FREE GUIDE

The *Automated* Content System.

The bots draft, you approve.

55 lessons · 10 modules · Free

viktorszabo.

viktorszabo.com

CONTENTS

The lessons

55 lessons across 10 modules (Modules 0 to 9). Lesson counts include each module's overview. This PDF is the full lesson text. The 90 files the lessons describe are in the zip.

Module 0 · Start here: what "automated, you approve" really means

- Module 0 · Start here: what "automated, you approve" really means 4
- 0.1 First step: import dr eggbot and let it set up your team 6
- 0.2 Your team of bots: who does what, and who hands work to whom 7
- 0.3 The one rule: bots draft, you approve 9
- 0.4 What you still do yourself 11
- 0.5 The tool list 12

Module 1 · The shared brain: brief, rules, approvals, routines

- Module 1 · The shared brain: brief, rules, approvals, routines 14
- 1.1 The brand brief every bot reads first 16
- 1.2 The approvals log: every yes, written down 17
- 1.3 The routines file 19
- 1.4 Honesty rules: no money claims, no invented stats, claim checks 20

Module 2 · Brand: logo, brandbook, and an optional avatar

- Module 2 · Brand: logo, brandbook, and an optional avatar 22
- 2.1 Three logo directions, pick the simplest 23
- 2.2 The brandbook as the source of truth for every bot 25
- 2.3 Bonus: avatar (optional) 26
- 2.4 The cover generator: every cover image in one style 28

Module 3 · Research engine: never run out of fresh ideas

- Module 3 · Research engine: never run out of fresh ideas 30
- 3.1 The daily trend feed routine 32
- 3.2 Turning the feed into post ideas for each channel 33
- 3.3 The weekly trend brief 35

Module 4 · A website your bots build and update

- Module 4 · A website your bots build and update 37
- 4.1 Site brief → your bot builds → you review the previews 39
- 4.2 Hosting: putting the site online 40
- 4.3 SEO and AEO basics 42
- 4.4 Scheduled publishing: approved posts go live on their date (optional) 43
- 4.5 Contact forms without a backend (FormSubmit) 44

Module 5 · Instagram, published for you (after your yes)

- Module 5 · Instagram, published for you (after your yes) 46
- 5.1 The daily drafts routine: tomorrow's set in one message 48
- 5.2 Rendering carousels and Stories in the brand style 49
- 5.3 Publishing through the API 51
- 5.4 "API-publishable only": no manual steps in your drafts 52

5.5 Comment-to-DM with ManyChat: links only by DM 54
5.6 Engagement: comment batches for quick approval 56

Module 6 · X: scheduled posts, replies, honest scoreboard

Module 6 · X: scheduled posts, replies, honest scoreboard 58
6.1 Weekly batches into the scheduler 60
6.2 Replies, and when a standing permission makes sense 61
6.3 The post-score tracker 62
6.4 The weekly scoreboard, flat weeks included (optional) 64

Module 7 · YouTube without a camera

Module 7 · YouTube without a camera 66
7.1 Script and claim check 68
7.2 Narration in your own voice: recorded or cloned 69
7.3 Bonus: avatar render (optional, if you chose the bonus avatar) 71
7.4 Screen captures and other visuals, done by the bot 72
7.5 The Remotion edit 73
7.6 Music, mix to -14 LUFS, captions 75
7.7 Thumbnails, metadata, upload, pinned comment 76

Module 8 · Reels and short video

Module 8 · Reels and short video 78
8.1 A monthly credit plan, so nothing goes unused 79
8.2 Reels style: what the polished accounts do 80

Module 9 · Running your AI team

Module 9 · Running your AI team 83
9.1 The CEO bot: daily summary and batched decisions 85
9.2 Daily self-review per bot 86
9.3 The maintenance bot: chat and routine healthchecks 87
9.4 Handoffs through shared files, not memory 88
9.5 What to switch off 90

MODULE 0

Start here: what "automated, you approve" really means

Import dr eggbot, map your team, the one rule, what stays human, the tool list.

You'll have: a starting team of bots and a clear line between their work and yours.

6 lessons · 3 files

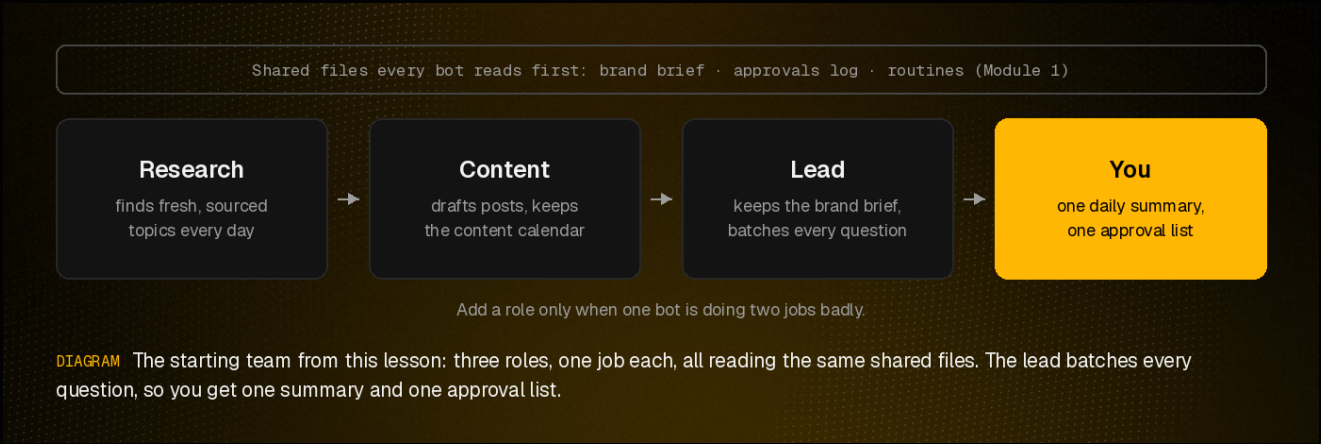


DIAGRAM The starting team from this lesson: three roles, one job each, all reading the same shared files. The lead batches every question, so you get one summary and one approval list.

The workflow from Module 0.

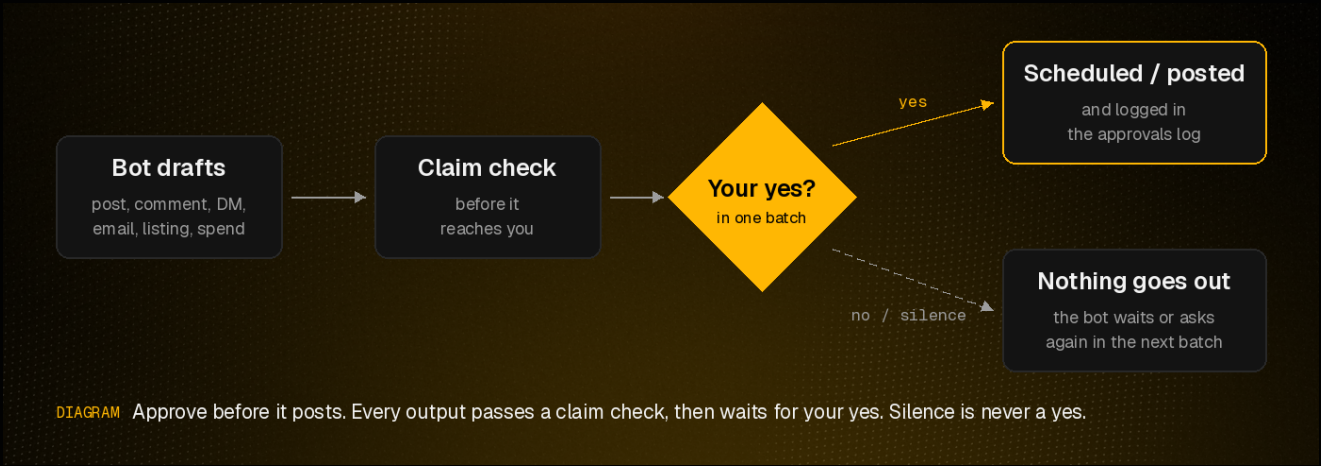


DIAGRAM Approve before it posts. Every output passes a claim check, then waits for your yes. Silence is never a yes.

Every draft passes a claim check, then waits for your yes.

LESSON

Module 0 · Start here: what "automated, you approve" really means

In this module: 6 lessons, 3 files. Reading time about 25 minutes.

What this free guide is

This free guide shows you how to build a brand with a small team of AI bots. The team researches, writes, designs, renders and schedules. You approve what goes out. It works for anyone building a brand: a creator, a freelancer, a coach, a consultant, an artist, a small or local business.

My own team runs on Grok Bot. Each bot has one job, works from the same shared files, and hands work to the others through those files. Throughout the free guide I show how I did it ("here's how I did it"), and then tell you what to adapt for your brand. Where an example is made up, I say so.

The honest version of the promise

Every step of content production can be handed to an AI team. That is not the same as everything. Three things stay with you:

- Approval. You say yes or no to everything that goes out. This is a design choice, and Lesson 0.3 explains why it is the most important rule in the whole system.
- One-time setup only a person can do. Creating accounts, signing in, passing platform checks, connecting payment methods, approving spend.
- Taste. Which logo, which thumbnail, which music mood. The team gives you options. You pick.

So the honest headline is this: *the AI team does the production work; your job is to say yes or no.* Automated, with you approving.

What this free guide is not

- It is not a promise of followers, sales, clients or rankings. I make no promises about any of those, and I don't share my own numbers. The free guide teaches the method; what it brings you depends on your brand, your offer and your audience.
- It is not coaching. There are no calls, live sessions or community. Support is by email (hello@viktorszabo.com), for access or download problems only.
- It is not "set it and leave it". The team works every day, and it asks you every day.

Where the files live (read this once)

The free guide comes with templates in `files/`. You don't need to build folders on your computer. You ask your bot to create and keep these files in its own workspace, using the templates, and to show them to you in chat or attach them whenever you want to read or edit them. If your bots run in the same workspace, they all see the same files. If they don't, you attach or paste the file into the other bot's chat. Module 1 shows exactly how. If you prefer to keep files on your own computer, you can, but then you paste or attach them to your bots yourself.

How the free guide is laid out

Each lesson has the same parts:

- What it does: the job in one or two paragraphs.
- Steps: what to ask your bot for, and what you do yourself, in order.
- Here's how I did it: my own setup as an example, with private details removed.
- You approve: exactly where your yes is needed.
- Common mistakes: the one thing people (and bots) usually get wrong.

The templates use `{{PLACEHOLDERS}}` in capitals. Each one starts with a short header that says what to replace.

The lessons in Module 0

- 0.1 First step: import dr eggbot and let it set up your team of bots.
- 0.2 Your team of bots: who does what, and who hands work to whom.
- 0.3 The one rule: bots draft, you approve.
- 0.4 What you still do yourself: accounts, sign-ins, platform checks, spend, taste.
- 0.5 The tool list: what I use and why, with "approve before paying" boxes.

Files in this module

- `files/00-start-here/00-team-map.md`
- `files/00-start-here/00-what-stays-human.md`
- `files/00-start-here/00-tool-checklist.md`

> You approve
> Nothing yet. Read Module 0 before you create any account or pay for any tool.

Common mistakes: starting with the tools. Start with the team and the rules (Lesson 0.1 and Module 1). A team without shared rules produces a lot of confident drafts that contradict each other.

LESSON

0.1 First step: import dr eggbot and let it set up your team

What it does

Before you write a single rule or template, you need the bots themselves. The fastest honest way I know to get them on Grok Bot is a bot that designs bots.

dr eggbot is a bot on the Grok Bot marketplace. Its page is here: dr eggbot. It is made by Lauren Tan, not by me. Its listing describes its job as designing Grok Bots: it asks you a few preference questions, then creates the bots. Each bot it designs gets one job, one voice, and an explicit list of things it must not do (for example, a bot that drafts does not send). That matches the way this free guide works: one job per bot, and nothing goes out without your yes.

