Run a Chat Session

Enrollee The enrollee just chats. Underneath, models compete, one trusted judge picks, and every exchange gets graded.

Screens My studies `enrollee-landing.html` | Chat `enrollee-view.html` | See-all `see-all-responses.html`

Writes each exchange `chat_round` | every candidate answer `chat_round_candidate` | presence `run_enrollment.last_activity_at`

Key fields the nudge `experiment_run.nudge_id` | how a turn-back is worded `experiment_run.socratic_enabled` | blinded? `experiment.blinded` | memory on? `experiment.enable_chat_history` | the answer `chat_round.response` | passes `num_iterations · stop_reason` | what was asked `prompt · raw_prompt` | turned back first? `clarification · retry_count` | did it refuse? `declined` | what it remembered `history_window` | which session, which turn `session_seq · round_seq` | rewound away? `rewound_at` | the scores `factor_scores · score_composite`

1 · Given / When / Then

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

GIVEN The enrollee is enrolled: a `run_enrollment` row links them to a run whose `state` is `running`. What they can see is fixed by `experiment.blinded` and `time_limit_min`.

WHEN From the **My-studies Screen** they enter the study and chat. The server first puts each message through the **sufficiency gate**: a message the study's corpus has nothing to say about, or one that leaves out something the experiment declared it needs, is turned back rather than answered — with a question for one missing piece when the run's `socratic_enabled` is on, or with a list of what is missing and a request to resend the whole thing when it is off — and the enrollee tries again. Once a message passes, it becomes one exchange. The server resolves the condition from the run's definitions (the experiment's prompts, the nudge, and the model set-up), replays as much of the current chat session as fits when `experiment.enable_chat_history` is on, rewrites the prompt if the study enables enhancement, asks every generator model, lets the study's one trusted judge pick the best candidate, keeps improving until a stop condition fires, and returns the winner. When memory is on the enrollee also has two controls over it: **Rewind to here** forgets everything after a chosen exchange, and **Clear chat & start new** forgets the whole conversation; at 80 percent of the memory budget the screen recommends the latter.

THEN One `chat_round` row is written per *answered* exchange: the prompt the models actually saw (`prompt`), the enrollee's own words when enhancement rewrote them (`raw_prompt`), the winning `response` , how hard it worked (`num_iterations` , `stop_reason` , `latency_ms` , `tokens`), and which model won or lost (`best_model` and `worst_model`). Any attempts the gate turned back first are kept on that same row in `clarification` and counted in `retry_count` — they are part of the record, but they are not rounds of their own. The row also says which session and turn it is (`session_seq` , `round_seq`) and, when memory is on, exactly what the models were shown (`history_window` : turns replayed, the budget, whether the oldest were dropped, whether the alert fired). A rewind stamps `rewound_at` on the rounds it forgets; a clear starts the next round at `session_seq + 1` ; neither deletes anything. Every candidate is kept as a `chat_round_candidate` . Then that same trusted judge grades the finished exchange on the rubric's factors and writes `factor_scores` and `score_composite` . This is the number all within-experiment comparison rests on.

reads `experiment` , `nudge` , `model_config` , `experiment_context_file` **writes** `chat_round` , `chat_round_candidate` , `run_enrollment.last_activity_at`

