

Build Your Own Scheduling Tools

Two working replacements you build yourself and host on a domain you already have — a booking page instead of Calendly, and a group time poll instead of Doodle. Both run on Google Calendar. Neither one costs anything per month.

The whole design rests on one idea — your calendar is the database. Open time slots are just calendar events, and the poll answers live in a spreadsheet. That single choice is what keeps this free and keeps it yours.

1 What You're Building

Two tools that share one backend. Build the booking calendar first — the poll reuses its Apps Script project, so the second build is far shorter than the first. You can also build either one by itself.

The booking page — instead of Calendly

Someone opens your page, sees your open times, picks one, and fills out a short form. The slot vanishes for everyone else, they get a confirmation email and a calendar invite, and you get a notification. You control availability by creating recurring calendar events titled `... - OPEN`. Nothing else to maintain.

The meeting poll — instead of Doodle

You propose a few possible times and get one share link. The group votes — works, works if needed, or no — and a grid shows who chose what. You also get a second, private organizer link. Click one time from it and the meeting is created with a video link, invitations going to everyone who voted.

Three moving parts, both tools. The first is a static web page you upload to any host. The second is a Google Apps Script Web App that runs as your calendar's Google account and does the calendar reading and writing. The third is Google Calendar, where the data lives. No database server, no login screen, no framework, no monthly fee.

2 Starting Prompt — The Booking Page

Hand this to Claude, or another capable AI coding assistant. Fill in the *bracketed parts* first. It is also in the kit as `prompt-1-booking-calendar.txt`, ready to copy.

Build me a free, self-hosted booking calendar to replace Calendly.

What it does. Visitors come to a web page, see a list of open dates and times, pick one, fill out a short form, and submit. On submit, the slot is instantly claimed, the booker gets a branded confirmation email and a calendar invite, and I get an admin notification.

My setup. I have a Google account at *[scheduling@yourdomain.com]*, and its calendar is the one bookings should read from and write to. I have web hosting at *[host name]* where I can upload static files, or I need you to recommend a free option. I want to host it at *[book.yourdomain.com]*.

The booking flows I need. *[Describe each one — for example, one public flow for prospective clients and one private flow, shared by link only, for existing clients. Say how long each meeting runs.]*

Architecture I want (keep it free, no subscriptions). The frontend is static HTML, CSS, and JS that I upload to my host. The backend is a Google Apps Script Web App that runs as my calendar's Google account and does the calendar read and write. Availability is controlled by pre-creating recurring calendar events titled exactly *[Meeting Name] – OPEN*. The script lists those as open slots, and on booking it rewrites the title to claim the slot and adds the booker as a guest.

Availability, the other way. Instead of one recurring event per slot, offer me the option of one recurring daily block titled *[Meeting Name] – OPEN* that the script slices into start times. Use a fixed step, and pick a step that goes evenly into the meeting length plus the buffer so bookings pack without odd gaps. In that design a booking becomes its own event and the block is left alone; the busy check hides the taken time. Explain the trade-off and let me choose.

Page shape. *[Pick one: a flat list of upcoming times, or a month calendar where I click a day, see its times, then fill in the form. If you are not sure, ask me.]*

Booking rules I want. *[Pick what applies — a minimum notice window such as 24 hours, a buffer around every booking such as 15 minutes, a cap on bookings per day, and how far ahead people can book. Put each one in a named constant at the top of the script so I can change it later.]*

Details the script must get right.

- Enable the Calendar Advanced Service with the identifier exactly *Calendar*, and set the project time zone in Project Settings. Every date in every email comes from that setting.
- Take a script lock on every claim, so two visitors cannot take the same slot.
- Enforce every booking rule twice: once when listing slots and again at the moment of booking. A tab left open for an hour can otherwise book past the daily cap.
- Free/busy must fail closed. If any blocking calendar cannot be read, show no times at all rather than risk a double booking.

