Let's Coordinate

Two submodules under one umbrella: Let's Meet (group meeting scheduling with tentative calendar holds) and Let's Decide (structured survey with five question types and aggregated results).

Up to date as of v1.29.0

CONTENTS

- Troubleshooting

What Let's Coordinate is

Let's Coordinate is an umbrella module that solves the two operational pain points that crop up most often around a complex matter: scheduling a meeting with several people at once, and getting structured input from a small group on a decision the licensee needs to make. It ships as one sidebar entry on your dashboard that opens a hub picker with two tiles: Let's Meet (group meeting scheduling poll) and Let's Decide (structured survey). Each submodule is its own end-to-end workflow with its own create form, participant portal, dashboard list, and closing emails, but both share the same 12-people-total cap, the same bilingual participant page, the same per-participant token model, and the same reminder cadence.

The module is intentionally a small-group tool. Both submodules cap at 12 people total — you (the licensee) plus up to 11 invitees. The framing is 'a real conversation among people you actually know to make a real decision'. It is not a marketing-survey tool, it is not a booking page (it does not collect payment), and it is not a public-facing form. Every invitee gets a personal email with their own unique tokenized link; no public registration, no random sign-ups, no anonymous traffic from search engines.

When to use Let's Coordinate

Reach for Let's Meet when you need to schedule something with three or more people whose calendars do not auto-share with yours — a co-counsel + your client + the client's sponsor, an internal team meeting that includes your assistant + a translator + an outside specialist, a strategy call with the client and their family members in two timezones. Booking takes one client one slot; Let's Meet collects availability from several invitees, surfaces the times that work for everyone (or the most important subset of everyone), and materialises a real calendar event on the chosen slot. Reach for Let's Decide when you need explicit input from a small group on a question that has structured shape — picking a meeting time is not it (Let's Meet handles that), but picking between three retainer-fee options the client and their two co-signers all need to agree on, getting four family members to rate

four candidate dates of arrival, asking a translator team to confirm language coverage on a six-person interview — these are the shape of decision Let's Decide is built for.

Who can create and manage polls

The module is available to every tenant on every tier (Basic and Premium both); there is no Premium gate on the Let's Coordinate sidebar entry. Creation and management permissions follow the per-member module-access matrix the platform uses everywhere else — by default the Owner and Admin roles can create and manage polls in either submodule, staff members can view but not create. The Owner can override per-member from Settings ’ Team ’ that member ’ Module Access. Calendar block creation for Let's Meet honours the three-branch calendar resolver: per-service owner first, company-level calendar second, Owner-member fallback third. Whichever calendar is connected (Google or Microsoft 365) gets the holds and the final confirmed event.

The 12-people-total cap

Both submodules enforce the same headcount: 12 people total, which means you (the licensee) plus 11 invitees. The cap is enforced server-side at every invitee mutation route — adding a 12th invitee returns an error, and the form mirrors the cap with an inline counter so you see the limit as you build the invitee list. The 12-person framing is deliberate. The bigger you make the poll, the more diluted each response becomes; past about a dozen people, what you actually want is a booking page (live) or a public RSVP form (a different category of tool entirely). Let's Coordinate is for the closed-list group meeting where each invitee's response matters individually and you would call them on the phone if they did not respond.

Building the invitee list

Each invitee row carries four fields you fill in on the create form. Name (required — appears on the participant page and the email salutation). Email (required — gets the personal tokenized invitation; the form validates against the standard email regex). Essential flag (a checkbox per invitee — when ticked, that invitee is treated as essential

to the decision; the decision rules and the auto-confirm logic respect the flag, particularly in Let's Meet where an essential-not-yet-responded invitee can block auto-confirm). Internal note (optional, capped at a few hundred characters — never shown to the invitee themselves, only visible to you on the dashboard detail; useful for context like 'is the client's sponsor — must be at the meeting' or 'language: Spanish, may need translator coordination first').

