from the live defense

Define what will prove the claim before you build toward it

Core proof

What visible workflow will run from a known state, and what result will prove the final claim?

Write the shortest credible demo path first.

Challenge

What wrong implementation, edge case, or failure should your independent check expose?

Design the discriminator before implementation.

First slice

What is the smallest end-to-end artifact that moves from the known state to an observable result?

Make that slice rerunnable this week.

Derive the plan backward from the proof, then build forward through evidence you can rerun.

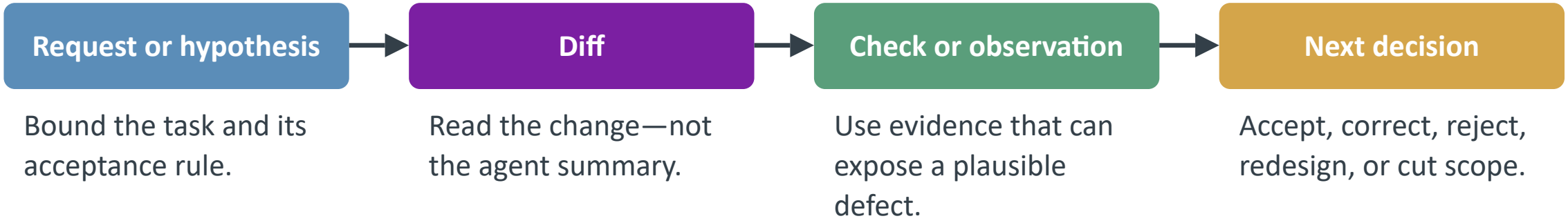
Four milestones build the evidence

Every milestone leaves a reviewable artifact in the repository

Milestone	Repository evidence	Exit check
E0 · Contract	Final claim, known state, setup, and first acceptance case	A fresh checkout runs one smoke command
E1 · Behavior	Core workflow plus a meaningful edge, failure, or invariant	The observable result matches the declared claim
E2 · Control	Two agentic episodes and one consequential correction or redesign	Each links request → diff → observation → next decision
E3 · Defense	Independent challenge, safe fixture/replay, demo script, limitations	The proof survives a clean reset and a bad network day

Agentic control is a loop, not a prompt

Keep two substantive episodes, including one correction, rejection, redesign, or scope cut



The gradeable unit is the engineering decision that survived evidence—not the amount of agent activity.

Record each loop where the diff and check live

A chat link may corroborate the request; the repository must preserve the decision and its proof

```
# docs/decisions/02-offline-replay.md

Claim or hypothesis:
Bounded agent request:
Changed files / commit:
Acceptance command:
Observed result or failure:
Next human decision:
Regression evidence:
Remaining limitation:
```

Preserve the failed check and the next decision; a polished success narrative erases the control loop.

The 23-point item has six independent axes

Score comes from axis evidence first; no overall impression or breadth bonus replaces it

Points	Axis	Decisive evidence
5	Behavior and correctness	Known-state core workflow, meaningful cases, agreement with final scope
5	Verification and evidence	Relevant checks, independent challenge, claim-to-evidence trace
3	Architecture and changeability	Owned seams/invariants, consequential change, justified dependencies
5	Agentic control and decisions	Human contract, two control loops, verification/correction/rejection
3	Reproducibility and resilience	Declared setup, rerunnable core path, useful failures and fallback
2	Defense, attribution, limits	Artifact-grounded explanation, accurate reuse/roles, honest boundaries

Missing essential evidence triggers hard caps

The grader scores all six axes, then applies the lowest cap whose exact trigger is present

Maximum	Cap	Trigger	Prevent it before the deadline
23	none	Deadline source and essential proof are available	Keep the repo, replay, and checks current
17	reproducibility	No staff-runnable/replay path and no durable executable proof of the core claim	Commit one safe setup/check path and its evidence
11	functional proof	Neither live behavior nor a faithful local fixture/ replay shows the core workflow	Build the local replay before demo polish
5	artifact unavailable	No inspectable deadline repository/source after URL verification and retry	Verify the submitted URL and commit from another account

A cap prevents strength in one evidence type from laundering the absence of an essential artifact; it is not an extra deduction.

The live demo is a defense, not a product tour

Every team receives these five prompts in substance; adapt the nouns, not the bar

#	Live prompt	Have this evidence open
1	Show the core workflow from a known state. What result proves the claim?	One command or short interaction plus visible result
2	Show a check that catches a plausible wrong implementation. Why is it independent?	Oracle, fixture, mutation, fault, or held-back case
3	Trace one consequential human/agent decision through the full loop.	Request/hypothesis, diff, observation, next decision
4	Show one failure, edge case, or external-dependency fallback.	Bounded diagnostic and faithful offline path
5	Explain one owned boundary or invariant and one known limitation.	Code path, tradeoff, attribution, honest unsupported case

Design the demo for a bad day

The core claim should survive service outages, missing credentials, and a clean machine

Fragile path	Resilient path
Live paid API call	Committed fixture or protocol replay, plus optional live mode
Private account or production data	Safe local seed data with the same graded contract
Manual hidden setup	One reset/setup command establishes a known state
Screenshots only	Rerunnable local command or app path from the deadline commit
Long feature tour	Core workflow, independent challenge, one control loop, one limitation

Make the repository answer before you speak

Keep claims, decisions, checks, and limitations close enough that a grader can follow the chain

```
README.md                    # versions; setup; demo-check
docs/CLAIMS.md               # claim -> command/case/report
docs/decisions/
  01-core-contract.md        # request -> diff -> check -> decision
  02-offline-replay.md
tests/fixtures/              # safe known states and challenges
reports/                     # inputs + commands + dated results
scripts/demo-check           # reset, run, expose result
LIMITATIONS.md               # real gaps and next experiment
```

Generated reports count only when their inputs and commands are identified and the grader can distinguish current evidence from stale output.

Pairs share one score and explain their work

Contribution accounting supports attribution; it does not split the rubric by commits, lines, hours, or speaking time

Shared artifact	Each partner must be able to show
One repository/demo score for the pair	Responsibilities they owned and work performed jointly
One common evidence package	One concrete decision or verification artifact they can explain
Comparable live questions	An artifact-grounded answer at the same bar
No improvised contribution penalty	Consistent attribution; contradictions trigger a separate review process

The writeup is a 5-point evidence index

Two to three pages is a useful target, not a length requirement; prose and visual polish do not earn points

Points	Section	What it should index
1	Identity and reproduction	Team, repository, graded commit, purpose, shortest safe setup/ demo path
2	Decision-to-evidence account	At least two consequential decisions, acceptance checks, next steps, and exact evidence locations
1	Architecture, dependencies, attribution	State and boundaries, external services/data, reused material, team roles
1	Limitations, change, learning	A real failure or scope change, the evidence that caused it, one remaining limitation

Do not write a sales pitch. Give the grader the shortest path from each decision to the pinned evidence.

Key Takeaways

1. Choose one final claim, a known starting state, and an independent check that can prove it wrong.
2. Preserve two request → diff → observation → decision loops, including one real correction or rejection.
3. Submit one coherent evidence package: pinned repo, resilient demo, honest attribution, and indexed writeup.