Capstone Assessment Rubric

Capstone Assessment Rubric (Canvas Version)

> **Instructor note:** This rubric is intentionally coarse and holistic. Each dimension is evaluated as a whole. There are no checklists. High scores require visible tradeoffs.

Capstone Assessment Rubric: How to Read It (https://canvas.ubc.ca/courses/185076/pages/capstone-assessment-rubric-how-to-read-it)

---

1. Commitment vs Flexibility

Assesses: Whether the student made concrete commitments early enough for reality to test them, and whether they revised those commitments when appropriate.

High performance shows:

- Early, specific assumptions or design decisions
- Evidence of revision, abandonment, or justified stability
- Clear explanation of *why* change did or did not occur

Lower performance shows:

- Prolonged vagueness that avoided falsification, or
- Rigid defense of early decisions despite contrary evidence

---

2. Depth vs Breadth

Assesses: Whether the student explored at least one part of the system deeply enough to encounter real constraints.

High performance shows:

- One or more subsystems examined in meaningful depth
- Conscious decisions about what was not explored
- Evidence that depth revealed tradeoffs or limitations

Depth must be visible—in your code, your reflection, or your video. If you went deep but didn't articulate what you found, we cannot assess it.

Lower performance shows:

- Superficial treatment of many components, or
- Depth without awareness of surrounding system context

---

3. Control vs Realism

Assesses: Whether the student balanced simplification with exposure to inconvenient or surprising system behavior.

High performance shows:

- Some modeling or simplification for understanding
- At least one encounter with messy, real behavior
- Awareness of what the system does *not* control

"Messy, real behavior" includes production-scale surprises (network issues, cost spikes, cloud quirks) and development-time friction (race conditions, API limitations, debugging dead ends). Either counts if you learned from it.

Lower performance shows:

- Overly clean or toy systems with no meaningful resistance, or
- Unanalyzed complexity that obscures understanding

---

4. Optimization vs Understanding

Assesses: Whether optimization was approached as a tool for learning rather than an end in itself.

High performance shows:

- Awareness of performance, cost, or reliability tradeoffs
- At least one concrete optimization attempt or cost model
- Reflection on what the optimization revealed

Lower performance shows:

- Premature optimization without understanding, or
- Avoidance of optimization and cloud-specific constraints entirely

---

5. Narrative Coherence vs Epistemic Honesty

Assesses: Whether the student explained their work clearly without smoothing away uncertainty, contradiction, or unresolved questions.

High performance shows:

- A coherent explanation grounded in evidence
- Explicit acknowledgment of mistakes, uncertainty, or limits
- Clear separation between belief, evidence, and speculation

Lower performance shows:

- Over-polished narratives with no visible cost, or
- Disorganized reporting that obscures learning

Global Requirement (Applies Across Dimensions)

At least once in the project, the student must make explicit a tradeoff of the form:

> "We deliberately sacrificed **X** to preserve **Y**, and this is what that cost us."

The tradeoff must be *substantive*—connected to the nature of your system, not merely to project logistics. "We sacrificed test coverage for features" is ordinary. "We sacrificed consistency to preserve latency under partition, which meant users occasionally saw stale data" is substantive.

Projects that do not demonstrate a real, system-specific cost cannot receive top marks.

> **Reminder:** Final assessment considers the *maximum* demonstrated learning across the code artifact, written reflection, and video showcase. The code artifact is frozen at submission; later artifacts must analyze the implemented system as it exists.