- List events with `singleEvents: true` and loop on `nextPageToken`. Without both, recurring series never expand and a busy calendar gets cut off.
- Only timed events with a title block time. All-day and untitled events do not, unless I say otherwise. Include the script's own calendar in the free/busy list, or blocking time on that account does nothing.
- Claim a recurring instance in two steps: first break the single date out as its own exception, then add the guest in a second patch. One combined patch puts the guest on every occurrence.
- Pass `conferenceDataVersion: 1` whenever you create or remove a Meet link.
- On claim, append to the event description. Never overwrite it; I put join links there.

Testing. Add a `LIVE` constant, false by default, that suppresses all guest emails until I flip it to true for go-live. Add a `testSetup()` function I can run from the editor that checks calendar access, counts my OPEN blocks, and lists the next few bookable start times.

Build me the HTML pages, the shared CSS and JS, the complete Apps Script `.gs` file, and a DEPLOY.md with setup instructions. Show me how to deploy the Apps Script Web App and wire its `/exec` URL into the frontend. Use my colors and fonts — *[paste your brand specs, or write "ask me"]*. Ask me any clarifying questions before you start.

Why it works. It specifies the *architecture*, not just the goal. Naming the static-frontend-plus-Apps-Script-plus-calendar pattern is what makes the build free and durable. Leave it out and the assistant reaches for a paid service or a server you have to maintain.

3 Starting Prompt — The Meeting Poll

Also in the kit as `prompt-2-meeting-poll.txt`. If you already built the booking page, tell the assistant to merge these actions into that same Apps Script project so both tools share one `/exec` URL.

Build me a free, self-hosted group scheduling poll to replace Doodle.

What it does. I propose a handful of possible meeting times and get back one share link. Anyone with that link votes on each time — works, works if needed, or no — with their name and email. A results grid shows who chose what. I also get a private organizer link. When I pick the winning time from that link, the script creates a calendar event with a video link and sends the invite to every person who voted.

My setup. I have a Google account at `[scheduling@yourdomain.com]` whose calendar this uses and whose address sends the mail. I have static hosting at `[host name]`. I want the poll pages at `[book.yourdomain.com/poll/]`.

Architecture I want (keep it free, no subscriptions). The frontend is static HTML, CSS, and JS. The backend is a Google Apps Script Web App that runs as that Google account. A Google Sheet is the database — the script creates it on first use and remembers its id in Script Properties, with one tab for polls and one for votes. Every poll gets a random public id and a separate secret organizer token. The actions I need are `poll-create`, `poll` (read), `poll-vote`, and `poll-finalize`. On finalize, the script writes the calendar event, attaches a video link, and invites every voter. A time that already carries its own join link keeps it and never gets an auto-generated room. If the video service fails, fall back to a Google Meet link so the group is never left without a room.

Guardrails I want.

- Creating a poll requires a passphrase stored as a Script Property, so nobody but me can open polls on my domain. Voting needs only the poll link. Never link the create page publicly.
- Between 2 and 15 candidate times per poll, future times only.
- Maximum 100 voters, one vote per email address — a repeat email replaces that person's earlier answer rather than adding a second row.
- Voter email addresses are never shown to the group. The results grid shows names only.
- Take a script lock on every vote and every finalize, so two submissions arriving together cannot cross.
- Storage must tolerate new columns. Read cells through a helper that treats a missing cell as empty, and backfill the header row on load, so polls created before I extend the sheet still open.

Features I want beyond the free tier of Doodle. *[Keep the ones you want.]* A voting deadline, after which votes are refused and I get a nudge to pick a time. An invitee email list — send the invitation at creation, then one automatic reminder to people who have not voted, in the final stretch before the deadline. A "who hasn't voted yet" view that only I can see. Hidden polls, where voters see the totals but only I see the names. An email to me each time someone votes. A check that flags any proposed time that collides with something already on my calendar, before I send the poll out.

Reminders and deadline nudges run from a `pollReminders()` function on an hourly time-driven trigger. Tell me to install that trigger in DEPLOY.md. Give each poll a flags column so every reminder goes out exactly once.