Steps

- Open the Grok Bot marketplace and go to dr eggbot.
- Click Import Bot. You sign in and confirm the import yourself.
- Tell dr eggbot what you want to build, in plain words. For example: who you are, what your brand is about, which channels you want to use, and what you want each bot to do. Say clearly that bots draft and you approve.
- Answer its preference questions.
- Let it create your team. Read what it made: each bot's name, its one job, and what it must not do. Change anything that doesn't fit before you start using them.
- Start small. Three bots is a good first team (Lesson 0.2). You can always ask dr eggbot or your lead bot for more later.

What it does not do

- It does not guarantee a particular set of bots. What it creates depends on what you tell it and how you answer its questions.
- It does not make your accounts, sign in for you, or pay for tools. Those stay with you (Lesson 0.4).
- It does not write your brand rules for you. That is Module 1, and you approve every word.

Here's how I did it

My team grew one role at a time: a lead bot that keeps the brief and batches my approvals, then content, research and the rest as each job needed its own bot. If I started again today, I would begin by telling dr eggbot what I want and letting it propose the team, then trim the list to the three roles I need first.

> You approve
> The import, and the team it proposes: each bot's job and its "must not do" list. Edit them until they read like your team.

Common mistakes: asking for ten bots on day one. More bots means more messages and more to approve. Start with three and add a bot only when one is doing two jobs badly.

LESSON

0.2 Your team of bots: who does what, and who hands work to whom

What it does

A team works when every member knows its one job and where to pick up and drop off work. Your team is a set of bots, each with a role, its own routines, and the same shared files (Module 1). They share files, not memory.

Start with three

You do not need ten roles on day one. A good starting team for almost any brand is three:

- Lead: keeps the brand brief, sends you one daily summary, and batches every question into one quick approval list.
- Content: drafts posts for your main channel and keeps the content calendar.
- Research: finds fresh, sourced topics in your niche every day (Module 3).

Add a role when one bot is doing two jobs badly. Common next roles: a second channel (X, YouTube), engagement (comments and DMs for your approval), images and short video, and a website or SEO bot.

Example: my team

This is my team, as an example of how far it can grow. Yours will look different, and that is the point: adapt the roles to your brand.

Role	Owns	Hands work to
CEO	The brand brief, the daily summary, batching every bot's questions into one quick approval list	Me (one message a day)
CMO	Instagram feed and Stories, the content calendar, the website and its design	Engagement, X Channel Builder, SEO & AEO Desk
Engagement	Daily comment and DM batch for my approval	Me (one daily batch)
X Channel Builder	X posts, replies, the queue check and weekly post scores	CEO, CMO
Short Video Creator	Avatar poses, Reels, every image or video credit spend	CMO, Video Editor
Video Editor	Edits for YouTube and Reels	CMO
Digital Products (optional)	Offers and listings, only if you have them	CMO
Business Development	Leads and partners (every lead comes to me before any contact)	Me
SEO & AEO Desk	Keywords, search reports, site metadata	CMO
Research	The daily trend feed and the weekly trends brief	Every content bot

Plus one maintenance bot that reviews chats and routines for waste and friction, and proposes fixes.

What to adapt: a local business might swap X for Google reviews. A coach might replace the products bot with a bookings or enquiries bot. An artist might put more weight on images and short video. Keep the shape (one job per bot, one lead who batches questions), change the roles.

Steps

- Open `files/00-start-here/00-team-map.md` and paste it into your lead bot's chat, or ask the bot to create the file from the template in its workspace.
- Write your roles in the table. Start with three. Give each one a single sentence saying what it owns.
- For each role, fill in "reads first" (which shared files it opens before working) and "hands to" (who gets its output).
- Mark one role as the only one allowed to spend credits on each paid tool.
- Ask the lead bot to keep the map as a shared file (Module 1) and to show it to you in chat whenever you ask.

Here's how I did it

The avatar pose flow shows a clean handoff. The CMO needs a new pose for a carousel. It does not generate one. It adds a row to a shared request file with status `open`. The Short Video Creator reads that file before generating anything, skips rows marked `replaced` or `cancelled`, checks whether an existing image already covers the request, and only then spends credits. It writes the result back into the row.

> You approve
> The team map itself: who owns what, and which single bot may spend on each tool. Change it whenever it stops matching reality.

Common mistakes: two bots owning the same channel. When a job moves from one bot to another, write it into the brand brief the same day, and say which bot it moved from.

LESSON

0.3 The one rule: bots draft, you approve

What it does

Every part of the system follows one rule: nothing gets posted, sent or published, and no money is spent, without your yes. Bots research, write, design and schedule. You approve. This is what keeps an automated system safe to run every day.

My brand brief states it in plain words, and every bot reads it before working. Here is the template version:

> "Nothing gets posted, sent or published, and no money is spent, without {{YOUR_NAME}}'s yes. That includes comments, DMs, outreach emails and new subscriptions, unless {{YOUR_NAME}} has already given a standing permission for that exact thing."

Why it is a feature, not a bottleneck

- It catches wrong claims. In my setup, a queue check once found a scheduled post with an out-of-date fact about what I offer. The fix came to me as a single yes/no before the post went out.
- It makes approval fast. Because bots know they must ask, they batch their questions. My lead bot collects every open question into one numbered list, and I answer most items in one or two words.
- It keeps you in your own voice. A draft is cheap. A wrong post under your name is not.

Standing permissions

A standing permission is a yes you give once for one exact, low-risk, repeatable thing. Keep them few. Mine: my X bot may post short replies on X without asking each time, because replies are frequent and time-sensitive, and a reply that waits a day is useless.

Even that permission has guard rails: the routine logs every reply to a file, sends me a one-line count each day, and reports immediately on any platform warning.

Other things stay behind approval on purpose. I decided bots would not post outbound Instagram comments at all. The bots prepare a list of posts with the exact comment for each, and I post them by hand.

Steps

- Ask your lead bot to put the rule above, with your name in it, at the top of your brand brief (Module 1).
- List every type of output your team produces: posts, comments, DMs, emails, listings, site changes, purchases.
- Next to each, write "ask every time" or "standing permission". Start with "ask every time" for all of them.
- Give a standing permission only when the action is small, frequent and easy to undo. Write it down with its limits (how many a day, what to report).
- Ask every bot to log each approval in the approvals log (Lesson 1.3).

Here's how I did it

My Instagram drafts routine runs every evening. It renders the next day's post and Stories, writes captions and alt text, and sends everything in one message. I reply once. Only after that yes does the bot schedule each item. No yes, no post.

> You approve
> Every post, comment, DM, email, listing, site change and purchase, unless you wrote a standing permission for that exact thing. You also approve each standing permission itself.

Common mistakes: treating silence as a yes. A bot that hears nothing must wait or ask again in the next batch, never go ahead.

LESSON

0.4 What you still do yourself

What it does

This lesson is the honest list of what an AI team cannot do for you, or should not. Knowing it up front saves you from a week of "why is the bot stuck?"

The five things that stay human

1. Approvals. Covered in Lesson 0.3. Everything that goes out, and every spend.

2. Accounts and sign-ins. You create the accounts and you sign in. A bot can use a connector or a browser session after you have connected it, but it should never create accounts in your name on its own or handle your passwords. In my setup, when a step needs a sign-in, the bot stops and asks me to do that one step.

3. Platform checks. Platforms check that a real person is behind an account. For example, a video platform may ask for an ID or video check before it turns on advanced features such as clickable links. No bot can pass that for you. A bot can set a reminder for you; it changes nothing on the platform.

4. Spend. Buying a subscription, topping up credits, connecting a card. The team can research the options and show you the plan and where to read the price. You decide and you pay. In my setup I pay at the checkout page myself, and the bot logs it afterwards. When a bot notices a setting nobody approved, such as an auto top-up, it asks about it instead of leaving it on quietly.

5. Taste. The logo, the website direction, the thumbnail, the music mood, the favicon. Bots make options and explain the trade-offs. You pick. I chose my website direction and my favicon from options the team built. Those were my calls, and the free guide treats them that way.

Steps

- Open `files/00-start-here/00-what-stays-human.md`, or ask your lead bot to create it from the template and show it to you.
- For each channel you plan to use, tick which sign-ins and platform checks you will need to do yourself.
- Set aside a short daily slot for approvals. Ask your lead bot for one daily summary, so this is one sitting, not twenty interruptions.
- Tell the lead bot to batch every "I need you for one step" request into that summary.
- Ask the lead bot to write your spending rule in the brand brief: which bot may ask, and that nothing is paid without your yes.

Here's how I did it

My lead bot's daily summary lists what is waiting on me, but only steps no bot or connector can do. It also reads the approvals log first, so nothing already approved shows up as waiting again. That keeps the human list short and real.

> You approve
> Every account you create, every sign-in, every platform check, every payment, and every taste decision. These are yours by design.

Common mistakes: asking a bot to "just get past" a sign-in or a verification step. It should not, and a well-set-up bot will refuse. Do the one step yourself and let the bot continue.

LESSON

0.5 The tool list

What it does

This lesson lists the tools my setup uses, what each one does in the system, and which module covers it. I list no prices. Prices and plans change and differ by country, so at the time you read this, check each tool's own pricing page before you pay.

Affiliate disclosure: some links in this free guide are affiliate links. If you sign up through one, I earn a commission, at no extra cost to you. Each one is labelled "affiliate link" right next to it. I only link tools I use or name in the free guide, and you don't need any of them to follow the method.

The tools, by job

Job	Tool I use or name	Module
The AI team itself	Grok Bot (one bot per role, with routines); start with dr eggbot (Lesson 0.1)	0, 1, 9
Shared files and rules	Plain Markdown files your bots create and keep in their workspace	1
Website hosting	Cloudflare Workers (static assets), or any static host	4
Website lead form	FormSubmit (form posts to an email address, no backend)	4
Search data	Google Search Console, a keyword tool	4
Instagram publishing	Instagram API through a Composio connector	5
Comment-to-DM	ManyChat	5
Avatar poses (optional bonus), short video	Higgsfield	2, 8
Voice clone	ElevenLabs	7

Job	Tool I use or name	Module
Talking avatar video	Hedra	7
Video editing in code	Remotion	7
X scheduling	X's own scheduler, Tweet Hunter	6
Scheduling across several channels (optional)	Postiz	any channel module

Links for the tools that have an affiliate program I'm part of:

- Higgsfield: higgsfield.ai (affiliate link: I earn a commission if you sign up, at no cost to you)
- Postiz: postiz.pro (affiliate link: I earn a commission if you sign up, at no cost to you)
- ManyChat: manychat.com (affiliate link: I earn a commission if you sign up, at no cost to you)

Many of these have a free tier. You do not need all of them to start, and you can swap any of them for a tool you already use.

Steps

- Open `files/00-start-here/00-tool-checklist.md`, or ask your lead bot to create it from the template and show it to you.
- Pick the minimum set for your first month. A sensible start: the AI team, a website host, and one social channel.
- For each tool, fill in the plan you want and the price you read on its pricing page.
- Tick the "approved by me" box only after you have read the price yourself.
- Name the one bot that may use each paid tool (Lesson 0.2).
- Once a month, ask your lead bot to compare what you pay for with what you used, so nothing sits unused.

Here's how I did it

One rule worked well for me: one bot owns each credit-based tool. My Short Video Creator is the only bot that spends image and video credits. Other bots request images through a shared request file. Each run logs what it spent, so I always know the balance. My lead bot keeps a tracker of every subscription and credit balance, so nothing is bought twice.

> You approve
> Every plan, every upgrade, every top-up, and every auto-renew setting. Read the price on the tool's own page. A bot's summary is a starting point, not the final number.

Common mistakes: buying every tool in week one. Start small. Add a tool when a lesson needs it and a bot is ready to use it fully.