The shared bilingual participant page

Both submodules use the same participant-portal architecture. When you click Send Invitations on a draft poll, the platform mints a 32-byte cryptographically secure token per invitee, hashes it, stores the hash, and emails each invitee a personalised link at /lets-meet/respond/<token> (or /lets-decide/respond/<token>). The plaintext token never persists anywhere — it lives in the URL only. The token is the capability: anyone with the URL can open the participant page and respond as that invitee. Each token is unique to one invitee on one poll; cross-poll reuse is impossible.

The participant page is bilingual end-to-end. The browser's language hint sets the initial locale (English or Quebec French); a `?lang=fr-CA` URL parameter overrides; an inline EN/FR toggle near the header lets the invitee flip without losing their in-progress response. A cookie remembers the choice across refreshes. Every visible string — the title, the question prompts, the option labels, the submit button, the success page — is translated through the platform's standard i18n surface, not via runtime machine translation, so the legal weight of the response is unambiguous in both languages.

Let's Meet — what it is

Let's Meet is the group-meeting scheduling submodule. You configure a date range, a meeting duration, optional invitee constraints, a decision rule, and a response deadline. The platform generates a grid of candidate start-time slots inside the date range (filtered against your weekly availability and your connected calendar's busy intervals),

each invitee marks the slots they can attend on their personal page, and once the decision rule is satisfied (or you manually pick a slot), the platform creates a real Google Calendar or Microsoft 365 event on that slot with every invitee CC'd. Confirmation and cancellation emails fan out automatically.

Let's Meet — meeting mode

Three meeting modes shape what kind of event the confirmed slot becomes. 'online_auto' (Google Meet) — the platform auto-creates a Google Meet conference URL on the confirmed calendar event; every invitee gets the join URL in their confirmation email; the asynchronous re-fetch handles Workspace domains where the conferenceData lands a moment after the event create. 'online_custom' — you provide a custom URL (Zoom, Microsoft Teams, Webex, Whereby, your own org's meeting platform); the platform uses that URL as-is and does not validate the destination beyond URL parseability. 'in_person' — you provide a physical address; the address prints on the confirmation email and on the calendar event's location field, and any GCal-side map embed picks it up automatically.

Let's Meet — date range and duration

The date range is two dates: a start and an end, anchored at noon UTC of each day to dodge timezone-rollover bugs. The platform walks each day in the range, looks up your weekly availability rules in the company timezone for that day-of-week, and surfaces every slot that fits. Practical date ranges are a few days to two weeks; ranges past about three weeks rarely produce useful coordination because invitees forget about an open poll, and the platform does not auto-extend if no one has responded yet.

Duration is the length of the meeting you are scheduling, picked from a fixed set of options: 15, 30, 45, 60, 90, or 120 minutes. The duration affects which start-time slots fit — a 90-minute meeting needs a 90-minute clean window in your availability, so the slot generator naturally drops half-hour gap slots that a 15-minute meeting would

have happily filled. Slot interval defaults to 15 minutes; this is the granularity the generator walks your availability with (e.g. start times 09:00, 09:15, 09:30, 09:45 in a 9-to-5 window with a 30-minute meeting duration).

Let's Meet — how slots are generated

Slot generation runs a four-filter cascade against every candidate start time in the date range. Filter 1 — must fit your weekly availability rules for that day-of-week. Filter 2 — must not overlap a confirmed booking on your tenant (same row-set used by the booking-page availability route). Filter 3 — must not overlap an existing Q&A session or event on your tenant calendar. Filter 4 — must not overlap your connected calendar's busy intervals (Google Calendar freebusy + Microsoft 365 calendar freebusy, both via the standard providers; freebusy queries run with a 15-minute buffer to keep you from back-to-back commitments). The output is a set of candidate slots the invitees see on their participant page.

Slots themselves are stored in the database as discrete rows (start_time, end_time, status) so they can carry state — held, confirmed, externally_booked, released. Once invitees start responding, the slot rows accumulate counts (how many invitees can attend each slot, how many have selected it) which feed both the decision rule and the optional tentative-hold logic.