Optional extras. *[Keep the ones you want, or delete this block.]*

- An edit page, gated by the organizer token, where I can change the title, note, length, deadline, invitees, privacy and the times while the poll is open.
- When times change on edit, carry each existing answer across by its time value, never by position in the list. A new time starts at "no"; a removed time drops out.
- A switch on the edit page, off by default, to tell everyone who already answered. Fixing a typo should not email thirty people. When it is on, the email shows the new list. It says that answers on times that did not change are kept, and that a time that moved or is new counts as a no until they answer it. People only need to act if the new list no longer works for them.
- A platform label and an organizer-supplied join link on each time, shown on the vote page, the results, the event and the emails.
- Time zone conversion: I enter times in my zone, and every voter sees the times in their own.
- When re-syncing guests onto the event, read the event first, keep any guest who is not a voter, preserve RSVPs, and set `guestsCanInviteOthers` to true, so guests I add from the calendar survive.
- Sign-up mode: each time is a session with a set number of places, one claim per person, and a session that fills closes itself.
- In sign-up mode the claim is the booking. The person goes straight onto that session's calendar event, and switching sessions frees the old place.

Build me the vote and results page, the poll creation page, the complete Apps Script `.gs` file, and a DEPLOY.md with setup instructions. Use my colors and fonts — *[paste your brand specs, or write "ask me"]*. Ask me any clarifying questions before you start.

Why it works. A poll needs somewhere to keep answers between visits, and naming a Google Sheet as that store is what keeps the whole thing free. The secret organizer token is the other load-bearing detail — it separates the person who picks the time from the crowd who only votes, with no login screen anywhere.

4 What It Takes

Need	What to use	Cost	Notes
Calendar account	A dedicated Google account	Free	Use a role account such as <code>scheduling@</code> , not your personal one. Events live on its calendar and mail comes from its address.
Backend	Google Apps Script	Free	No server to run. Executes as the calendar account, with a generous free quota. One project can serve both tools.
Poll storage	Google Sheets	Free	Created automatically on the first poll. Open it any time to inspect a poll. Section 7 says which cells are safe to edit by hand.
Web hosting	Any static host	Free or existing plan	Anything that serves plain files works — your current host, Netlify, Cloudflare Pages, GitHub Pages.
Domain	A subdomain of a domain you already have	Existing domain	Something like <code>book.yourdomain.com</code> , with the poll living at <code>/poll/</code> under it.
Video links	Google Meet or Zoom	Free tier fine	Meet is created automatically. For Zoom, either store one standing link in the script or connect the Zoom API.
AI assistant	Claude	Your existing plan	Generates the code and guides you through the deploy.

What you do not need — no database, no paid scheduling tool, no login screen, no JavaScript framework, and no monthly fee.

5 Think These Through First

Your calendar is the database

Open slots are recurring events titled `... - OPEN`. That keeps it free and keeps you managing availability from the calendar app you already use. You need to be comfortable creating and editing recurring events — that is the one skill this asks of you.

THE DEPLOY TRAP — LEARN THIS BEFORE YOU DEPLOY

To update the backend, always use **Manage deployments** → **edit (pencil)** → **New version**. Choosing **New deployment** generates a brand new URL, and your live site, still pointing at the old one, quietly stops working. This is the single thing people get wrong most often.

EXACT-STRING MATCHING

The script finds open slots by an exact, case-sensitive title match, such as `Intro Call - OPEN`. A hyphen versus an em-dash, or `Open` versus `OPEN`, makes a slot invisible with no error anywhere. Copy your titles, never retype them.

THE RECURRING-EVENT GUEST BUG

Adding a guest in the same calendar patch that first edits a recurring instance spills that guest onto every occurrence in the series. The fix is a two-step claim — the first patch breaks the single date out as a separate exception, then a second patch adds the guest. Tell your assistant this up front, and check the generated script for it.

WHO IS ALLOWED TO CREATE A POLL

Voting should need nothing but the link. Creating a poll should need a passphrase, kept as a Script Property and never linked from a public page. Without that gate, anyone who finds the create page can open polls on your domain and send mail from your account.

