Run a Nudge-Comparison Series

Experimenter Set up a study once, then run it several times. Each run changes only the nudge (`experiment_run.nudge_id`), the short steering line that tells enrollees how to phrase their questions. Compare the runs' scorecards afterward.

Screens used Design-mode `design-mode.html` · Model-configurations `model-configurations.html` · Cohorts `cohorts.html` · Launch-run `launch-run.html` · Live-monitor `experimenter-monitor.html` · Watch-drawer `watch-drawer.html` · Results `results-analytics.html` · See-all `see-all-responses.html`

Data it touches the study `experiment` · the model set-up `model_config` · the cohort `cohort` / `cohort_member` · one run `experiment_run` · who's in a run `run_enrollment` · each scored exchange `chat_round` · the candidate answers `chat_round_candidate` · the run scorecard `run_result`

Key fields the nudge `experiment_run.nudge_id` · launch defaults `experiment.run_defaults` · run status `experiment_run.state` · immutable while referenced `RESTRICT` · headline score `run_result.composite_mean`

1 · Given / When / Then

The spine — reads as a how-to and doubles as an acceptance test. Human wording first; the exact table/field in `mono` .

GIVEN

A study (the `experiment` record) is set up in the **Design-mode Screen**. It holds the parts that stay constant across every run: the assistant's hidden instructions (`system_prompt`), the enrollee-visible framing (`scenario`), the opener (`opening_message`), and whether enrollees see the model competition (`blinded` , on by default). It also holds the grading set-up: the chosen scoring rubric (`score_rubric_version`), picked from a drop-down of the rubrics the middle tier bundles rather than typed by hand; the one trusted judge model (`score_judge_provider` · `score_judge_model`); and the weights that fill `score_weights` , seeded from the install default. Its launch defaults (`run_defaults`) supply a starting model set-up, a starting nudge, and whether to keep candidate answers.

- A reusable model set-up (`model_config` and its member models) exists, authored in the **Model-configurations Screen**.
- A cohort, the reusable group of enrollees a run targets (`cohort` with `cohort_member` rows), exists, built in the **Cohorts Screen**.

reads `experiment.run_defaults` , `model_config` , `cohort_member`

WHEN

For each nudge in the series, the experimenter opens the **Launch-run Screen**. The form pre-fills the model set-up, the nudge, and the candidate-keep rule from the study's launch defaults. The experimenter keeps the same group and model set-up, changes only the nudge, and clicks Launch.

THEN One run (`experiment_run`) is created, meaning one live execution of the study. It starts at `state = running` and points at this run's nudge, model set-up, and cohort through foreign keys (`nudge_id` , `model_config_id` , `cohort_id`). Those definitions, and the parent `experiment` , are held immutable while the run references them, so later edits cannot change what this run used — with no copy to store.

- As enrollees chat, each exchange is saved as a round (`chat_round` , with its `chat_round_candidate` answers). The study's one trusted judge fills the rubric's factor scores (`chat_round.factor_scores`) and the headline `chat_round.score_composite` .
- When the run reaches `state = done` , a one-row scorecard (`run_result`) is built. It records the composite mean and median, a five-number summary per factor, and pass-count, speed, and token statistics, all computed with the enrollee as the unit.
- Each nudge produces one run, so the series is a set of runs that differ only in the nudge. On the **Results Screen**, comparing them is one grouped query over their scorecards. Drill into the answers on the **See-all Screen**.

writes `experiment_run` , `run_enrollment` , `chat_round` , `chat_round_candidate` , `run_result`