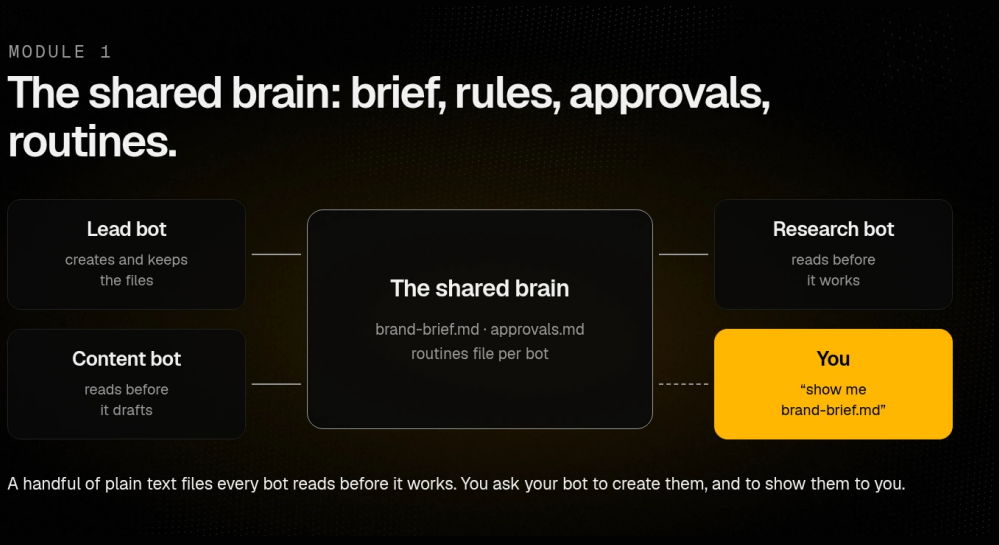
MODULE 1

The shared brain: brief, rules, approvals, routines

Your lead bot creates and keeps a brand brief, an approvals log and a routines file. Honesty rules and a claim-check prompt.

You'll have: the files every bot reads before it works, and a written record of every yes.

5 lessons · 5 files



The workflow from Module 1.

LESSON

Module 1 · The shared brain: brief, rules, approvals, routines

In this module: 5 lessons, 5 files. Reading time about 30 minutes. This is the module everything else depends on. The lesson count includes this overview.

Why a shared brain

A bot starts each task fresh. If the facts it needs live only in a chat from last week, it will guess. Guessing is how you get a post with the wrong price, an offer that does not exist, or a question you already answered three times.

The fix is simple and a little boring: a handful of plain text files that every bot reads before it works. In my setup these are:

- `brand-brief.md`: who you are, the goal, the rules, who does what.
- `approvals.md`: every yes, one line each, newest at the bottom.
- A routines file per bot, listing its scheduled jobs.

Files beat memory for three reasons. Every bot sees the same version. You can read and correct them. And when a bot is wrong, you can point to the line that should have stopped it.

Who creates the files, and where they live

You don't build folders or copy files around. You ask your bot to do it. The templates in `files/01-shared-brain/` are starting points you paste or attach in chat. Your lead bot creates each file in its own workspace from the template, fills it in with you, and keeps it current.

How your bots share the files depends on your setup, and it's worth being clear about it:

- Bots that run in the same workspace see the same files. One bot updates `approvals.md`, the next one reads the new line. That is how my team works.
- Bots that don't share a workspace can't see each other's files. Then you attach or paste the file into the other bot's chat, or ask the lead bot to send it the current version.
- You see a file whenever you ask: "show me brand-brief.md" or "attach approvals.md". Ask for that often. It's how you check the bots wrote down what you meant.

If you prefer, you can also keep your own copy on your computer and attach it to bots when needed. Just decide which copy is the real one, and tell the bots.

What you will build

By the end of this module your lead bot will be keeping the core files for your brand, plus a claim-check prompt your bots run before anything comes to you for approval.

The lessons

- 1.1 The brand brief every bot reads first.
- 1.2 The approvals log: every yes, written down, read before asking again.
- 1.3 The routines file: schedule, status, what it does, quiet when nothing is new, last run.
- 1.4 Honesty rules: no money claims, no invented stats, "made with this setup" wording, and claim checks.

Files in this module

- `files/01-shared-brain/01-brand-brief-template.md`
- `files/01-shared-brain/01-approvals-log-template.md`
- `files/01-shared-brain/01-routine-file-template.md`
- `files/01-shared-brain/01-team-rules.md`
- `files/01-shared-brain/01-claim-check-prompt.md`

Steps for the whole module

- Tell your lead bot: "Create and keep our shared files: brand-brief.md, approvals.md, and a routines file for each bot. Use these templates." Attach or paste the templates.
- Fill in the brand brief together first (Lesson 1.1), then the approvals log (1.2).
- Ask the bot to leave the approvals log empty except for its header. It fills up as you say yes.
- Ask the bot to add one line to every bot's instructions: read the brief and the approvals log before any task.
- Ask the bot to show you each file in chat or attach it, and read it yourself.
- If your bots don't share a workspace, ask the lead bot how it will get the current files to the others, or attach them yourself.

> You approve
> The brand brief and the team rules. They are your voice and your limits, so they should be in your words. Bots may suggest edits; you accept them.

Common mistakes: writing the brief once and never touching it again. Mine has an "updated" line at the top, and the lead bot updates it the same day anything changes, such as a role moving from one bot to another.

LESSON

1.1 The brand brief every bot reads first

What it does

The brand brief is the one file that tells every bot who you are, what you are aiming for, what the rules are, and who does what. It is the first file in every role prompt's "read before working" list.

The structure I use

My brief has three parts, plus a short header:

- Header: an `updated:` line saying when and which bot changed it, then the approval rules restated in two or three sentences. These come first because they matter most.
- Part 1: What changes (summary). The goal, the positioning, the channel plan, the content direction, new offers, how we work. Bullet points, short.
- Part 2: Who does what. One bullet per role, saying what it owns and what it does not.
- Part 3: Your own words, verbatim. The original message you gave the team, pasted in unchanged.

Part 3 is the one most people skip, and it is the most useful. When a summary in Part 1 is unclear, a bot can go back to what you actually said. When a bot misreads something, the lead bot adds a correction line at the top (for example, "Correction: the tool name in my message is misspelled; the right name is ...") instead of editing your words.

Steps

- Attach or paste `files/01-shared-brain/01-brand-brief-template.md` into your lead bot's chat and say: "Create brand-brief.md from this template and keep it current."
- Write or record a long, honest message to your team: what you want, what you do not want, what worries you. Do not polish it.
- Ask the bot to put that message into Part 3, unchanged.
- Ask it to write Parts 1 and 2 from it. Then ask it to show you the whole file in chat, and check Parts 1 and 2 against Part 3 yourself.
- Ask it to put the approval rule (Lesson 0.3) at the very top.
- Ask it to keep an `updated:` line and change it on every edit.
- Ask it to add the brief to every bot's instructions as required reading. If a bot doesn't share the workspace, attach the brief to that bot.

Here's how I did it

My brief's positioning line went through a real decision. The team proposed options, I picked one, and the brief now says to use it word for word:

> "I run AI experiments and write them down: what worked, and what didn't. I work with Claude and Grok Bot. Free notes and guides at viktorszabo.com."

It also notes that the old line is retired, and that each live bio or site change still comes to me for a look first. That single note stopped bots from quietly rewriting bios across platforms.

Part 2 in my brief is specific about boundaries. The engagement role, for example, says what it covers and also what it does not: "Not X, not feed or Reels, not leads."

> You approve
> Parts 1 and 2 after the bot drafts them, and every later change to the rules or the positioning. Part 3 is yours and is never edited.

Common mistakes: putting private details in the brief "for context". Every bot reads this file, and some of what they write ends up public. Keep money targets, budgets and personal history out of it unless every bot truly needs them, and add a line saying which facts must never appear in public content.

LESSON

1.2 The approvals log: every yes, written down

What it does

The approvals log is a single file where every bot writes one line right after you approve something. Bots read it before asking you anything. If the answer is already there, they do not ask again.

The format I use

The file header explains itself in three lines, then the entries begin:

```
# Approvals log
Every bot appends one line right after {{YOUR_NAME}} approves something, newest at the bottom:
`- {{WHEN}} | bot | what was approved | where it lives`
```

Four fields: when, which bot, what exactly, and where the thing lives (a file name or a link). "What exactly" should be specific enough that another bot could act on it, including your own words in quotes when they matter.

Steps

- Attach or paste `files/01-shared-brain/01-approvals-log-template.md` into your lead bot's chat and say: "Create approvals.md from this template. Every bot appends to it after my yes."
- Ask the lead bot to add one instruction to every bot: "Append one line to approvals.md right after an approval. Read approvals.md before asking for one."
- Ask it to add a second instruction: when a bot acts on an approval, it logs that too ("acting on the earlier yes: scheduled, verified").
- Ask the lead bot to read the log before its daily summary, so nothing already approved shows as "waiting".
- Lines are never deleted. If something changes, the bot adds a new line that says what it replaces.
- If a bot works outside the shared workspace, it reports the line to the lead bot, or you paste it in. One log, not one per bot.

Here's how I did it

A typical pair of lines from my setup, with details made generic:

```
- {{WHEN}} | Content bot | Approved (my words logged): "yes, publish all three posts as written,
  at the times shown" | drafts file for the week
- {{WHEN}} | Content bot | Acting on the yes above: the 3 posts are scheduled (queue checked) | scheduler
```

The log also records the narrow edges of a yes. When I approved one profile change, the line said it covered "ONLY the change exactly as shown in chat", and that two related items were "NOT yet approved". The bot did not stretch one yes into three.

It records my changes, too. When I edited a bio myself after a bot had set it, the next line recorded that the bot must not change it back.

> You approve
> Nothing extra. The log is the record of what you already approved. Ask the bot to show it to you once a week and check it says what you meant.

Common mistakes: vague lines like "Instagram stuff approved". Write what, when, and where it lives. A vague yes gets read as a wide yes.

LESSON

1.3 The routines file

What it does

A routine is a job a bot runs on a schedule: a daily draft, a weekly report, a one-off reminder at an exact time. Each bot keeps a routines file that lists all of its routines in the same short format. You ask for it in chat ("show me your routines file") and see, in one minute, what the bot will do and when, and whether it worked last time.

The format I use

```
## Routine name
- schedule: when, in your time zone, and how flexible it is
- status: enabled | paused | one-off
- what it does: one or two sentences, including what it never does
- quiet when nothing new: yes / no, and what it sends if no
- last run: when, and the result (succeeded / failed and why)
```

Five fields. Each one earns its place:

- schedule with a reason. For example: "daily including weekends, because it's silent unless something happened."
- status, so a paused routine is not mistaken for a broken one.
- what it does, including the limits: "read-only", "never edits listings", "posts nothing without my yes".
- quiet when nothing new. This is how you stop routines from flooding you. A watcher that messages you only when something happens is useful. One that says "nothing new" every evening is noise.
- last run, with the real result. A failure is written as a failure.

Steps

- Attach or paste `files/01-shared-brain/01-routine-file-template.md` into each bot's chat and say: "Keep a routines file in this format, and update it every time you add, change or run a routine."
- When you ask a bot to create a routine, ask it to add the entry in the same turn and show it to you.
- Decide "quiet when nothing new" deliberately for each routine. Reports you rely on weekly should report zeros. Watchers should stay silent.
- For one-off routines, include "deletes itself after running", so they do not fire again later.
- Once a week, have your lead or maintenance bot read every routines file and flag waste: too frequent, noisy empty runs, dead routines, duplicates. If your bots don't share a workspace, ask each one to paste its routines file to the lead bot, or attach them yourself.

Here's how I did it

A watcher routine in my setup looks like this (details made generic):

```
## Daily check
```

```
- schedule: every evening (flexible ±15 min)
- status: enabled
- what it does: read-only check of a connected dashboard; writes what it found to a file
  with a timestamp; never edits listings
- quiet when nothing new: yes, it messages only when something happened or the read failed
- last run: {{WHEN}}, succeeded
```

Another honest detail: when one of my routines failed, its file said "failed" with the reason, and that the run was moved to a new time. Nobody had to guess.

I also learned that one-offs need a clear end. A one-off that isn't deleted can fire again on the same date next year. The "deletes itself after running" line prevents that.

> You approve
> Each new routine before it is switched on, and any change to what it does or who it messages. Routines that post or publish need your yes for their content every time, not just once for the routine.

Common mistakes: routines with no "quiet" rule. Ten bots each sending a daily "nothing new" message is a lot of unread messages every week. Make silence the default for watchers.

LESSON

1.4 Honesty rules: no money claims, no invented stats, claim checks

What it does

An AI team writes fast and sounds confident. Without rules, it will round numbers up, imply results nobody had, and describe a product by what it sounds like rather than what it does. Honesty rules are short, written down, and checked by a separate step before anything reaches you.