Let's Meet — response deadline

Three deadline types shape when invitees must respond by. 'date_only' (the simplest) — pick a calendar date; the deadline collapses to end-of-day in your company timezone. 'date_time' — pick a date AND a time; the deadline is that exact moment. 'relative' — express it as 'N hours / days before the earliest candidate slot'; the platform resolves the absolute moment at create time. Whatever shape you pick, the absolute moment is stored as a timestamptz so reminder crons and the auto-confirm logic have one canonical value to compare against. The participant page renders the deadline in the invitee's locale and timezone (clear and unambiguous: 'Please respond by Thursday, June 26 at 6:00 PM Eastern Time').

Let's Meet — the five decision rules

The decision rule defines what 'enough invitees can attend' means for your specific poll. Five rules are available; the platform's recommended default is essential_plus_percentage with an 80% threshold, which works for most cases without further tuning.

- all_essential — a slot is a winner if EVERY invitee flagged essential can attend (optional invitees do not block). Use when the meeting genuinely cannot proceed without the named essential people.
- essential_plus_percentage — a slot is a winner if EVERY essential invitee can attend AND at least the configured percentage of optional invitees can also attend. The recommended default; balances 'the important people must be there' against 'most of the room shows up'.
- percentage_only — a slot is a winner if at least the configured percentage of ALL invitees can attend, no special handling for essential flags. Use when nobody is more important than anyone else.
- everyone — a slot is a winner only when 100% of invitees can attend. Strict, useful for very small groups (3-4 people) where the whole point is everyone shows up.
- tenant_manual — the platform never auto-confirms; you pick a slot manually after reviewing the response grid. Use when the decision involves judgement calls the rule engine cannot encode (e.g. you want to favour an essential invitee but only if a specific optional invitee is also free).

Let's Meet — essential-nonresponse policy

What happens when an essential invitee has not responded by the deadline? Three policies shape the answer. 'strict' (the recommended default) — non-responding essential invitees block auto-confirm; the poll transitions to pending_resolution and you decide manually. 'treat_as_unavailable' — the evaluator treats a non-responding essential invitee as if they had marked every slot unavailable; the decision rule may still find a winner among the

slots the responding invitees agreed on. 'manual_review' — the platform never auto-confirms when any essential invitee is missing, even if the decision rule would otherwise be satisfied; pending_resolution sits until you act.

Let's Meet — tentative holds on your calendar

Tentative holds are a feature that puts placeholder events on your calendar as invitees start picking slots, so a fast-moving day does not produce a conflict between the live booking flow and an in-progress poll. Three hold policies control when holds materialise: 'never' (no automatic holds; the platform only writes the calendar event after you confirm a winner), 'notify' (no automatic holds, but you get a heads-up email when a trigger condition fires so you can manually hold from the detail page), 'auto' (the platform creates the hold itself on the trigger condition).

Five hold-trigger conditions decide when the policy fires. 'first_select' — the first invitee to pick a slot triggers it. 'two_or_more_select' (the recommended default) — two or more invitees agreeing on a slot triggers it. 'any_essential_select' — any essential invitee picking a slot triggers it. 'rule_satisfied' — the decision rule has enough convergence to declare a winner. 'custom_threshold' — you set a specific N-of-invitees number. Holds auto-release if a different slot ultimately wins, if the poll is cancelled, or if the deadline expires without a winner. The hold's calendar event title is clearly marked as tentative so it does not confuse a tenant who glances at their week ahead.

Let's Meet — what invitees see on the respond page

Each invitee opens their unique URL and sees a participant page tailored to the poll. The header shows the poll title, your company branding, your description (if you set one), and the response deadline rendered in their local time. The body is a day-grouped grid of candidate slots: each day is a row, each slot within that day is a tile with a start time + duration. The invitee multi-selects the slots they can attend; the page shows a small 'Also picked by:' peer-name list under each slot so the invitee can see who else has already committed (privacy-preserving:

names only, no email addresses). A free-text comment field at the bottom lets the invitee add context — 'I can do Thursday only if it's before 4 PM' is the kind of caveat that shapes the licensee's manual review.