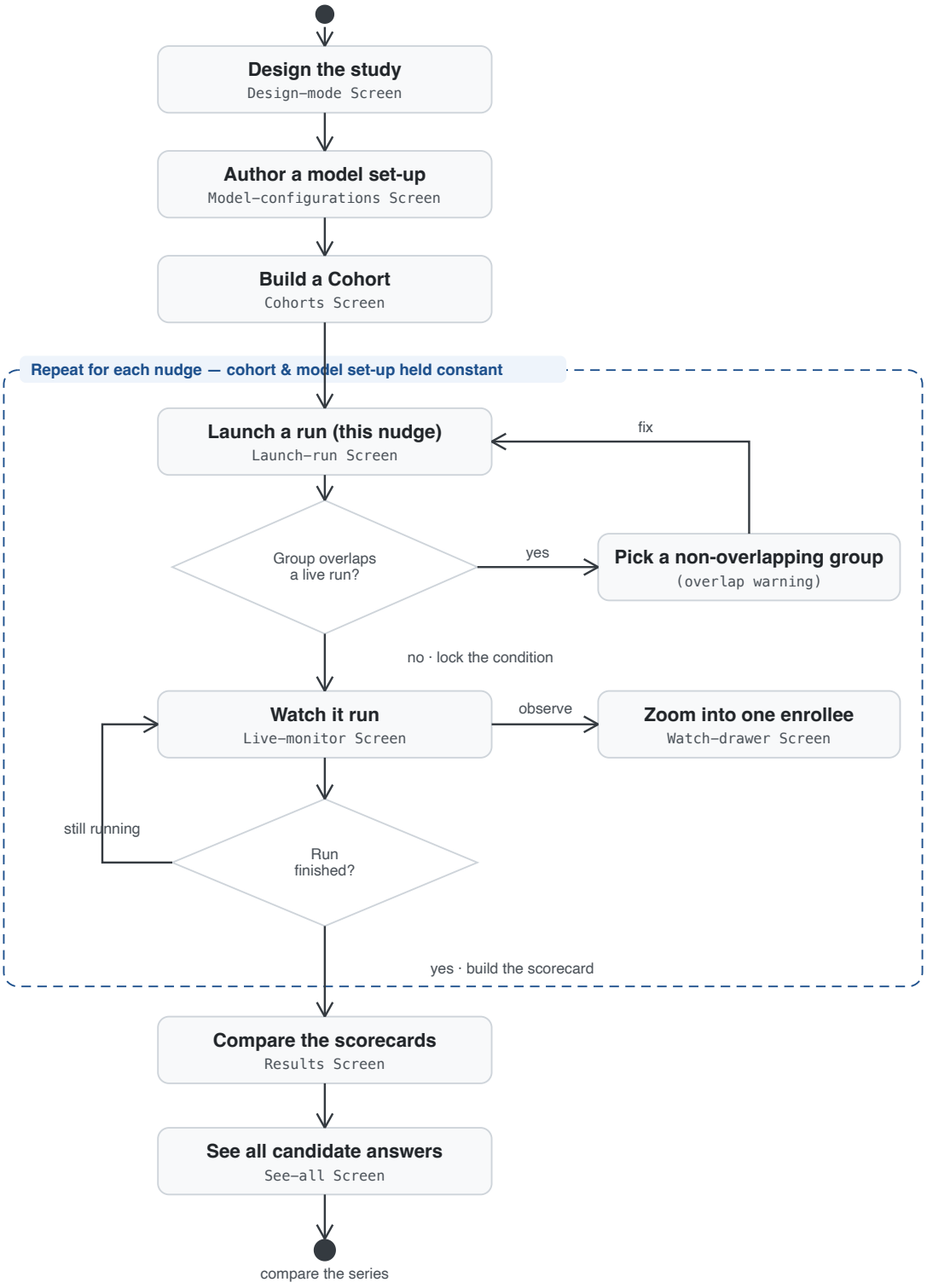
Variations and exceptions

IF…	THEN…
An enrollee in the chosen group is already in another run that is `running` or `paused`	The launch is blocked and the Launch-run Screen shows an overlap warning. End or wait for the other run, or pick a non-overlapping group — which is why a series on one group runs one run at a time.
Fewer than two runs have finished	There is nothing to compare yet.
They try to clear a run that is still live	Refused; a run is cleaned up or purged only once it is `done` or `aborted` .

Acceptance test — *Given* the study, model set-up, group and launch defaults exist, *When* the experimenter launches N runs changing only `experiment_run.nudge_id` (same group and model set-up) with no overlap, *Then* N runs reach `state = done` , N `run_result` scorecards exist, and the Results Screen shows a by-nudge comparison of their `composite_mean` .