CACHING

Every style or script change needs its version parameter bumped on both tags: `styles.css?v=6` becomes `?v=7`, and `app.js?v=6` becomes `?v=7` in the same edit. Miss the script tag and you get new CSS against stale JS, which looks like a broken page. The stale copy is often held by your host, not your browser, so an incognito window proves nothing. Only a changed query string gets through.

HOSTING HYGIENE

Uploading into the wrong folder, or letting a zip extract into the parent, can overwrite a whole site. On shared hosts a zip extract can also report success and write nothing. Create nested folders such as `poll/` by hand, upload files one at a time into the folder you have confirmed, and open the live URL of each file to check it landed.

THE CALENDAR SERVICE AND THE TIME ZONE

In the script editor, add the Google Calendar API under Services with the identifier exactly `Calendar`. A different identifier makes every calendar call fail. Then open Project Settings and set the project time zone to yours. That one setting drives every date and time the script writes into an email, so a wrong zone means every confirmation is off by hours.

WHAT BLOCKS TIME, AND WHAT DOES NOT

By default only a timed event with a title blocks a slot. An all-day event or an untitled one does not, unless you tell the assistant it should. And blocking time on the calendar the script runs as does nothing on its own; that calendar has to be in the free/busy list like any other, or the script never looks at it.

SENDING AS YOUR REAL ADDRESS

Mail goes out from the role account unless Gmail has a verified "Send mail as" alias for the address you want. Set that up in the role account's Gmail settings, have the script use the alias, and give it a fallback to the default sender in case the alias is missing. Set the account's display name too, or invites arrive from a bare address.

"PAGE NOT FOUND" FROM THE /EXEC URL

This almost never means broken code. It usually means the deployment behind that URL was archived, often by clicking New deployment and then tidying up. Open Manage deployments, find the live one, and point the frontend config at its URL.

PRIVACY

A page shared by link only stays private because the link stays private. If it should never surface in search, block it with a `robots.txt` file, a `noindex` tag in every page head, and a `noindex` header from your host. Decide too what a voter can see — showing names is friendly for a small team, and totals-only is kinder for a mixed group.

6 Step by Step

Phase 1 — Prep, before any code

- 1 Create or choose the dedicated Google account whose calendar this will use.
- 2 Decide your meeting types and their exact event titles, such as `Intro Call – OPEN`.
- 3 Confirm where you will host and which subdomain you will use.
- 4 Gather your colors and fonts, or decide to let the assistant choose.

Phase 2 — Build with the assistant

- 1 Paste in Starting Prompt 1 with your details filled in, and answer its clarifying questions.
- 2 Have it produce the HTML pages, the shared CSS and JS, the full Apps Script `.gs` file, and a `DEPLOY.md`.

Phase 3 — Stand up the backend

- 1 Go to `script.google.com` signed in as the calendar account. Create a project, paste in the `.gs` code, and save. Under Services, add the Google Calendar API with the identifier exactly `Calendar`. In Project Settings, set the time zone to yours.
- 2 Choose Deploy, then New deployment, then Web app. Set it to execute as you, with access for anyone. Authorize it, then copy the `/exec` URL.
- 3 Paste that URL into the frontend config file. From here on, every backend update goes through Manage deployments → pencil → New version. Never New deployment.

Phase 4 — Stand up the frontend

- 1 Point your subdomain at your host. Upload the HTML, CSS, and JS into one confirmed folder, a file at a time.
- 2 Pre-create your recurring `... - OPEN` calendar events as far out as you want to take bookings. A year ahead is a reasonable default.

Phase 5 — Test before you share it

- 1 In the script editor, run `testSetup()`. It should report that it can read the calendar, how many OPEN blocks it found, and the next few bookable start times. If any of those is empty, fix that before touching the page.
- 2 With `LIVE` still false, book a test slot from another browser. Confirm the slot disappears and that your admin notification arrives. Guest emails stay off in this mode, so a test booker's inbox stays clean.
- 3 Set `LIVE` to true, redeploy with Manage deployments → pencil → New version, and book one more test slot. This time confirm the booker's confirmation and calendar invite arrive too. Then decline it the Section 7 way.
- 4 Learn the three calendar moves in Section 7, then share your link.