The rules I use

From my brandbook's voice section and team rules:

- No money promises or fake results. Never write "make money on autopilot" or "get rich".
- No invented numbers. Every number comes from a file or a source the bot opened. If a number is missing, the draft says "missing", it does not estimate.
- Never say "tested" for anything that wasn't actually run.
- "Made with this setup." Work produced by the team is described that way, not as something you did by hand, and not as something it is not.
- AI-generated scenes are fine as mood shots, never shown as your real past.
- Label affiliate links right where the link is, in plain words.
- Label fictional examples as fictional.
- Report flat weeks honestly. A weekly numbers report shows zeros as zeros.

The claim check

Before any post, caption, script, email or listing goes to you, a bot runs a claim check:

- List the claims. Every factual statement about a product or service: what it connects to, what's included, price, speed, results, who it's for.
- Find the source in brand-brief.md, then open any page or file the brief points to.
- Verify each claim. A claim passes only if the files say it or plainly show it.
- Fix, don't flag. Rewrite a failing claim so it matches the files. Drop what can't be verified.
- Note it on the draft: "claims checked against <files>", plus what changed.

Steps

- Attach or paste `files/01-shared-brain/01-team-rules.md` and `01-claim-check-prompt.md` into your lead bot's chat and ask it to keep them as shared files and add them to every bot's instructions.
- Add your own "never" list: topics, names, or facts that must not appear publicly.
- Put "run the claim check" in every content bot's role prompt, before the approval step.
- Ask one bot to also check queued posts that were approved before a product changed.

Here's how I did it

A claim check on my queued X posts once found several failing posts in one run. One said a free tool "checks your email and calendar". The tool's own guide says it works from notes you paste in. The check rewrote the line, and the fixed version came to me as one yes/no. Another failure was simply out of date: a post described my offers as they were before a change.

My X queue check now compares every upcoming post with the claim-check files and with recent posts, so near-repeats get flagged too.

> You approve
> The rules themselves, and every fixed draft. The claim check makes your yes faster; it does not replace it.

Common mistakes: checking claims against the bot's memory. The check only counts if the bot opened the product's files in that run.

MODULE 2

Brand: logo, brandbook, and an optional avatar

Three logo directions to pick from, a brandbook every bot follows, and a cover generator script your bot runs. Plus the optional avatar bonus.

You'll have: one look across every image, page and post.

5 lessons · 6 files

LESSON

Module 2 · Brand: logo, brandbook, and an optional avatar

In this module: 5 lessons (one of them an optional bonus), 6 files. Reading time about 25 minutes.

Why brand comes before content

Your bots will make a lot of images, pages and posts. If each one invents its own colours and fonts, your feed looks like ten different people made it. If they all follow one short brandbook, everything looks like you, even when no human touched the design.

This works the same for a creator, a café, a consultant or an artist. The core of this module is three pieces:

- A logo that is simple enough to render in code, at any size, on any background.
- A brandbook that bots read as the source of truth: colours with hex codes, type, layout rules, voice, and what you never do.
- A cover generator, a small script your bot runs to make every cover image (blog post, event, product, episode) in the same style from just a title and a subtitle.

Plus one optional bonus: an avatar with a pose library. Not everyone wants one, and nothing else in the free guide depends on it.

What stays human here

Taste. This is the module where your choices matter most. Bots generate options, explain the trade-offs and show you previews in chat. You pick the logo and approve the brandbook. My logo, website direction and favicon were my picks from options the team made.

The lessons

- 2.1 Three logo directions, pick the simplest.
- 2.2 The brandbook as the source of truth for every bot.
- 2.3 Bonus: avatar (optional).
- 2.4 The cover generator: every cover image in one style.

Files in this module

- `files/02-brand/02-brandbook-template.md`
- `files/02-brand/02-role-cmo.md` (instructions for the bot that owns brand and content)
- `files/02-brand/02-cover-generator/` (a working Python script plus a README; your bot runs it)
- `files/02-brand/02-bonus-avatar-pose-prompts.md` (only for the optional bonus)

Steps for the whole module

- Set up your content (CMO) bot with `02-role-cmo.md`: paste it in the bot's chat with the placeholders filled in, or ask the bot to fill them in from your brand brief.
- Ask it for three logo directions as images in chat (Lesson 2.1). Pick one.
- Ask it to create your brandbook from the template using your pick (Lesson 2.2). Read it, edit, then approve.
- Ask it to run the cover generator once with a test title and attach the images (Lesson 2.4). Check they match the brandbook.
- Ask it to add the brandbook to the brand brief and to every content bot's instructions, so every bot finds it.
- Optional: if you want an avatar, see the bonus lesson 2.3.

> You approve
> The logo, the brandbook, and every later change to colours or type. If you do the bonus, the avatar and its first set of poses too.

Common mistakes: a complex logo. Bots render images in code and in generated scenes. A plain wordmark survives that; a detailed illustration gets redrawn wrong every time.

LESSON

2.1 Three logo directions, pick the simplest

What it does

Your content (CMO) bot designs three logo directions as plain HTML and CSS, renders them to images, and sends you one sheet with all three side by side in chat. You pick one. Because the logo is made of type and a simple shape, every other bot can reproduce it exactly, in code, at any size.

Steps

- Give the bot a short brief in chat: your name or brand name, three words for the feeling (mine: calm, honest, warm), and one accent colour you like.
- Ask for three directions, each made with one free font, and a contact sheet image attached in chat showing all three on dark and light backgrounds and at small sizes (profile picture, favicon).
- Ask it to explain each direction in two lines: what it says, and where it might struggle.
- Pick the one that still reads clearly at 32 pixels. That is usually the simplest.
- Ask for the final files, attached in chat so you have your own copy: the full wordmark, a short version for small spaces, dark and light versions, and an icon. The bot keeps the source files in its workspace.
- Write the logo rules into the brandbook (Lesson 2.2).

Here's how I did it

My first logo was a plain wordmark that a bot drew in code from three type directions, exactly as in the steps above. My current logo kit replaced it. It is a small mark (three steps down, a turn, three growing suns) next to my name:

- Versions: a horizontal lockup (the main one), a stacked version for square spaces, the name on its own when the icon is already nearby, and the icon alone for avatars and the favicon.
- Rules: clear space on every side equal to the largest sun dot; minimum sizes for each version; the dark version on dark backgrounds and the light version on light ones.
- Don'ts: don't recolour it, stretch it, slant it, retype it, rearrange its parts, or add glow, shadow or blur.

Because the mark is artwork rather than plain type, my bots never redraw it. Every cover, slide and page uses the official files from the kit, unchanged.

What to adapt: a business name works the same way as a personal name. Pick the one direction that stays readable small, and keep its rules strict.

> You approve
> The final direction, the short version and the icon. Also any later variation (a favicon, a banner version) before it goes live.

Common mistakes: picking the direction that looks best on a big screen. Look at the contact sheet at profile-picture size, on a phone. If you squint and can't read it, pick another.

LESSON

2.2 The brandbook as the source of truth for every bot

What it does

The brandbook is a short document that every bot that makes something visible reads first. It turns your taste into rules a bot can follow: exact colours, exact fonts, layout patterns, voice, and a clear "never" list. Mine is 14 short sections and fits on a few pages. Yours can be shorter.

The sections I use

- Who we are: two or three sentences, mission, values, what you offer (and what you don't).
- Logo: wordmark and icon (Lesson 2.1).
- Logo rules: do and don't.
- Colour: each colour with a name, a hex code and its job. Mine: Onyx #020202 (main background), Ivory #F2F2F2 (text), Amber #FFB703 (main accent, one focal point), Solar #FB5607 (small highlights and the sunrise gradient). Plus one rule: "Mostly dark, one Amber focal point per screen, and text on Amber or Solar is Onyx."
- Typography: Geist Semibold for headings and Geist Regular for body text at 1.6 line height; Geist Mono for small uppercase labels; one Instrument Serif Italic accent word per heading; Unbounded Bold only for big numbers and short statements.
- Visual style: "A fine Ivory dot grid on Onyx, bent by one soft fold, behind every post. Big type in a quiet part of the field. The logo as a small signature, never the hero."
- Voice: "Calm, honest, warm, proof-first." Example lines it sounds like, and a "never" list.
- Content pillars: the few topics you post about.
- Story rules: which personal facts may be used and how.
- Channels: where you live online and what each one is for.
- Per-channel rules (mine has an Instagram page).
- Free things (optional): what you give away and how.
- Offers (optional, if you have any): short, pointing to the brand brief for facts.
- Affiliate links: how to label them.

Steps

- Attach or paste `files/02-brand/02-brandbook-template.md` into your content bot's chat.
- Ask it to create brandbook.md from the template, filled in from your logo pick and your brand brief, and to show it to you in chat.
- Check the colour section yourself. Every colour needs a hex code and a job. "One focal point per screen" is the kind of rule that keeps a feed calm.
- Write the "never" lists yourself. Bots can't know what you'd hate.

- Ask the bot to render the brandbook as a PDF and attach it, so you can read it as a document. The Markdown version in its workspace stays the source bots read.
- Ask it to add the brandbook to every content bot's instructions (attach it to any bot outside the shared workspace).

Here's how I did it

My brandbook has a "Visual style" rule that does a lot of quiet work: "No stock 'robot' imagery, no fake screenshots, no money piles." It also says AI-generated scenes are fine as mood shots, never shown as my real past.

The brandbook is versioned. When my positioning changed, the bot produced a new version and kept the old one for reference. My current version added one required element: the dot-grid background that every post and the website sit on, so new posts match the site.

> You approve
> The whole brandbook, and every version after. The "never" lists and the story rules especially.

Common mistakes: colour names without hex codes ("warm yellow"). Every bot will pick a slightly different yellow. Write the code.

LESSON

2.3 Bonus: avatar (optional)

> Optional bonus. Not everyone wants an avatar, and nothing else in this free guide depends on one. Skip this lesson if you'll use your own photos, your logo, or no face at all. It's here as a plus, with my example.

What it does

An avatar is an approved image of a character that appears in your content: pointing at a headline, holding a report, looking surprised. A pose library is a set of these images, each with a transparent cut-out version, so bots can drop the right pose into a carousel or a thumbnail without generating anything new each time.

What the avatar looks like is your choice: a character, a stylised look, or your own likeness. Any of them can work. Whatever you choose, check each platform's current rules on labelling AI-generated images and video, and follow them.

Steps

- Decide the look. A character, a stylised version of you, or your own likeness. Write two lines on the style you want (for example, painterly illustration, warm light, your brand colours in the background).
- Make the base image once. Ask the bot that owns image credits to generate a few drafts with an image tool that accepts reference images, and to attach them in chat. Give it reference images if you want a likeness. Pick one. Expect several rounds of small fixes.
- Write the identity paragraph. Once you approve the base, ask the bot to write one paragraph describing exactly what must stay the same in every pose: hair, glasses, clothing, accessories, expression range. You check it. This paragraph goes at the start of every pose prompt.
- Use references every time. The bot attaches the approved base image (and a close crop of the face) to every generation. Text alone drifts.
- Generate poses in small batches. Each pose: one expression or action, plain background, a full version and a transparent cut-out. Ask the bot to attach them in chat for review.
- Review each one against the base. Reject anything that drifts.
- Keep a request file. Other bots ask for poses by adding a row to a shared request file. One bot spends the credits.

What I did

I use Higgsfield, an image and video generation tool: higgsfield.ai (affiliate link: I earn a commission if you sign up, at no cost to you). Any image tool that accepts reference images can work.

My avatar is a painterly digital illustration with warm lighting, on my brand's dark background with one accent glow top right. The base needed a few rounds of fixes (glasses, beard) before it looked right. One pose was later rejected for the wrong glasses and retried with a stronger identity paragraph and a face crop.

My pose library holds poses like laptop, thinking, arms crossed, pointing, clipboard, surprised, laughing and facepalm, plus props for specific posts. Each pose has a `_transparent` version for carousels. The shared pose prompt starts with the same identity paragraph every time, then one line for the pose. My rules: the only brand mark allowed in props is my logo, and the palette is fixed to the brandbook colours. When I asked for more varied poses, my CMO bot added rows to the request file, the Short Video Creator generated them, and each row was marked `done`.