Variations and exceptions

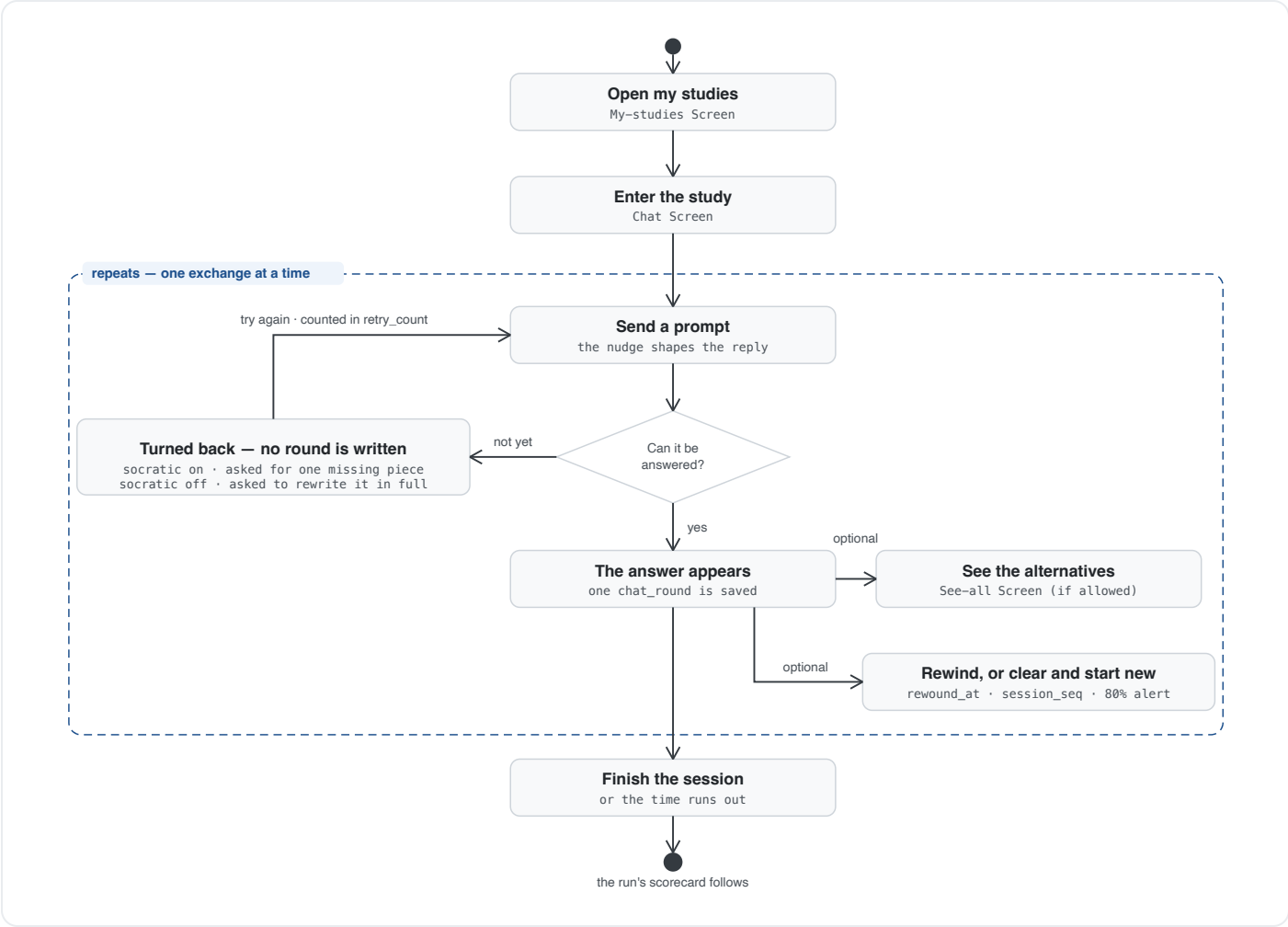
IF…	THEN…
The message has nothing in the study's context documents to ground it, or leaves out something `experiment.required_slots` declares	The trusted judge turns it back instead of answering. No `chat_round` is written, the models under test never see it, and there is no cap on attempts — somebody who gives up shows as an abandoned session rather than as a bad answer. Every attempt is counted in `chat_round.retry_count` on the round it eventually leads to.
…and the run's `socratic_enabled` is on	The judge asks for one missing piece at a time. Answers accumulate and reach the models as question-and-answer pairs, so the enrollee is never made to restate the whole request.
…and `socratic_enabled` is off	The judge names everything missing and asks for one rewritten, self-contained request; only that final request reaches the models. The two settings are compared on `run_result.retry_stats` .
The winning answer declines the request rather than attempting it	It is recorded in `chat_round.declined` as `appropriate` or `unwarranted` . An appropriate refusal takes relevance to N/A instead of scoring it badly; an unwarranted one stays scored, so declining is never a way to dodge a low score.
Enhancement is on but the rewrite fails validation	The enrollee's original prompt is used unchanged and `chat_round.enhancement` records the fallback, so a rejected rewrite is not mistaken for enhancement being off.
The replayed history reaches 80 percent of the shared window	The enrollee is told the session is filling and *recommended*, not required, to start a new one. Past the budget the oldest turns drop and the conversation continues; turns replayed, budget, truncation and the alert all land in `chat_round.history_window` .
The enrollee rewinds to an earlier exchange	`rewound_at` is stamped on every later round of the session, so they stop being replayed while the earlier context stays. Nothing is deleted: a rewound round keeps its scores and still counts.
The enrollee clears the chat	`session_seq` increments, so history assembles from empty — the blunter sibling of rewind.