If the invitee genuinely cannot attend any slot in the range, the page offers a 'Decline' action — distinct from 'submit zero slots', because the platform respects intent: a declined invitee tells the licensee 'this poll cannot accommodate me; please coordinate separately', while a zero-slot submit tells the licensee 'I am responding but I disagree with all candidate times'. Both decisions are visible on the dashboard. Each invitee can edit their response any number of times before the deadline; the platform replaces the response in place and recomputes slot counts immediately.

Let's Meet — response tracking states

Each invitee row carries a per-invitee response status visible on the dashboard detail page. 'pending' — invitation sent, the invitee has not yet opened the link. 'opened' — the invitee opened the link at least once but has not submitted. 'responded' — submitted at least one slot selection. 'declined' — explicitly declined the poll. The status updates in near-real-time so you can see who is silent vs who is just slow to land on a commitment. The detail page also surfaces the per-slot heatmap — a count of how many invitees can attend each slot — so even before the decision rule fires you can eyeball the obvious convergence point.

Let's Meet — auto-confirm and manual confirm

When the decision rule is satisfied AND auto-confirm is enabled on the poll (a checkbox separate from the decision rule itself), the platform automatically picks the winning slot, creates the calendar event, fans out confirmation emails to every invitee, and transitions the poll to status confirmed. If auto-confirm is off (or the rule is tenant_manual), the poll transitions to pending_resolution: the dashboard detail page shows the candidate-slot table with a 'Qualifies' badge on every slot that satisfies the rule, and a per-slot Confirm button lets you pick.

Confirming manually fires the same calendar-create + email-fan-out as auto-confirm — the only difference is when, and who clicked the button.

Let's Meet — the confirmed calendar event

Once a winner is locked in, the platform calls the connected calendar provider (Google Calendar or Microsoft 365 via the provider abstraction). The event is created on your calendar with every invitee added as an attendee, the meeting URL or address embedded according to the meeting mode, the poll title as the event title, your description as the event description, the duration as the event length. Each invitee gets a confirmation email with the .ics calendar invite they can accept into their own calendar; for Google Workspace invitees and any invitee whose calendar provider supports RSVP, the event invite handles the rest.

Any tentative holds the platform created earlier are released as part of the confirm action — the calendar reverts to one real event instead of several placeholders. If the calendar create itself fails (an OAuth token expired mid-poll, a Workspace permission was revoked), the platform surfaces the failure on the dashboard with a 'Reconnect your calendar' callout, and the poll sits at confirmed-but-calendar-unwritten until you reconnect and retry.

Let's Meet — cancel, extend the deadline, reopen

The detail page carries lifecycle actions you can fire while the poll is still open or pending. Cancel — releases every tentative hold, deletes the calendar event if one was confirmed, fans out a cancellation email to every invitee, transitions the poll to cancelled (terminal). Extend deadline — bumps the response_deadline_at forward; the participant page refreshes; reminder emails do not re-fire for invitees who already responded. The platform does not support reopening a cancelled poll — start a new one if the conversation needs to restart. There is no per-invitee 'remove just this one person' action mid-poll; the workflow assumes the invitee list is right at create time and that adding or removing people mid-poll would invalidate the responses already received.

Let's Decide — what it is

Let's Decide is the structured-survey submodule. You build a poll with up to 30 questions across five question types, invite up to 11 people, send the invitations, the participants respond on their own per-person tokenized page, you watch the aggregates roll in on the result-explorer page, and when you are ready you close the poll. Closing fans out a closing email to every participant and to your tenant address summarising the outcome. The whole flow is purely informational — there is no calendar event, no payment, no follow-up commitment locked in — but the structured data the platform captures lets you make whatever downstream decision the survey informed.

Let's Decide — the five question types

Five question types cover the operational decisions a small group typically needs to make. Adding a new question on the builder picks the type up front, and the form then renders the type-specific configuration controls.