The template I use is in `files/02-brand/02-bonus-avatar-pose-prompts.md`.

> You approve
> The base avatar (it represents you), the identity paragraph, and each credit spend unless you gave a standing permission for a monthly credit plan.

Common mistakes: letting bots generate a new avatar per post. Faces drift. Generate a library once, reuse it, and add poses only when a post truly needs one.

LESSON

2.4 The cover generator: every cover image in one style

What it does

Most brands need a lot of cover images: blog post headers, event announcements, episode thumbnails, product or service images. Instead of designing each by hand or generating each with an image model, I use one small Python script. It takes a title and a subtitle, and draws the background, the headline, the accent dot and the wordmark the same way every time, in a wide (1280x720) and a square (600x600) version.

You don't run it yourself. You ask your bot to run it and attach the images. Because it is code, any bot that can run Python in its workspace can use it, and every cover matches the brandbook exactly.

What the script does

- Draws the dark background with a soft accent glow from the top right.
- Writes the title in big, tight, bold type, shrinking it until it fits (3 lines on the wide cover, 4 on the square).
- Puts an accent-coloured dot after the last word of the title, echoing the logo.
- Adds the subtitle in muted grey.
- Draws the wordmark bottom left and your domain bottom right.
- Saves both sizes as PNG files.

Steps

- Attach the files from `files/02-brand/02-cover-generator/` (the script, the README and the CSV) in your content bot's chat.
- Ask it: "Fill in the CONFIG section from my brandbook (background and accent colours, fonts, wordmark, domain), run the script with a test title, and attach both images."
- Look at both images in chat. Ask for changes until they match the brandbook.
- Tell the bot: "Every new cover comes from this script. Titles must match the brand brief exactly."
- For several covers at once, ask it to fill the CSV and run the batch mode, then attach the results.

If your bot can't run code, you can run the script on your own computer instead (it needs Python 3 and Pillow; the README explains how). Only do that if you're comfortable with it; the bot route is the main one.

What to adapt: change the colours, fonts and wordmark to yours. A café could use it for weekly specials, a coach for workshop dates, a podcaster for episode covers.

Here's how I did it

My covers come from a script built the same way. A first version made covers for several ideas in one batch, plus a contact sheet to compare them side by side. A later version made one cover at a time, so a bot could add a cover as part of the same run that created the post or product.

When some older covers didn't match the style, a bot regenerated them and kept the old files with an `-old` suffix, so nothing was lost.

> You approve
> The cover style once (the first test run), then each cover as part of approving the post or listing it belongs to. Covers are public, so they go through the same yes as the text.

Common mistakes: putting claims on the cover ("Save 10 hours a week"). The cover is copy too, and it gets the claim check like everything else. Keep it to the title and a plain description.

MODULE 3

Research engine: never run out of fresh ideas

A daily trend feed for your niche, post ideas for each channel, and a weekly trend brief.

You'll have: one research bot, one shared feed, sources tagged by how sure the bot is.

4 lessons · 4 files

LESSON

Module 3 · Research engine: never run out of fresh ideas

In this module: 4 lessons, 4 files. Reading time about 20 minutes.

Why a research bot

The hardest part of posting every day is not writing. It is knowing what to write about. If each content bot searches the web on its own, you pay for the same search five times and get five slightly different versions of the same story.

So one bot does the research, once a day, and writes it to one shared file. Every content bot reads that file. The research bot tags how sure it is about each item, links the source, and ends with a short list of post ideas. Content bots then turn those ideas into drafts for their own channel.

This works for any niche. My niche is AI experiments: I run them and write down what worked, and what didn't. So my feed covers AI news and tool releases. A fitness coach's feed might cover new research and common client questions. A local bakery's might cover local events, seasonal ingredients and food trends. An illustrator's might cover art challenges, techniques and what other artists are posting.

What you get

- A daily trend feed (Lesson 3.1): news in your niche, new tools or products, things people are talking about, people worth studying, and 5 post ideas.
- Post ideas per channel (Lesson 3.2): how each content bot uses the same feed differently.
- A weekly trend brief (Lesson 3.3): the phrasings and pain points your audience uses, for planning a week of posts on any channel.

What stays human

Choosing. The feed suggests; you approve the drafts that come out of it. And checking: anything the feed marks as unverified gets checked against its source before a bot uses it in a draft. The tags make that easy.

The lessons

- 3.1 The daily trend feed routine.
- 3.2 Turning the feed into post ideas for each channel.
- 3.3 The weekly trend brief.

Files in this module

- `files/03-research/03-role-research.md`
- `files/03-research/03-routine-daily-trend-feed.md`
- `files/03-research/03-routine-weekly-trends.md`
- `files/03-research/03-feed-example.md` (a fictional, clearly labelled example of the format)

Steps for the whole module

- Set up a Research bot: paste `03-role-research.md` in its chat (placeholders filled in) and ask it to use it as its instructions.
- Ask it to create the daily feed routine from `03-routine-daily-trend-feed.md`, at a time an hour or two before your content bots start.
- Ask your lead bot to point every content bot at the feed (research/latest.md). Bots outside the shared workspace get the feed sent or attached.
- After a week, ask the research bot to show you a few feeds in chat. Tell it what to drop and what to find more of.
- Add the weekly brief when you start planning a week of posts at a time.

> You approve
> The research bot's topic list and sources once. After that, it needs no daily approval, because it publishes nothing. Your yes comes later, on the drafts built from it.

Common mistakes: asking the research bot to also write the posts. Keep the jobs apart. A researcher that writes posts starts choosing stories for how easy they are to post, not for how true or useful they are.

LESSON

3.1 The daily trend feed routine

What it does

Every morning, the Research bot builds one feed file: what happened in your niche recently, which tools or products are new, what people are talking about, who is worth studying, and five post ideas grounded in those items. It saves the file as `latest.md` (and keeps older days), then sends the lead bot the top three ideas.

The structure of each feed

- Header: when it was updated, how bots should use it, who it is for, and the tag legend.
- News in your niche (last 24 to 48 hours).
- New tools, products or releases your audience would care about.
- What people are talking about: trending discussions, questions, repos, hashtags, whatever fits your niche.
- People worth studying: creators or businesses in your space and what they do well (never copies of their posts).
- Top 5 post ideas for today, each with a channel, a hook, an angle, sources, and a "check first" line when needed.

Each item has what happened, why it matters, an angle, and source links. Adapt the sections: a local business might swap "new tools" for "local events this week".

The tags that make it safe

Every item carries one tag:

- [VERIFIED]: the bot opened a primary or major-outlet source in this run.
- [SECONDARY]: trade or aggregator source only.
- [SOCIAL-SUMMARY]: only a social media summary, which may contain errors; check before posting.
- [UNVERIFIED]: shaky or not confirmed.

Content bots treat anything below VERIFIED as "check before you draft".

Steps

- Attach `files/03-research/03-routine-daily-trend-feed.md` in your Research bot's chat and ask it to create the routine from it.
- Tell it your niche, your channels, and the time it should run.
- Set it "quiet toward you": it messages you only if it fails. It always pings the lead bot.
- Ask for a "new compared with yesterday" rule, so a story already covered only reappears as an UPDATE.
- Ask it to add the routine to its routines file and show you the entry.
- Ask it to attach the first few feeds in chat so you can read them.

Here's how I did it

My feed runs every morning, seven days a week, because I want fresh ideas daily. It covers AI news, tool releases, trending code repos and creators in my niche. The habit that makes it safe is the tagging: when a story only appeared in a social summary, the feed says so on that line, and no content bot uses it until someone opens the real source. When a story is still developing, the feed tells the other bots what not to claim yet.

See `files/03-research/03-feed-example.md` for a fictional example of the format.

> You approve
> Nothing daily. Read the first week's feeds yourself and approve the topic list and the source rules.

Common mistakes: no "new since yesterday" rule. Without it the feed repeats the biggest story for days, and your channels repeat it too.

LESSON

3.2 Turning the feed into post ideas for each channel

What it does

One feed, many channels. The research bot ends each feed with five post ideas, each one already tied to a channel and a format. Then each content bot reads the feed and picks what fits its own channel, rewriting for that audience. An X bot wants a sharp take. An Instagram bot wants a step-by-step carousel. A video bot wants something it can show on screen.

The shape of a good post idea

Each idea has the same parts:

- Channel and format: "Instagram Reel + X post", "X thread", "Instagram carousel", "blog section".
- Hook: the first line, written out.
- Angle: what the post explains, and the honest limit.
- Sources: the links behind it.
- Check first: what must be true before anyone drafts it ("try the tool before claiming any result", "confirm the date on the organiser's page").

That last line is the important one. It stops a bot from posting "this new thing does X" before anyone has tried it.

Steps

- Ask your research bot to end each feed with five ideas, each with channel, hook, angle, sources and "check first".
- Ask for a few backup ideas too, for days when the top five don't fit.
- Ask your lead bot to add this to each content bot's instructions: "Read latest.md. Pick ideas that fit your channel. Do the 'check first' step before drafting. Note which feed item each draft came from."
- Ask the lead bot to make sure two channels don't post the same idea on the same day unless the calendar says so.
- When a draft comes to you, look for the source line. No source, no yes.

Here's how I did it (fictional example of the shape)

This example is made up, to show the shape. Say the feed reports that a popular app changed how it asks for permissions. The idea might be:

- Format: X thread + Instagram carousel.
- Hook: "Your favourite app just changed what it can do without asking. Here's the one setting to check."
- Angle: what changed, and three plain rules for giving any tool access: least access, approvals for anything you can't undo, read-only where possible. No fear-mongering, no pitch.
- Sources: the company's own announcement plus one major outlet.
- Check first: open the announcement and confirm the setting exists before drafting.

The same idea fits each channel differently: a thread on X, a carousel on Instagram ("3 rules before you give an app access"), a short blog section. In my setup it also connects naturally to how my own team works (bots draft, I approve) without selling anything.

What to adapt: for a local business the "check first" line might be "call the venue to confirm the event"; for a coach, "find the actual study, not the headline".

> You approve
> Every draft built from the feed, as usual. The feed itself goes out nowhere.

Common mistakes: skipping "check first" because the source looks big. A major outlet can be wrong on a detail, and a new tool can fail when you try it. Check, then draft.

LESSON

3.3 The weekly trend brief

What it does

The daily feed is about what happened. The weekly brief is about what people in your niche are searching for and struggling with. Once a week, the Research bot gathers 15 to 20 topics, the exact phrasings people use, and the pain points behind them. It saves them to a file and sends the top 5 to the bot that plans your week of posts, with a short heads-up to you.

That can be your X bot (Module 6), your Instagram bot. An SEO bot can use the same phrasings for blog topics.

Daily feed vs weekly brief

	Daily trend feed	Weekly trend brief
Question it answers	What's new today?	What does my audience keep asking?
Main reader	Every content bot	The bot that plans the week (and SEO)
Output	News, releases, discussions, people, 5 ideas	15 to 20 topics, phrasings, pain points, top 5
Schedule	Every morning	Once a week, before the week is planned
Quiet when nothing new	Yes, toward you	No, it always delivers

Steps

- Attach `files/03-research/03-routine-weekly-trends.md` in your Research bot's chat and ask it to create the routine from it.
- Describe your niche in one line. Mine: "I run AI experiments and write them down." Yours might be "home workouts for busy parents" or "wedding photography in my city".
- Ask for phrasings in your audience's own words ("how do I...", "is there a way to..."), not marketing language.
- Schedule it the morning of the day your planning bot plans the next week.
- Ask the bot to keep each brief, so the planning bot can compare weeks, and to attach the first one in chat for you.
- After a few weeks, ask the planning bot which topics turned into its strongest posts, and feed that back to the research bot.

Here's how I did it

I set mine up for my X bot, in my niche, with the same format as above: 15 to 20 topics, phrasings, pain points, and a ranked top 5 sent to the X bot plus a short heads-up to me. The routine always delivers, even in a quiet week, because the planning bot relies on it.

> You approve
> The niche line and the topic boundaries once. Then, as always, the posts your bots draft from the brief.