IF…	THEN…
`enable_chat_history` is off for the experiment	Every round is self-contained: system prompt, retrieved context and this prompt alone. Conversation length stops being a variable, and the setting is frozen once the experiment has runs.
`experiment.blinded` is true (the default)	The See-all Screen is unreachable; the enrollee sees one answer and never learns that several models competed.
`time_limit_min` runs out	The session closes itself; the countdown was always shown, so it never surprises the enrollee.
The run is paused	Sending is disabled until an experimenter resumes it; nothing is lost.
A model errors	The failure is recorded in `chat_round.error` and the round still exists, so a failed exchange is data, not a gap.
`experiment_run.keep_candidates` is off	The losing answers are cleared after the run; the winner in `chat_round.response` is never affected.
The enrollee is withdrawn mid-run	`run_enrollment.withdrawn_at` stops the condition being served; the exchanges already produced stay.

Acceptance test — *Given* an enrolled enrollee in a running run, *When* they send N answerable prompts, *Then* N `chat_round` rows exist with a non-null `response`, `stop_reason` and `score_composite`, each with its `chat_round_candidate` rows; *and* when a prompt is turned back by the gate first — as a question or as a request to rewrite, per `socratic_enabled` — no extra row appears: the attempt is recorded in that round's `clarification` and `retry_count`; *and* when the enrollee rewinds to exchange k, every later round of the session carries `rewound_at` and the next round's `history_window.turns_included` counts only rounds 1…k; *and* when they clear the chat, the next round has `session_seq + 1` and `turns_included = 0`, while every earlier row is unchanged; *and* the See-all Screen is unreachable when blinding is on.

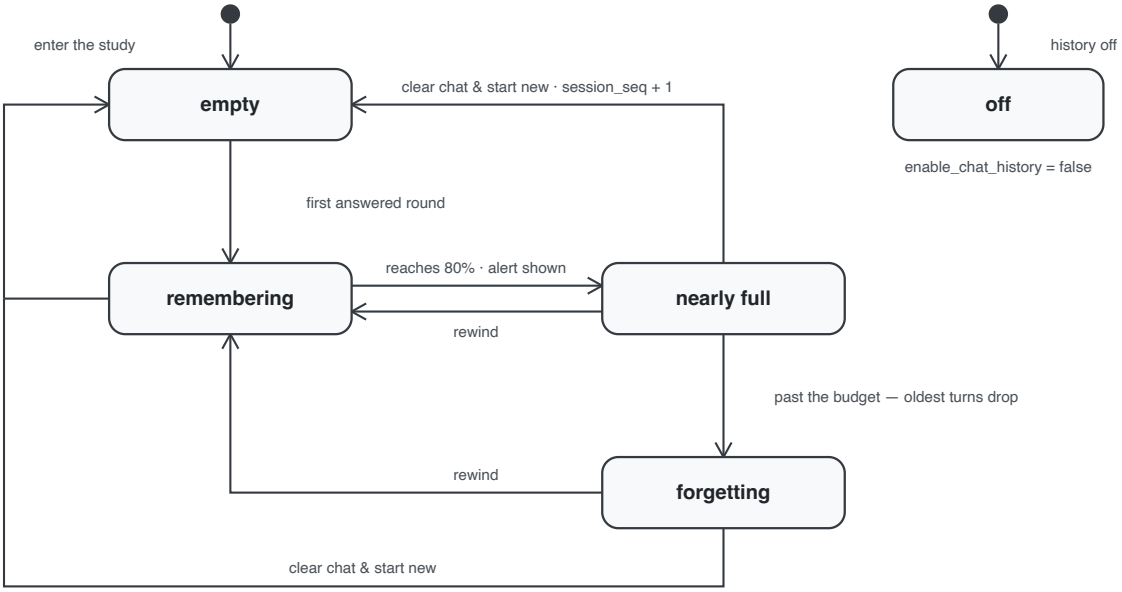
2 · The Journey, Screen by Screen

One screen, repeated exchanges, a gate before every answer, and two controls over what the assistant remembers.