- single_choice — one of several options. The classic radio-button question. Useful for picking between mutually-exclusive paths ('which retainer tier should we go with: Bronze / Silver / Gold'). You author 2 to many choices; participants pick exactly one.
- multi_choice — any number of several options. Useful for collecting multiple preferences ('which of these four dates work for the interview' — invitees may pick all four if they are all open). The configuration includes optional minSelections and maxSelections to bound the answer (e.g. 'pick at least 1, at most 3'); the participant page enforces the bounds and the server re-validates on submit.
- rating_scale — a numeric rating on a fixed scale. Two scale options: 1-to-5 (typical 'how strongly do you agree' / star rating) and 1-to-10 (NPS-ish, finer-grained for opinion questions). Wider scales (1-to-7 Likert, custom ranges) are deferred to future versions in favour of keeping the question-renderer narrow.
- yes_no — a fixed binary. The platform auto-seeds the Yes and No choices when you add the question; you do not author them. Use for confirmation questions ('Are you available to attend in person, yes or no'). Maps internally onto the single_choice machinery so the result aggregates are consistent.

- open_text — a free-text answer. Two length variants: 'short' (up to 200 characters, renders as an <input>) and 'long' (up to 2000 characters, renders as a <textarea>). Use for the question that does not fit any structured shape ('any additional context we should know about'). The result explorer cannot aggregate open-text into a chart, but every individual response shows up in the per-question table.

Let's Decide — question prompts, descriptions, and required flag

Every question carries a prompt (the headline text the participant sees, capped at 500 characters), an optional description (up to 1000 characters, rendered as supporting context below the prompt), and a required flag that gates whether participants can submit the form leaving that question blank. Required defaults to true so the build-it-fast path is the safer default; toggle it off per-question if a question genuinely is optional context-gathering. Open-text questions in particular are often non-required so the participant can skip past 'any additional comments' without an asterisk-blocking submit.

Let's Decide — choice options for single / multi / yes-no

Single-choice and multi-choice questions need a list of options the participant picks from. You add as many as you need (the platform does not enforce a hard upper cap, but in practice past about a dozen options the question is better split into two questions). Each choice has a label (1 to 200 characters), displayed in the order you authored them — the platform does not randomise. Yes-no questions auto-seed the two Yes / No choices when the question is created; you do not edit them. The participant page renders single-choice as radio buttons, multi-choice as checkboxes, and yes-no as a pair of buttons styled distinctly so the answer reads as a deliberate Yes-or-No decision rather than a multi-select between options.

Let's Decide — anonymous responses and results-visibility settings

Two top-level poll settings shape who knows what. Anonymous responses (a poll-level toggle, defaults to ON) controls whether the result explorer and the closing email show individual respondent names. When on, every

response is recorded with the invitee_id but the result explorer aggregates only counts and percentages — you cannot tell which named invitee gave which answer. When off, the result explorer surfaces an additional 'Who answered what' table showing each named invitee's response per question. Tenants typically leave anonymous responses ON for opinion questions and turn it OFF for accountability questions where seeing who said what matters.

Three results-visibility settings control whether participants see aggregated results. 'private_to_tenant' (the default) — only you see the results, never the participants. 'shared_after_close' — participants see the aggregated results in their closing email once you close the poll. 'shared_live' — participants see live aggregated results on a 'View results' section of their participant page even while the poll is still open. The choice is a judgement call: private_to_tenant works for sensitive decisions where you do not want the room to anchor on early responders; shared_live works for genuinely collaborative decisions where the group benefits from seeing convergence as it happens.

Let's Decide — what invitees see on the respond page

The participant page is a single-screen survey form. The header shows the poll title, your company branding, your description, and the response deadline. The body is the questions in the order you authored them; each question renders with the type-appropriate control (radio buttons, checkboxes, rating-scale buttons, short text input, long textarea). Required questions are marked with an asterisk; if the poll has allowPartialResponses turned off (the safer default), the submit button is disabled until every required question is answered. A 'Save progress' affordance lets the invitee come back later without losing their work — the platform persists their partial response and they can finish on a subsequent visit.