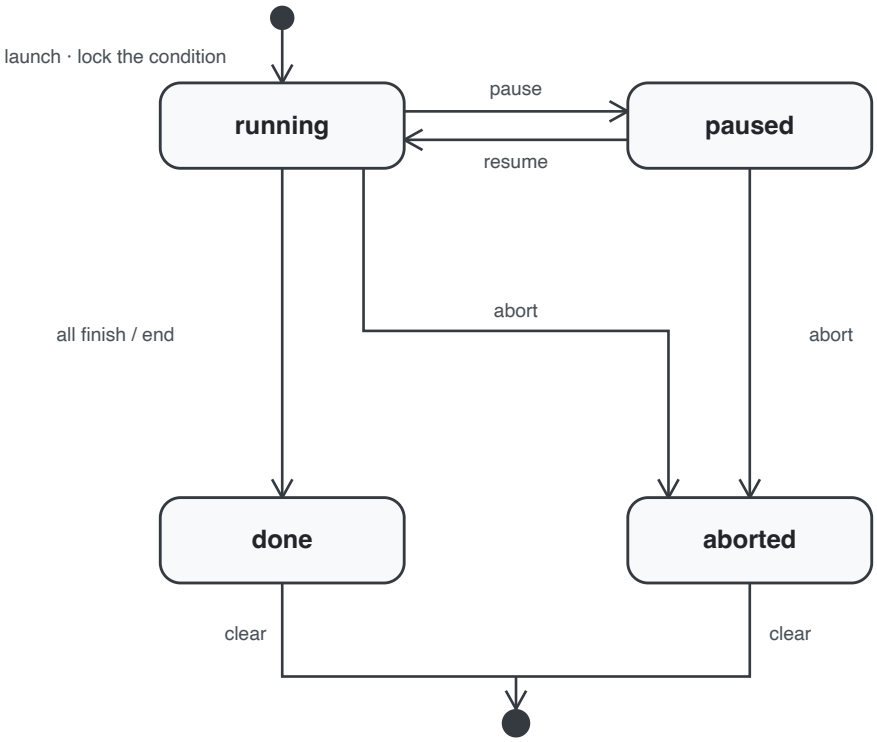
2 · The Journey, Screen by Screen

Boxes are the real screens; diamonds are decisions; the dashed frame is the loop repeated for each nudge.



3 · The Life of One Run

The status of a single run (`experiment_run.state`) — and what happens on each change.



