Compose an Experiment

Experimenter Set up the reusable study frame on one screen, meaning everything that stays constant across runs.

Screen Design-mode `design-mode.html`

Writes the study `experiment`

Key fields hidden instructions `system_prompt` · enrollee framing `scenario` · `opening_message` · enrollee visibility `blinded` · grading `score_judge_provider/model` · `score_rubric_version` · `score_weights` · prompt enhancer `enable_prompt_enhancer` · `prompt_enhancer_version` · chat memory `enable_chat_history` · launch defaults `run_defaults`

1 · Given / When / Then

Human wording first; the exact field in `mono`.

GIVEN The experimenter is signed in with the **experimenter** or **admin** role. The install has a default grading set-up in `.env` (seen read-only on the **Scoring-defaults Screen**). A model set-up already exists, authored in *Author a Model Configuration*; this screen only names it in the launch defaults, because the models are a per-run variable, not part of the study.

WHEN They open the **Design-mode Screen** and fill the fixed frame: a name and description; the assistant's hidden `system_prompt`; the enrollee-visible `scenario` and `opening_message`; whether the study is `blinded` (on by default, so enrollees see one answer rather than the model competition); the grading (the scoring rubric picked from a built-in list, the one trusted judge model, plus the `score_weights`, copied from the install default and editable); whether to enhance enrollee prompts before the models answer (`enable_prompt_enhancer`, off by default, its enhancer version picked from a list); whether the assistant remembers the conversation (`enable_chat_history`, on by default — turn it off to make every round self-contained, which removes conversation length as a variable); and the launch defaults(`run_defaults`: a starting model set-up, nudge, and whether to keep candidate answers). Then they Save.

THEN One `experiment` row is written. It holds only what stays constant across every run. The per-run variables, meaning the nudge, the model set-up, and the group, are chosen later at launch. Once the study has runs, its substantive fields are locked (only `notes` and the launch defaults stay editable), so every run in the series shares one fixed frame; to change the frame you clone the study.

writes `experiment`

Variations and exceptions

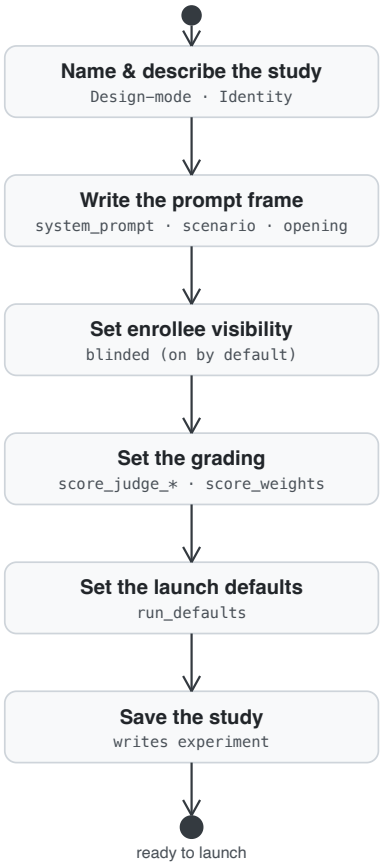
IF…	THEN…
The name is already in use	Refused on save: `experiment.name` is unique.
An enrollee-visibility control is left unset	It reads as its safe value: a missing See-all key keeps enrollees blinded.

IF…	THEN…
No context documents are attached	Allowed, with a warning that the groundedness factor will not be scored — there is nothing to ground answers against — so the omission is a deliberate choice, not a silent gap.

Acceptance test — *Given* a signed-in experimenter, *When* they fill the Design-mode frame and Save, *Then* exactly one `experiment` row exists with the prompt frame, the `blinded` flag, grading config and `run_defaults` set, and no per-run fields.

2 · The Journey, Screen by Screen

One screen, filled top to bottom; Save writes the study.



3 · In Plain English

Think of a study as a reusable recipe card. Before you can run anything, you write the parts that must stay the same every time, so that the only thing changing between runs is the one variable you are testing.

On this one screen you set what the assistant is secretly told to do, what the enrollee sees, whether the study is blinded, how answers are graded, and whether to polish enrollee prompts before the models answer. You do not pick the AI models or the coaching nudge here. Those are chosen fresh each time you launch, which is what lets you compare them.

You also set some launch defaults, so the launch form comes pre-filled and you are not re-typing the same choices for every run. Save, and the study is ready. From then on, changing it affects future runs only, never ones already in progress.

Fits the schema cleanly. Every field maps to a column on `experiment`; the per-run variables deliberately live on `experiment_run`, not here. `enable_chat_history` is a boolean with no depth beside it: the depth is a platform constant bounded by the smallest context window in whichever model set-up a run uses, so memory is identical across every arm of a comparison — a per-experiment depth would let it vary with the model set-up, which is itself a swept variable.