Common mistakes: letting the brief chase whatever is trending everywhere. Tie it to your niche and your audience's problems, or your posts drift away from what you actually offer and teach.

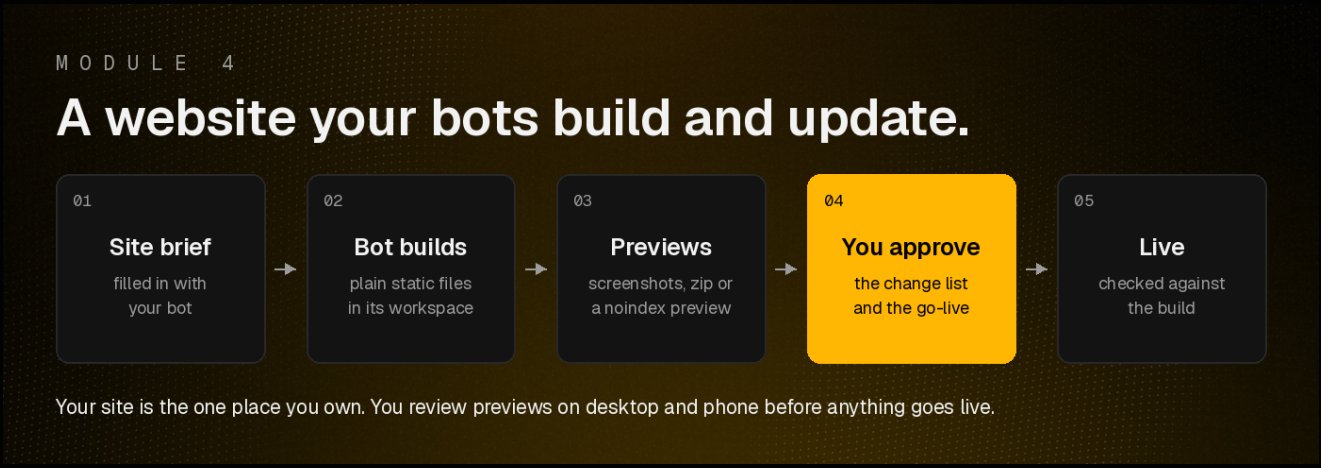
MODULE 4

A website your bots build and update

A site brief, the bot's build, your review of the previews, two ways to get it online, SEO and AEO basics, optional scheduled publishing, and contact forms without a backend.

You'll have: a static site you review on your phone before anything goes live.

6 lessons · 17 files



The workflow from Module 4.

LESSON

Module 4 · A website your bots build and update

In this module: 6 lessons, 17 files. Reading time about 35 minutes.

Why your own site

Social platforms change their rules and their reach. Your site is the one place you own. It works for any brand: a portfolio, a local business page with opening hours and a contact form, a coach's booking page, a creator's home for their best work. In my setup every channel points back to the site.

The good news: a fast, clean site does not need a database, a page builder or a server. Mine is plain static files (HTML, CSS, a little JavaScript, images). Bots are very good at writing static files, and static files are easy to check before they go live.

How you see the site while the bot builds it

The bot builds the site in its own workspace. You don't need to open that workspace. You see the work through what the bot sends you:

- Screenshots of every page at desktop and phone width, attached in chat.
- A zip of the site, attached in chat, if you want to open `index.html` on your own computer.
- A hidden preview address (once hosting is set up), so you can click around a real site on your phone before saying yes.

What you will build

- A site brief your content bot builds from, and a review pass where you check it (Lesson 4.1).
- Hosting: putting the site online, with my Cloudflare setup as the example (4.2).
- The SEO and AEO basics: sitemap, robots.txt, llms.txt, schema tags, blog structure (4.3). No ranking promises; these are the basics that let search engines and AI assistants read you correctly.
- Scheduled publishing (optional), so an approved blog post goes live on its date without you being there (4.4).
- A contact or enquiry form without a backend, using FormSubmit (4.5).

What stays human

- Creating the hosting account, buying or connecting the domain, and (if you want the bot to deploy) giving it access in a safe way. You do these.
- Picking the design direction from the options the bot shows you.
- Approving each page and each post before it goes live, and approving the go-live itself.

The lessons

- 4.1 Site brief → your bot builds → you review the previews
- 4.2 Hosting: putting the site online
- 4.3 SEO and AEO basics
- 4.4 Scheduled publishing: approved posts go live on their date (optional)
- 4.5 Contact forms without a backend (FormSubmit)

Files in this module

- `files/04-website/04-site-brief-template.md`
- `files/04-website/04-starter-site/` (a small, working static site with placeholder copy; give it to your bot as the base)
- `files/04-website/04-wrangler-example.jsonc` (Cloudflare config, for the bot)
- `files/04-website/04-stage-and-deploy-runbook.md` (the bot follows it)
- `files/04-website/04-seo-aeo-checklist.md`
- `files/04-website/04-routine-scheduled-deploy.md` (optional)

> You approve
> The site brief, the design direction, every page and post, and every change to your live site. A hidden preview needs no approval to build, only to go live.

Common mistakes: letting the bot "just fix it on the live site". Every change goes through a preview and a check, then goes live after your yes. That one habit is what makes it safe to let bots update your site at all.

LESSON

4.1 Site brief → your bot builds → you review the previews

What it does

You fill in a one-page site brief with your bot. Your content bot builds the site as static files in its workspace, then sends you screenshots on desktop and phone (and a zip, or a preview link, if you want one). You do one review pass, list changes, and approve. Only then does it go live.

Steps

- Fill in the brief. Attach `files/04-website/04-site-brief-template.md` in your content bot's chat and fill it in together. The key parts: who the site is for, the one action you want visitors to take (book, buy, enquire, subscribe, visit), the pages, the brandbook to follow, and the facts the bot may use (from your brand brief, never invented).
- Give it the starter site. Attach the files from `files/04-website/04-starter-site/` and say: "Use this as the base." It already has the structure, the SEO files and a contact form. The bot may also start from scratch if your brief needs something different.
- Ask for the build, kept separate from anything live.
- Ask for screenshots of every page at desktop and phone widths, attached in chat, plus a link check (every internal link works; every anchor exists).
- Review once, in a list. Number your changes. The bot applies them all and sends new screenshots.
- Want to click around? Ask for a zip of the site and open `index.html` on your computer, or, once hosting is set up (Lesson 4.2), ask for a hidden preview address marked `noindex`, so search engines ignore it.
- Approve the go-live and ask the bot to log it in the approvals log.

Here's how I did it

My first homepage redesign went through exactly this. The team researched design directions and built options. I picked one: a dark page with a soft 3D shape. The bot put it on a hidden, noindex preview address not connected to my real domain. I reviewed it on my phone, picked a favicon from two options, and approved going live with one change: remove a section from the About page, because visitors already reach it elsewhere.

The bot made that one change, went live, checked the live files against the build, and kept a backup of the old site as a rollback. All of that went into one approvals log line.

The site has been redesigned again since, to match my current brandbook. Smaller pages followed the same path. My About page was drafted in first person from the brand brief's confirmed facts, approved, then put live.

> You approve
> The brief, the design direction, the preview, any change list, and the go-live. Each live bio or "about" change comes to you first.

Common mistakes: reviewing on a laptop only. Most visitors arrive from a phone, from a link in a social bio. Ask for phone screenshots every time, and open the preview on your own phone before you approve.

LESSON

4.2 Hosting: putting the site online

What it does

A static site needs a host: a service that serves your files at your domain. Many hosts do this well; pick one that serves static files, lets you connect your own domain, and keeps previous versions so you can roll back. I use Cloudflare Workers static assets, and this lesson uses it as the example. Plans, limits and pricing change, so at the time you read this, check your host's current pricing and docs pages before you rely on them.

Two ways to get the site online

Option A: you upload it yourself. The bot sends you the finished site as a zip in chat. Many static hosts let you upload a folder or zip in their dashboard; check your host's docs for how. This is the simplest route and needs no access for the bot. Every update means the bot sends a new zip and you upload it.

Option B: the bot deploys for you. You create the account and an access token that can deploy (and nothing more), and give it to the bot through your bot's secure way of sharing a secret (a private form or secret prompt), never in a normal chat message. If your bot has no secure way to receive a secret, stay with Option A. The bot then deploys from its workspace, checks the result, and tells you what it changed. This is what I do, and what Lesson 4.4 needs.

The config file (Option B, Cloudflare)

My whole hosting setup is one small config file the bot keeps in its workspace:

```
{
  "name": "{{SITE_NAME}}",
  "compatibility_date": "{{SETUP_DAY}}",
  "workers_dev": true,
  "preview_urls": false,
  "assets": {
    "directory": "{{SITE_FOLDER}}",
```

```
    "not_found_handling": "404-page"
  }
}
```

`not_found_handling: "404-page"` serves your `404.html` for any missing page. A copy with notes is in `files/04-website/04-wrangler-example.jsonc`; attach it for your bot.

Steps (Option B)

- You: create the hosting account, add your domain, and create an access token limited to deploying. Share it with the bot through its secure secret prompt. Never paste it into a chat.
- Bot: prepares the config file and the site in its workspace.
- Bot: deploys with the host's command-line tool (Wrangler, for Cloudflare), reading the token without ever printing it.
- Bot: checks the free preview address first and sends you the link and screenshots.
- You: connect your domain to the site in your host's dashboard. This is a one-time step.
- Bot: after every deploy, checks that each live file matches the build, and records the version ID so it can roll back.

Here's how I did it

My deploy instructions have three safety lines about the token, and every deploy routine copies them: never print the token, never show the contents of the file it lives in, never dump the environment.

My site's `_headers` file (in the starter site) sets a few things Cloudflare reads at deploy: correct content types for `llms.txt` and `robots.txt`, and a `noindex` header on the free preview addresses, so test addresses stay out of search.

Every deploy is recorded with the version that was live before it. When my redesign went live, the log line included the new version and the previous one as the rollback target.

> You approve
> Creating the account and the token (you do these), connecting the domain, and every change to the live site.

Common mistakes: putting the token in the config file or a script that gets shared. The token goes to the bot only through a secure secret prompt, and is never written into a file you'd send anywhere.

LESSON

4.3 SEO and AEO basics

What it does

SEO (search engine optimisation) helps search engines understand your pages. AEO (answer engine optimisation) helps AI assistants such as Claude, Perplexity and Gemini find, read and cite you. I make no ranking promises. What follows is the set of basics that make your site readable and correct. They are cheap to do and bots do them well. They apply to any brand site, from a portfolio to a local shop.

The basics

- sitemap.xml: every public page with a `lastmod` date. The bot updates the date when a page changes.
- robots.txt: who may crawl. The starter version allows search engines, AI search crawlers, live-fetch agents and training crawlers (you choose), disallows `/api/` and `/cdn-cgi/`, and points to the sitemap. robots.txt states a preference only; enforcement lives in your host's settings.
- llms.txt: a plain-text map of the site for AI assistants: one-line summary, "Start here" links, blog posts, free things, offers (if you have any), contact. Mine opens with my approved positioning line.
- Schema tags (JSON-LD): a `Person` (or `Organization` / `LocalBusiness`, if that fits you better) entry on every page with the same name and description, so every page tells the same story. Blog posts get `BlogPosting`.
- Blog structure: `/blog/` index, one folder per post (`/blog/<slug>/index.html`), and an author box on each post.
- An About page and a plain facts list: who you are, what you offer, where to find you, how to get in touch. Assistants love plain facts. A local business adds address and opening hours.
- Clean basics: one H1 per page, a real title and meta description, Open Graph image, `noindex` on templates and test addresses.

Steps

- Attach `files/04-website/04-seo-aeo-checklist.md` in your content (or SEO) bot's chat. The starter site has all seven in place with placeholders.
- Ask the bot to fill in titles, descriptions and the schema from your brand brief, word for word, and to show you the list of titles and descriptions in chat.
- Ask it to keep `sitemap.xml`, `llms.txt` and the blog index updated in the same change that adds a post.
- Connect Google Search Console yourself (it needs your sign-in). If your bot has a way to read it (a connector you set up), ask it for a weekly read-only summary; otherwise check it yourself now and then and paste what you see.

Here's how I did it

When I approved a new positioning line, the bot put it into the schema on every page, the About page and `llms.txt`. A later approval aligned the schema on the remaining pages and updated their `lastmod` dates. Small, exact changes, each approved.