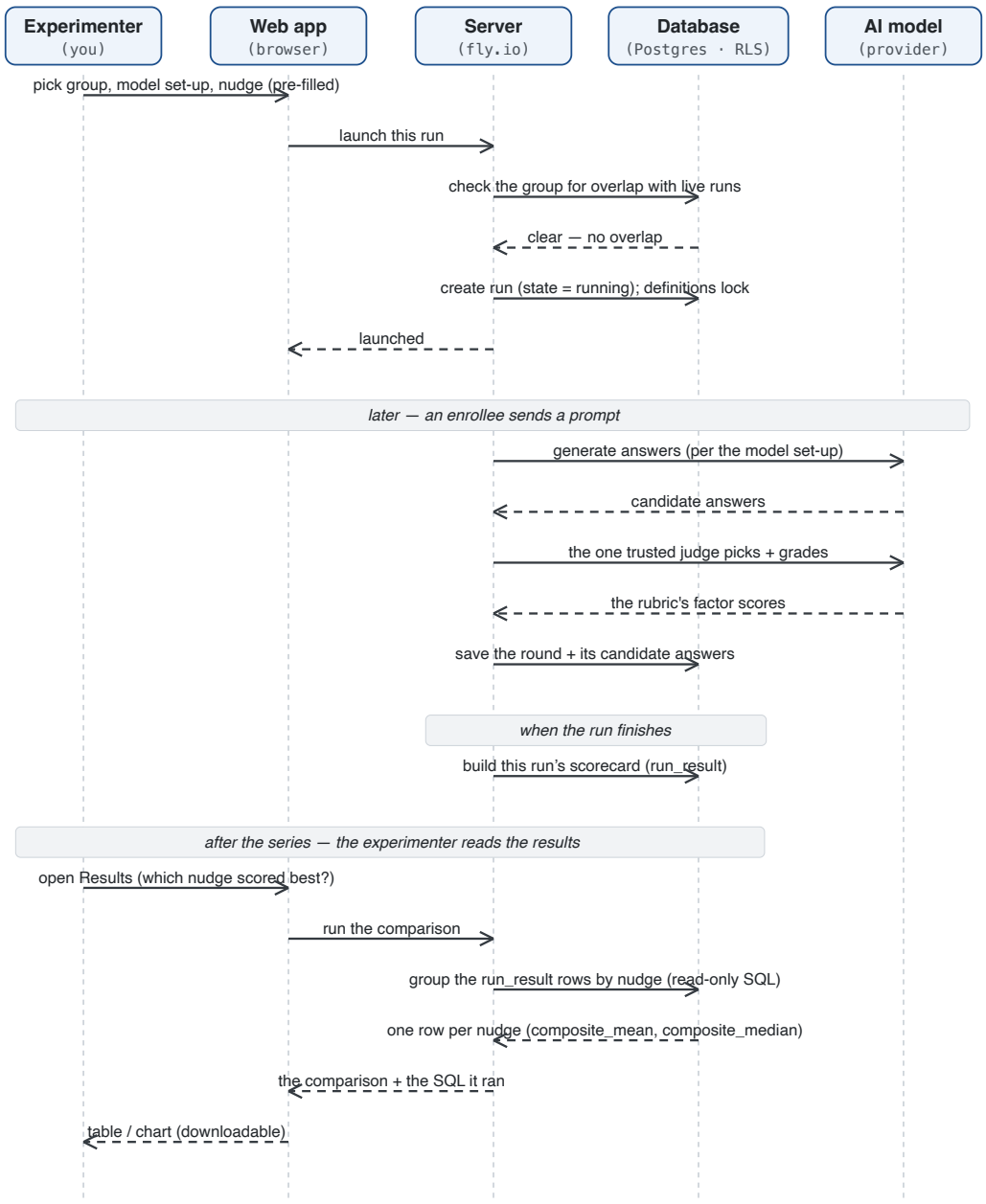
Clearing is blocked while a run is running or paused — only a done or aborted run can be cleared.

When a run finishes (done):
• build the scorecard (run_result)
• if “keep candidates” = at end of run, clear the per-answer detail

Seen and driven on the **Live-monitor Screen** (`experimenter-monitor.html`). Pause, resume, and abort are the buttons there. Reaching `done` builds the scorecard; only a `done` or `aborted` run can be cleared.

4 · Behind the Scenes

Who talks to whom — launching a run, scoring one exchange, and (at the end) computing the results comparison.



A dashed line is a reply. Two analytics moments appear here: the middle tier builds each run's `run_result` when the run ends, and the cross-run comparison is query-time SQL that groups those `run_result` rows by nudge. The blocked-overlap branch is in the journey above.

5 · In Plain English

An experimenter is trying to learn one thing: does the way you coach people to phrase their questions change how good the assistant's answers are? That coaching line is what ChatMaestro calls the nudge. Examples are “keep it short and direct” and “ask for step-by-step reasoning.”

To test the nudge fairly, everything else has to stay the same. So the experimenter sets the study up once: the assistant's hidden instructions, what enrollees see, which AI model set-up answers, and how answers are graded. They also build one group of enrollees to run it on. All of that is fixed.

Then they run the study several times in a row, on the same group, changing only the nudge each time. ChatMaestro pre-fills the launch form from the study's defaults, so the only thing they touch is the nudge. One rule keeps things honest: two runs cannot be live on the same people at once, because their answers would get tangled, so the runs go one after another.

While a run is live, a single judge AI scores every question-and-answer the same way. It grades four qualities: whether the answer is grounded in the source material, relevant to the question, internally coherent, and does what was asked, which is called instruction-following. Those four qualities roll into one headline score. When a run ends, ChatMaestro rolls the scores up into a one-line scorecard for that run.

Finally, the experimenter opens the Results Screen and puts the scorecards side by side. Because only the nudge changed between runs, any difference in the scores points at the nudge. To see why, they can open any exchange and read all the candidate answers the models produced, not just the one the enrollee saw.

Fits the schema cleanly. The series is deliberately not a stored object. It is simply the runs of one study, compared by grouping their `run_result` rows on the changing `nudge`. Every step above maps to existing tables, fields, and status values.