Branching Scenario Design Worksheet

Plan the graph, control the cost, and wire the tracking before anyone films. The companion article explains every line.

A. Design the graph - on paper, before the camera

- [ ] Write the objective as a judgment to practice, not a fact to recall
- [ ] Map setup -> confrontation -> resolution; one teaching point per decision
- [ ] Storyboard every node and edge in Twine (free); click-test it with the SME
- [ ] Make wrong choices plausible - the mistake a real, well-meaning person would make

B. Control the cost - branching explodes with depth

- [ ] Cap depth at 2-3 decisions; a 4th branch rarely earns its exponential cost
- [ ] Define the gold path: film the ideal route plus the most instructive wrong turns
- [ ] Re-merge wrong paths back to shared nodes so consequence clips are reused
- [ ] Count it: choices ^ depth = endings if fully expanded (3 ^ 4 = 81) - do not film that

C. Track the path - a choice is only data if the player reports it

- [ ] Emit an xAPI statement at EVERY decision (actor - chose - option), not just at the end
- [ ] Use the xAPI State Resource for resume (current node + path so far)
- [ ] If launched in an LMS via cmi5, set moveOn (Passed reached a passing ending)
- [ ] Plan path analytics: choice distribution, path frequency, drop-off

The build-vs-buy test before you commit

Branch only when the objective is a consequential judgment the learner must practice and getting it wrong in real life is costly or dangerous. For knowledge transfer, a single correct procedure, or a tight budget, a simpler in-player quiz or clickable hotspots deliver most of the engagement at a fraction of the cost. Start with free tools (Twine, H5P); move to a paid suite or a custom build only when you need deeper player control, branded experience, or analytics the off-the-shelf tools cannot give you. The honest default: branch rarely, branch shallow, reuse clips aggressively.