> You approve
> The positioning text in the schema and `llms.txt`, page titles and descriptions, and the crawler policy in `robots.txt`.

Common mistakes: different descriptions of you on different pages. Pick one line, approve it, and use it everywhere.

LESSON

4.4 Scheduled publishing: approved posts go live on their date (optional)

What it does

You approve a blog post today. It should go live next Tuesday morning, whether or not you are at your desk. A one-off routine publishes it at that time, but only if everything still matches what you approved. If anything has changed, it stops and tells you instead of publishing.

This lesson is optional. It only works if your bot deploys the site for you (Lesson 4.2, Option B). If you upload the site yourself, skip it: ask the bot to send you the updated zip on the day, and upload it.

How mine works

All of this runs in the bot's workspace; you see the result in chat. Each scheduled post has its publish date in its file. The build only includes posts whose date has arrived. On top of that, three checks:

- Live check. Before building, the bot compares its copy of the site with the live site. If anything differs, someone changed the live site, or a previous publish didn't sync. Stop.
- Approved inputs. At approval time, the bot saves a checksum list of everything the post needs: the post files, images, page copy. On the day, it checks them again. If anything changed since your yes, stop.
- Expected output. For each date there is a list of exactly which files should change. If the build produces anything different, stop.

Only when all three pass does it publish. After that, it checks every changed file is live and identical, that posts not due yet are still hidden, and that every link and image on the changed pages works.

Steps

- Attach `files/04-website/04-stage-and-deploy-runbook.md` (the procedure) and `04-routine-scheduled-deploy.md` (the routine) in your content bot's chat.
- When you approve a post, ask the bot to set its date, run a test build for that date, and save the expected list and the checksums.
- Ask it to create a one-off routine for the date, with "deletes itself after running", and to show you the routines-file entry.
- Make sure the stop rules are in the routine word for word: never work around a failed check by editing files, regenerating posts or skipping a step.
- Ask for a report after the run: published (with the link and checks passed), or stopped (with the step and the reason).

Here's how I did it

I approved two blog posts and a free landing page, with a standing yes to go live on their dates without another review. One routine is set to publish each on its date, then delete itself.

The checks earned their keep before the posts even went out. When my positioning line changed, the author box in both staged posts had to change too. The bot rebuilt them, saved new checksums and kept backups of the old ones. Without that, the publish run would have stopped, which is exactly what it should do.

> You approve
> Each post and its date. Optionally, a standing yes that approved posts go live on their date without a second look. Log it with the exact posts and dates.

Common mistakes: a publish routine that "fixes" a failed check. A check that fails means reality no longer matches your yes. The only correct move is to stop and ask.

LESSON

4.5 Contact forms without a backend (FormSubmit)

What it does

A static site has no server to receive form submissions. FormSubmit fills that gap: your form posts to FormSubmit's address with your email in it, and FormSubmit emails you each submission. No code, no database. Use it for enquiries, bookings requests, quote requests, or "work with me" forms. If your bot can read your inbox (through a connector you set up), it can then log each enquiry and draft a reply for your approval.

Steps

• Bot: adds the form. The starter site already has one on the home page: ``html <form action="https://formsubmit.co/{{YOUR_EMAIL}}" method="POST"> <input type="hidden" name="_subject" value="[{{TAG}}] New enquiry"> <input type="hidden" name="_captcha" value="true"> <input type="text" name="_honey" style="display:none"> <input type="hidden" name="_next" value="https://{{DOMAIN}}/confirmed/"> ...your fields... </form> ` - _subject puts a fixed tag in the email subject, so you (or your bot) can search for it. - _captcha: FormSubmit's captcha is on by default, and FormSubmit recommends leaving it on. true simply keeps the default. - _honey is a hidden "honeypot" field that spam bots fill and people don't; FormSubmit ignores any submission where it has a value. - _next sends people to your own thank-you page (/confirmed/, which the starter site marks noindex`) instead of FormSubmit's default page.

• You: once the site is live, submit the form once yourself. FormSubmit then emails you asking you to confirm the form; click it. Until you confirm, FormSubmit holds submissions instead of forwarding them. Only a person can do this step.

• You: add a filter in your mail app so these emails never land in spam.

• Bot (optional, needs inbox access you set up): a weekday routine that searches your inbox (and spam, separately) for the subject tag, ignores your own test submissions, logs each enquiry in a tracker and drafts a reply for your approval. Without inbox access, you forward enquiries to the bot yourself.

• Rule: every enquiry comes to you before any contact.

FormSubmit's features and limits can change, so at the time you read this, check its current docs.

Here's how I did it

My home page has a "work with me" enquiry form with a tagged subject. My business development bot has a weekday routine: it searches mail and spam separately for new tagged enquiries, rescues any from spam, ignores my tests, logs enquiries and prepares reply drafts for my approval. It is quiet when there is nothing new.

My site has a privacy page that says how form data is handled, and the site uses no analytics or tracking cookies.

> You approve
> The form fields and the privacy text, the first FormSubmit confirmation (you click it), the mail filter, and every reply to every enquiry.

Common mistakes: putting a personal email address in the form. It appears in your page source. Use a business address you are happy to show publicly. FormSubmit's documentation also describes a random-string form address it sends you when you confirm; using that in `action` keeps your email out of the page source.

MODULE 5

Instagram, published for you (after your yes)

A daily drafts routine, carousels and Stories in your brand style, publishing through the API, comment-to-DM with ManyChat, and engagement batches.

You'll have: tomorrow's set in one message, scheduled only after you reply.

7 lessons · 11 files

LESSON

Module 5 · Instagram, published for you (after your yes)

In this module: 7 lessons, 11 files. Reading time about 40 minutes.

What this module covers

This module turns Instagram into one calm daily routine for any brand: a bakery, a coach, an illustrator, a consultant, a shop. Every evening your content bot drafts the next day's full set (the feed post, its Stories, captions, alt text, and the DM replies for anyone who asks for a link), renders every image in your brand style, and sends it all in one message. You read it and say yes, or list changes. After your yes, the bot publishes each item at its time through Instagram's publishing API and checks that each post went live exactly as approved.

That is what "automated" means here: the production and the publishing are done for you, and your yes sits between them.

What you will build

- The daily drafts routine (Lesson 5.1).
- Carousel and Story rendering from HTML templates in your brand style (5.2).
- Publishing through the API: image hosting, containers, and one-off publish routines that never double-post (5.3).
- The "API-publishable only" rule: no manual steps hidden in your drafts (5.4).
- Comment-to-DM with ManyChat, so links reach people where they can tap them (5.5).
- Engagement: comment and reply batches for quick approval (5.6).

What to adapt

- What you post. A local business might post this week's offer, a behind-the-scenes photo, a customer question answered. A coach might post one tip a day. An artist might post work in progress. The routine is the same; the content plan is yours.
- What you hand out by DM. A free guide, a menu, a booking page, a price list, a sign-up page. Anything you would otherwise put behind "link in bio".
- Your pace. Choose one you can approve calmly every day, not the most a bot can produce.

What stays human

- Creating the Instagram account, switching it to a professional (creator or business) account, and connecting it to the publishing connector. These need your sign-in.
- Setting up ManyChat and connecting it to Instagram.
- Approving every day's set, and every comment or DM reply.
- Deciding whether bots may post outbound comments on other people's posts at all. In my setup they don't: I post those myself from a list the bots prepare (Lesson 5.6).

How much to post

I started with plans for more and cut back to one feed post a day plus Stories. My reasoning: while an account is small, posting a lot does little if few people are watching, and every extra post is one more thing to approve. Pick a pace you can keep up with your yes.

The lessons

- 5.1 The daily drafts routine: tomorrow's set in one message
- 5.2 Rendering carousels and Stories in the brand style
- 5.3 Publishing through the API
- 5.4 "API-publishable only": no manual steps in your drafts
- 5.5 Comment-to-DM with ManyChat: links only by DM
- 5.6 Engagement: comment batches for quick approval

Files in this module

Attach or paste these in your content bot's chat; it keeps its own copies in its workspace.

- `files/05-instagram/05-routine-daily-ig-drafts.md`
- `files/05-instagram/05-publish-notes-template.md`
- `files/05-instagram/05-carousel-template/` (working HTML/CSS slides plus a render script your bot runs)
- `files/05-instagram/05-caption-rules.md`
- `files/05-instagram/05-manychat-flow.md`

> You approve
> Every day's set, before anything is published. Every reply to a comment or DM. Every outbound comment (or you post those yourself).

Common mistakes: approving item by item through the day. One message, one yes, at one time each evening. It is faster for you and safer for the bots.

LESSON

5.1 The daily drafts routine: tomorrow's set in one message

What it does

Every evening your content bot builds the next day's complete Instagram set and sends it to you in one message. You approve the whole set together. Nothing is posted without that yes.

What goes into the set

The evening before, the bot:

- Reads the brand brief, the research feed (`research/latest.md`, checking anything marked unverified), and the approved content calendar if one exists.
- Drafts the next day's feed post (or posts), a Reel when one is scheduled, and the Stories.
- Renders the images in your brand style (Lesson 5.2).
- Writes captions, alt text and the DM replies for the comment keyword (Lesson 5.5).
- Makes sure everything is publishable through the API with no manual step (Lesson 5.4).
- Sends everything in one message, and adds one self-review line to its self-review file (Lesson 9.2).

Steps

- Attach `files/05-instagram/05-routine-daily-ig-drafts.md` in your content bot's chat and say: "Create this routine for me, fill in the placeholders from my brand brief, add Part A to your routines file, and show me both."
- Pick a time when you can review calmly. Mine is in the early evening, daily, because my approved schedule posts seven days a week. If you post three days a week, run it three evenings a week.
- Set "quiet when nothing new" to no: it always delivers drafts, or a one-line note if nothing is scheduled.
- Ask the bot to keep one folder per day in its workspace, with `drafts.md` (all text), the rendered images, and a contact sheet that shows every slide at a glance. It attaches the contact sheet and pastes the text in its message, so you never have to go looking for files.
- When you reply, be specific: "Yes to all", or "Yes, but slide 4: change X". The bot logs your yes in the approvals log, then creates the publish routines (Lesson 5.3).

What to adapt

The set doesn't have to be a carousel. A café's set might be one photo post with a short caption and two Stories. A coach's might be one carousel tip. Tell the bot what a normal day looks like for your brand, and let the routine fill that shape.

Here's how I did it

My `drafts.md` is laid out per post:

- Slides: each slide's headline and body text, and which image sits where.
- Caption: the exact caption, ready to publish.
- Alt text: one line per slide, describing what is on it.
- Keyword and DM variants: for example "REPORT", with three DM versions that rotate.
- Source: which product or help pages the slides were checked against.

Then a section for the Stories, and a closing section of notes and open items, such as "The link in the DMs is still [LINK]; the post can go out without it, but the DM flow needs a live link." That kind of note is the bot being honest about what isn't finished.

When I approve a weekly content calendar, the bot files it with the shared files and updates the routine to follow it, so the daily drafts match the plan without me repeating it.

> You approve
> The whole set, every evening: slides, captions, alt text, Stories and DM replies. Changes go back as a numbered list.

Common mistakes: approving from the text only. Open the contact sheet. A slide can read well in `drafts.md` and still have an image covering the headline.

LESSON

5.2 Rendering carousels and Stories in the brand style

What it does

My carousels are not designed in a drag-and-drop tool. Each slide is a small HTML page styled with the brandbook's colours and fonts, and a headless browser takes a screenshot of it at exactly 1080 × 1350 pixels (the 4:5 portrait size for Instagram feed posts). Stories use the same method at 1080 × 1920. Because the design is code, every slide matches the brandbook every time, and the bot can fix a typo by changing one line and rendering again.

You don't need to touch the code. Your bot runs the template and attaches the images.

How a slide is built

- Background: your brand's base colour, with a soft accent glow if your brandbook uses one.
- Logo top left, drawn in CSS (as in Lesson 2.1) or as an image.
- Kicker: a small pill above the headline ("Step 1", "This week's offer").
- Headline: big, tight, bold, with one highlighted phrase in the accent colour.
- Body: a card with numbered rows, a small table or pills.
- Footer: your handle bottom left, a slide counter bottom right.
- Optional image: a transparent cut-out bottom right on the cover and the last slide. That can be a product photo, your logo mark, a photo of you, or an avatar pose if you made one in the bonus lesson (2.3). Slides work fine without it.