3 · The Memory of a Session

What the assistant remembers, as `chat_round.history_window` records it — and the two controls the enrollee has over it.



Not a stored status: it is what history_window records on each round — the turns replayed, the budget (the smallest context_window in the run's configuration), whether the oldest turns were dropped, and whether the 80% alert fired. Rewind stamps rewound_at on every later round, so they stop being replayed but stay in the transcript and keep their scores. Clear chat & start new increments session_seq, so history assembles from empty. Both are always available to the enrollee. With enable_chat_history off none of this applies: every round stands alone.

Seen on the **Chat Screen** (`enrollee-view.html`): the filling-up notice, **Rewind to here** on every exchange, and **Clear chat & start new** in the toolbar. The budget is the smallest `model_catalog.context_window` in the run's configuration, so every model in the comparison is shown the same turns.

4 · Behind the Scenes

One prompt → gated → many candidates → one saved exchange → graded; then the memory controls, which stamp rather than delete.

One trusted judge appears four times here: it decides whether the request can be answered, rewrites it when the study asks for that, picks the winner during the loop, then grades the finished exchange. Every one of those is a decision that must be identical for every enrollee, which is why none of them is left to the models under test. Because it is the same judge on every run, the grade is comparable across the experiment's runs.

5 · In Plain English

To the enrollee this is just a chat window. They type, an answer comes back, and they carry on. Everything that makes it an experiment is invisible to them.

Not every message gets an answer. Some questions cannot be answered well: they ask about something the study's reference material does not cover, or they leave out a detail the study needs before anybody could respond usefully. Those come back from the assistant saying what is missing — as a question asking for one piece of it, or as a request to rewrite the whole thing, according to how the run is set up — and the enrollee simply tries again. The exchange that never got answered is not counted as one, because the round is the unit everything else is measured in: how many rounds, what each cost, how they scored. Letting an unanswered attempt in would quietly distort all three. So the attempts are kept alongside the round they led to, counted there as retries, and the numbers stay about answered questions.

Behind that single answer, several AI models may have each written a version, the study's one trusted judge scored them, the best was fed back for another pass, and weak performers were dropped. That repeats until the answer is good enough or the budget runs out. Only the winner is shown.

Some studies also rewrite the enrollee's message before the models see it, into a better-engineered version of the same request. When that happens both are kept: the rewrite the models answered and the enrollee's own words. The grading is done against the enrollee's words, because that is what they actually asked.

Whether the assistant remembers the conversation is a setting of the study, fixed before it starts. With memory off, each question stands alone — which is the cleaner way to compare single answers, because how long somebody has been chatting stops affecting what they get. With memory on, the recent conversation is replayed so that a follow-up like “shorter” means something.

Memory has a ceiling, set by the least capacious model in the line-up rather than the most: every model in a comparison must be shown exactly the same thing, or you end up measuring who was told more rather than who answered better. As a session nears that ceiling the enrollee is told, and advised — not forced — to start fresh. Going past it is allowed, and sometimes wanted; the oldest exchanges simply drop away and the chat carries on.

Two controls let the enrollee manage this. Rewind steps back to an earlier point and forgets everything after it; Clear-and-start-new forgets the whole conversation. Neither erases anything from the record: the exchanges stay in the transcript and still count, because backing up usually means something went wrong, and that is worth knowing.

Whether the enrollee ever learns any of this depends on the study. Most keep it hidden, because knowing that several AIs competed changes how people react. Some deliberately show the competition and let the enrollee browse the alternatives.

Afterward, that same trusted judge re-reads each exchange and grades it against the study's rubric — the scoring sheet that says which qualities count and how each is judged. The built-in default asks four things: whether the answer is grounded in the source material, whether it is relevant, whether it hangs together, and whether it did what was asked. A study measuring something else picks a rubric with its own qualities. One judge does all the scoring, which is the whole reason results can be compared to each other at all.

One case gets special handling: an answer that declines the request. Refusing and failing look identical to a grader otherwise, and a model that correctly refuses would score worse than one that plows ahead and answers badly. So the judge records that it refused, and whether the refusal was warranted. A warranted refusal is not marked down for irrelevance; an unwarranted one still is.

Fits the schema cleanly. The improvement loop's state machine is in *Author a model configuration*; the aggregation that follows is in *Analyze the results*.