On submit, the platform replaces any prior partial response with the final answer set, stamps a responded_at time-stamp, increments the response counter, and shows the invitee a confirmation page. If the poll has shared_live results-visibility, the confirmation page includes a 'View live results' button that opens the same aggregate view

you see on your dashboard. Invitees can re-open their participant link any number of times before the deadline; submitting again replaces the prior submission. Edits past the deadline are blocked; the page shows a friendly 'this poll has closed' message instead.

Let's Decide — the result explorer

On your dashboard at /dashboard/lets-coordinate/lets-decide/<id>, the result explorer shows aggregated answers across every responded invitee. The view is per-question: a heading with the question prompt, a small response-count indicator (e.g. '5 of 7 invitees responded'), and an aggregate render appropriate to the question type. The renders use HTML/CSS bars rather than a third-party charting library — no JavaScript dependency, no external assets, no print issues.

- single_choice + yes_no — a horizontal bar per choice option, sorted by descending count. Each bar shows the choice label, the count, and the percentage (rounded to whole numbers). The dominant choice visually leaps out.
- multi_choice — same horizontal bars per option, but the percentages are 'percentage of respondents who picked this option' rather than 'percentage of total selections', so a multi-pick option that 5 out of 7 invitees picked reads as 71% (not 'this option got X out of total Y selections').
- rating_scale — a histogram across the rating range (1-5 or 1-10), plus a numeric mean rounded to one decimal place (e.g. 'Average: 4.2'). Use the mean for a quick read; use the histogram to spot polarised responses (a 1-and-5-bimodal pattern means very different things than a tight 3-4-cluster).
- open_text — a list of every individual response (anonymous or named depending on the anonymous-responses toggle). No aggregation; the licensee reads each comment individually. The platform truncates very long responses in the list view with a 'show full' expander.

Let's Decide — closing the poll

Closing is the deliberate action that ends the response window and locks the aggregates. From the dashboard detail page, click 'Close poll' — the platform transitions the poll to closed (terminal), stamps ld_closed_at and ld_results_published_at, regenerates the aggregates one final time, and fans out closing emails. Every participant gets a per-participant closing email (RCIC App branded, sent from the tenant transport) thanking them for their response; if results-visibility is shared_after_close or shared_live, the email includes an aggregate-summary snippet so they see the outcome without needing to revisit the portal. You (the tenant) get a platform-branded tenant alert email summarising the response rate and the high-level aggregate so you have a record in your inbox.

Closed polls are not reopenable — once the aggregates lock and the closing email goes out, the participant page reads 'this poll has closed' for anyone who tries to revisit. If the licensee realises mid-close that one more respondent was supposed to weigh in, the right move is to start a fresh poll rather than re-opening the closed one; the integrity of the recorded outcome is preserved that way. Cancel is the analogue of close for the 'we are not going to use this poll' case — it transitions to cancelled without firing the result-share aggregate.

The reminder crons (both submodules)

Both submodules share a reminder cadence run by the `/api/cron/lc-reminders-24h` (daily) and `/api/cron/lc-reminders-30m` (every 5 minutes) cron routes. The 24-hour reminder fires once at roughly T-24h before the response deadline against every invitee whose status is still pending or opened (not yet responded, not declined). The 30-minute reminder fires once at roughly T-30m before the deadline against the same set. The platform rotates each invitee's tokenized link on every reminder send before firing the email — the email always carries a fresh URL, and any previously-sent URL becomes a dead link the moment the rotation happens. This is a deliberate security posture: a leaked older email cannot be exploited to respond on behalf of the invitee.

A third cron, `/api/cron/lc-evaluate` (every 15 minutes), is Let's-Meet-specific. It re-evaluates the decision rule on every open poll (handy when the deadline has passed and the strict essential-nonresponse policy needs to

transition the poll to pending_resolution), sweeps expired tentative holds, and auto-confirms any poll where the rule is satisfied and auto-confirm is enabled but the platform missed the moment of satisfaction at response-submit time. Let's Decide has no evaluator equivalent — its lifecycle is purely tenant-driven (no auto-close), so the evaluator skips Let's Decide rows by submodule discriminator.