Phase 6 — Add the poll

- 1 Paste in Starting Prompt 2, pointing it at the Apps Script project you already have.
- 2 Paste the updated script over the old one, add your `POLL_CREATE_KEY` as a Script Property, and run the script once from the editor so it authorizes access to Sheets and creates its spreadsheet.

3

Install the hourly trigger. In the editor, open Triggers (the clock icon), choose Add Trigger, pick the `pollReminders` function, set the event source to Time-driven and the type to Hour timer, every hour. Without this trigger, deadlines and reminders never fire and nothing tells you.

4

Redeploy with Manage deployments → pencil → New version, upload the poll pages beside your booking page, then run one throwaway poll end to end — create it, vote from a second browser, and finalize it — before you send one to a group.

7 Running It Day to Day

To do this	Do this
Decline a booking	Rename the event back to ... – OPEN and remove the guest. With no cache, it reopens on the next page load. If your build caches the slot list, which is worth doing on a busy calendar, it can take up to about 10 minutes unless the script clears that cache on every claim and every decline. Ask for that.
Skip a date, such as a holiday	Delete that single occurrence, not the series.
Change one week's time	Edit the single occurrence, not the series.
Add a new weekly time	Create another recurring ... – OPEN series. The page picks it up on the next load.
Stop taking bookings for a while	Delete or rename the upcoming OPEN events. The page shows nothing available.
Fix a poll by hand	Open the spreadsheet the script created. The title, note, deadline, invitee list and privacy cells are safe to edit, and the change is live at once. Never edit the times cell. Votes are stored as position-based lists, so adding or removing a time there misaligns every vote already cast and blocks re-voting. Change times through the edit page if you built one (Section 8), or create a new poll.
Change a booking rule	Edit the constant at the top of the script, then Manage deployments → pencil → New version.
Change what the script writes to events	Events already on the calendar keep the old shape. Ask for a one-time helper function that updates the existing events with sendUpdates: "none" , so nobody gets a fresh invite email for a change they cannot see.

8 Optional Extras

None of these is needed for a working build. Each one is already written into the prompts as an option; keep the lines you want and delete the rest.

Optional: two shapes for the booking page

The plain version is a flat list of upcoming times. The other is a month calendar: click a day, see its times, then fill in the form. The flat list is easier to build and fine for a few slots a week. The month view is kinder when you offer many times a day. Prompt 1 asks you to pick. It also offers a second way to set availability: one recurring daily OPEN block that the script slices into start times, instead of a year of individual recurring slots. That means less calendar upkeep, at the cost of a slightly more involved script.

Optional: a poll edit page

Without it, a poll's times are fixed the moment you create it. With it, the organizer link opens an edit page where you can change the title, note, length, deadline, invitees, privacy and the times while the poll is still open. The one rule that matters: when the times change, every answer already cast has to be carried across by the time it belongs to, never by its position in the list. Get that wrong and people inherit a neighbour's answer.

Add a switch that emails the voters after an edit, off by default so a typo fix sends nothing. The email has to say outright that nobody needs to do anything unless the new list stops working for them. An email that only asks people to check their answer leaves them guessing what checking means.

Optional: time zones

If everyone who votes is in your zone, skip this. If not, ask for conversion: you enter times in your zone, and the page shows each voter the same times in theirs, with the zone name printed beside them. The stored times never change; only the display does.

Optional: sign-up mode

The same poll, turned around. Instead of voting on which time wins, every time is a session that will run, and each one holds a set number of places. A person claims one session, the claim is the booking, and they go straight onto that session's calendar event. A session that fills closes itself, and switching to a different session frees the old place. Useful for office hours, workshop cohorts, or any run of repeated sessions. Prompt 2 has two lines for it under Optional extras.

This is the same setup running live booking pages and group polls today. Once it is standing, the running cost stays at nothing.

Build Your Own Scheduling Tools — a guide to a self-hosted booking page and meeting poll.