Audit log and event history

Both submodules write to the same `lets_coordinate_audit_events` append-only table on every meaningful action — invitation sent, reminder sent, response received, response updated, response declined, hold created, hold released, poll opened, poll auto-confirmed, poll manually confirmed, poll cancelled, deadline extended, Let's Decide poll closed, results published. The table has an UPDATE-blocking trigger so audit history cannot be tampered with after the fact. The dashboard detail page surfaces the audit timeline as a chronological list at the bottom — useful when you need to reconstruct who did what in what order (a participant claims they never got the reminder; the audit log shows the reminder sent timestamp and the rotated token's hash prefix so you can confirm).

Interactions with other modules

Let's Coordinate interacts with Calendar (Google or Microsoft 365) and Bookings the most. Let's Meet's slot generation reads the same weekly availability rules that drive your public booking page, so when you change your availability in Settings the change immediately flows into open polls' candidate slots. The tentative-hold and confirmed-event creation routes through the same provider abstraction that Bookings and Events use; whichever calendar provider you have connected gets the writes. Let's Decide has no calendar interaction at all — it is a pure information-collection workflow with email as its only external surface.

Neither submodule auto-creates downstream records in other modules (no Transfer Room is provisioned on a confirmed Let's Meet, no Service Agreement is seeded from a Let's Decide outcome). The licensee decides what

to do with the meeting or the decision after the poll closes; the platform stays out of that next step on purpose. The data the platform captured is durable — the confirmed event remains on your calendar, the Let's Decide aggregates remain on the detail page indefinitely — so you can come back to either months later for a record.

Troubleshooting

- Invitee says they never got the invitation — check the invitee row's response status on the detail page. If status is still 'pending' (never opened), the email may be in spam; if status is 'opened', the email landed but the invitee has not committed yet. Use the per-invitee Resend action on the dashboard to fire a fresh email with a rotated token (the original link dies the moment Resend fires).
- Let's Meet — no slots appear for some days in the date range. The slot generator filtered them out at the four-step cascade. Common causes: your weekly availability has no entry for that day-of-week (the most frequent reason — weekends with no availability rule produce zero slots); your connected calendar shows you busy all day on those dates; an existing tenant booking or event already occupies the otherwise-fittable windows. Check Settings ’ Availability and your calendar's busy intervals; for short-term fixes, extend the date range or shorten the meeting duration.
- Let's Meet — auto-confirm did not fire even though the rule looks satisfied. Most likely the essential-nonresponse policy is strict and an essential invitee has not responded. Check the response grid on the detail page; non-responding essential invitees show a 'pending' badge. Either extend the deadline + nudge them, switch the essential-nonresponse policy to treat_as_unavailable mid-poll (the dashboard supports the edit), or manually confirm the winning slot.
- Let's Decide — submit blocked by 'choose at least N' on a multi-choice question. The question has minSelections set to N, and the participant has selected fewer than N options. They need to widen their selection or you

need to lower the floor on the question config. Editing the question config mid-poll is supported but invalidates any in-flight submissions matching the old bounds.

- Let's Decide — aggregates look wrong on the result explorer. Refresh the page; the aggregator is recomputed on every detail-page render. If the numbers persist looking wrong, check whether the anonymous-responses setting is what you expected (anonymous OFF surfaces individual responses; ON shows only counts) and whether some invitees declined vs simply did not respond (declined invitees are excluded from the denominator on some aggregates).
- Calendar event was not created on a confirmed Let's Meet — your connected calendar's OAuth token may have expired. Open Settings ’ Calendar; the dashboard surfaces a 'Reconnect' banner when the token is dead. Reconnect, then go back to the poll detail page; the platform offers a 'Retry calendar create' action that re-fires the create without re-confirming the slot.
- Token rotated mid-poll and now an old email link does not work — this is by design. Every reminder rotates the token; the most recent email is the only one with a working link. If a participant insists on using a specific older email, the only recovery is to fire a fresh Resend from the dashboard so they get a brand-new email with a current token.