Steps

- Attach the files in `files/05-instagram/05-carousel-template/` in your content bot's chat. They are a shared stylesheet, three example slides (cover, step, call to action) and a render script.
- Ask: "Fill in slide.css from my brandbook, use my handle, render the three example slides, and attach the images and the contact sheet." The script renders each slide with headless Chrome, converts it to JPEG (Instagram's API takes JPEG) and builds the contact sheet.
- Review the contact sheet first: all slides shrunk side by side in one image. Look at it at phone size.
- Ask for changes in plain words ("bigger headline", "less text on slide 2") until the template feels like your brand. Approve the template once.
- For Stories, ask the bot to use the Story size (1080 × 1920) and keep all text between about 250 px from the top and 340 px from the bottom, so Instagram's own buttons don't cover it.
- From then on, the daily drafts routine (Lesson 5.1) uses this template for every post.

If you'd rather run the render script on your own computer, you can: the template's README lists what it needs (Chrome or Chromium, and Python with Pillow for the JPEGs and contact sheet). The bot route does the same thing.

What to adapt

A photographer or a bakery may post photos rather than text slides. Keep the template for the text-heavy posts (tips, prices, opening hours, a how-to), and let real photos carry the rest.

Here's how I did it

My first carousel taught one free tool in eight slides: a cover, a setup slide with three numbered steps, a slide of messy example notes, a slide of category pills, a small table, the one-page result, and a call-to-action slide with the comment keyword. Every slide was rendered from HTML, checked on a contact sheet, converted to JPEG, and after publishing compared with the live post, slide by slide, in order.

My Stories contact sheet draws red safe-zone lines so I can see the margins. The lines exist only on the sheet, never on the frames.

> You approve
> The template once (colours, layout, footer), then each day's slides as part of the daily set.

Common mistakes: text too small to read on a phone. My body text sits around 30 px on a 1080 px slide, and headlines between 80 and 108 px. Check the contact sheet at phone size before you say yes.

LESSON

5.3 Publishing through the API

What it does

After your yes, the bot publishes each post at its time through Instagram's official publishing API. In my setup it does this through a Composio connector that I connected once. The bot hosts the images, builds a container for each, publishes once, and checks the live post against your approved files. It has a dedupe gate, so a retry never posts twice.

The publishing flow

- Host the images. Instagram fetches images from a public URL. It must be a direct, public JPEG link. Signed links with query strings can fail.
- Build containers. For a single image: one container with the image URL and caption. For a carousel: one child container per slide (no caption on children), then a carousel container listing the children in slide order, with the caption.
- Wait until each container's status is `FINISHED`.
- Publish the parent container once. The response gives the published media ID.
- Verify the live post: media type, number of slides, exact caption with hashtags intact, permalink. Report to you.

Steps

- You: switch Instagram to a professional account and connect it to the publishing connector your bot uses. That is a sign-in only you can do.
- Attach `files/05-instagram/05-publish-notes-template.md` in your content bot's chat and say: "After I approve a day's set, fill in publish notes from this template for that day, and follow its run sequence for every post."
- Ask the bot to create, for each approved slot, a one-off routine that publishes only its own post, once, then deletes itself.
- Make sure the dedupe gate comes first in every run: (a) is there a state record saying this post is published? (b) does any recent post's caption start with this post's first line? (c) is the daily publishing quota OK? If any check says "already live", the run reports and stops.

- The run writes down the container IDs before publishing.
- If publish errors or times out, it doesn't call it again blindly. It re-runs the dedupe check first, because a publish can succeed even when the response fails.
- Ask the bot to send you the live link after each post, and to tell you at once if a run stopped.

Here's how I did it

On my first day of publishing, the image links were tested the evening before and worked. The next morning, every link for the first carousel returned an error. They had not survived the night. The routine's hosting check caught it, re-hosted the images, confirmed each one matched the local file by checksum, and published. It then warned the later runs that their links were probably dead too, and they re-hosted before building containers.

That is why my publish notes say: unless your image host documents how long links last, always check before use, and re-host if any check fails.

My notes also record smaller lessons: don't set a media type for image posts (it can trigger a "video required" error); containers expire after about a day, so create them just before publishing; and pass plain `#` in hashtags, then check the live caption.

> You approve
> The set (Lesson 5.1). The one-off routines only carry out that yes. If a run has to change anything you approved, it stops and asks.

Common mistakes: reusing yesterday's image links or containers. Re-check and re-host at publish time, every time.

LESSON

5.4 "API-publishable only": no manual steps in your drafts

What it does

This lesson is about one rule I added after my first day of publishing: everything the bot drafts for Instagram must be publishable through the API with no manual step for you. No in-app stickers, no reshares, no link stickers, no edits in the Instagram app.

Why

At the time you read this, check Instagram's current publishing API docs, but in my setup the API could post images, carousels, Reels and image Stories. It could not add the interactive extras you add by hand in the app: poll and quiz stickers, link stickers, or resharing a feed post into your Story.

My first Story plan included a link sticker. That meant someone had to open the app at the right time and add it by hand. If you are the "someone", the system is no longer doing the publishing for you; it is giving you chores at fixed times. So I changed the drafts instead of adding chores.

What to use instead

Instead of…	Use…
A quiz or poll sticker	A static quiz card baked into the image ("Quick quiz", options A/B/C), with the answer in the next Story
"This or that" sticker	Two static cards and "Reply to this story with your pick". Story replies arrive as DMs
Resharing a feed post	A static preview of the post's cover image inside the Story, with "New post on my profile"
A link sticker	"Comment KEYWORD on today's post and I'll DM you the link". Links go out only by DM (Lesson 5.5), never as a link or "link in bio" in captions or Story text

Steps

- Ask your content bot to add the rule to its instructions and to the daily drafts routine, word for word: "Everything you draft for Instagram must be publishable through the API with no manual step for me."
- Ask it to list, in each day's drafts, anything that would need a manual step. The list should be empty.
- Keep all Story text inside the safe zone (about 250 px from the top and 340 px from the bottom on a 1080 × 1920 frame).
- Keep links out of captions, Story text and Reels. Point to a keyword instead, and let the DM flow send the link (Lesson 5.5).

What to adapt

A local business might want a real poll sticker now and then ("Which new flavour?"), because the taps matter more than the convenience. That's a fine choice. Make it on purpose: add it to that day's set as a clearly marked manual step, and decide when you'll be at your phone to do it.

Here's how I did it

A typical Story day of mine has several slots, each a finished image Story: a warm-up quiz in the morning; the answer plus a preview of the morning's post; a "this or that"; and an evening frame that asks people to comment a keyword on the post to get the link by DM. My publish notes say "No stickers" for each one, and each is verified live as an image Story, with the live image matching the local file.

The trade-off is real: a static quiz is less interactive than a real sticker. I accepted that, because a sticker you have to remember to add at a fixed time is a sticker that sometimes doesn't happen.

> You approve
> The rule itself (it is a choice; you might prefer to add stickers by hand on some days). Then, as always, each day's set.

Common mistakes: a draft that says "add a poll sticker here" in small print. If it needs your thumb in the app, it is not API-publishable. Send it back, or approve it knowingly as a manual step.

LESSON

5.5 Comment-to-DM with ManyChat: links only by DM

What it does

Each post that offers something ends with a call to action: "Comment GUIDE and I'll DM you the free guide." When someone comments the keyword, ManyChat sends them a DM with the link automatically. You set up the flow once per keyword. The bot writes the DM text as part of the daily drafts, so you approve it with the post.

The rule, and why I recommend it

Links are handed out only by DM after a keyword comment: never a URL or "link in bio" in a caption, Story text, Reel voiceover or end card.

Why this is a good default for any brand:

- Links in Instagram captions aren't clickable. "Link in bio" sends people hunting, and many give up on the way.
- A DM puts the link exactly where they can tap it, and the conversation is started, so you can answer a follow-up question.
- One place to update. If a link changes, you change it in the flow, not in old captions.
- Fewer mistakes. The bot never has to remember which caption carries which URL, and the claim check covers one short DM instead of every caption.

It's your choice. If you'd rather use a link page in your bio, write that rule into your brand brief instead. Just pick one rule and let every bot follow it.

Steps

- You: create a ManyChat account, connect your Instagram professional account, and allow it to read comments and send DMs. These are sign-ins and permissions only you can give. If you don't have an account yet, this is my ManyChat affiliate link (I earn a commission if you sign up, at no cost to you).
- Pick one keyword per offer, short and in capitals: GUIDE, MENU, BOOK, PRICES. Use the same keyword every time that offer appears.
- Ask the bot for three DM variants per keyword, to rotate, so repeat commenters don't get an identical message. Each variant: a thanks, the link, one line of quick start, and one honest limit.
- Build the flow in ManyChat using `files/05-instagram/05-manychat-flow.md` (attach it in the bot's chat and ask it to fill in one block per keyword; then you build the flow from it): trigger on

comments containing the keyword on the chosen posts, send one of the variants, optionally reply publicly to the comment.

- Fill in the real link. Drafts use `[LINK]` until the page or download is live. The post can go out before the link exists, but the DM flow can't. Ask the bot to flag this in the notes every time.
- Test it yourself from a second account before the post goes live.

ManyChat's menus change; the template describes what to look for rather than exact button names. Check ManyChat's current help pages when you build the flow.

What to adapt

What you send by DM is up to you: a free guide, this week's menu, a booking page, a price list, a sign-up page. A local business might send its booking link and opening hours. An artist might send a print shop link.

Here's how I did it

My REPORT keyword sends a free tool of mine. One of its three variants:

> Hey! Here's your free Weekly Ops Report Skill: [LINK]
> Quick start: upload the zip in your AI app's skills settings, then paste your notes and say "Make my weekly ops report." There are example notes inside if you want to try it first.

Another variant carries the honest limit: "Anything it can't verify goes under 'Missing data' instead of being made up, so check that bit before you share the report."

The DM for an email-sorting tool says plainly what it doesn't do: "it only reads and sorts. No labels, archiving or sending unless you ask, and replies are saved as drafts for you to check." That is the claim check (Lesson 1.5) applied to a DM.

The last slide of each carousel shows the keyword in large text with a simple chat-bubble icon, which makes the "comment and I'll DM you" idea visual without showing any real app screen.

> You approve
> Each keyword's DM variants (with the daily set), the live link, and the ManyChat flow before it is switched on.

Common mistakes: promising more in the DM than the offer delivers ("this will save you hours"). Run the DM text through the claim check like any other copy.

LESSON

5.6 Engagement: comment batches for quick approval

What it does

Posting alone gets little attention on a new account. Commenting on other people's posts and replying to your own comments and DMs matter. A bot can find the posts and draft the words. In my setup, every comment and reply still goes through approval, as one daily batch.

Two kinds of engagement, two decisions

- Inbound replies (comments and DMs on your own posts): the bot drafts a reply to each, sends them as one numbered batch, and posts each reply only after your yes.
- Outbound comments (on other people's posts): decide whether a bot may post these at all. In my setup, it doesn't. The engagement bot gives me a list of posts plus the exact comment for each, and I post them by hand. I tried bot-posted outbound comments first, looked at the first batch, and decided I wanted these in my own hands. That's a good example of a person setting the line and the system adapting to it.

Steps

- Give engagement to one role, so it doesn't collide with the content bot's drafts. I call mine Engagement. On a small team, your content bot can do it as a separate daily job.
- Ask the bot to create a weekday routine that builds one numbered batch: outbound comments with the exact text and a link to the post; inbound replies with the original comment and the draft reply.
- Set it quiet when there's nothing new.
- Answer by number ("1–6 yes, 7 no, 8 change to..."). The bot posts the approved inbound replies, logs the batch in the approvals log, and leaves the outbound ones for you if that is your rule.
- Optional: once a week, ask the bot for a short note on what was sent and what came back. Keep the numbers real; a quiet week is reported as quiet.

Writing good comments (rules for the drafts)

- Add something: a detail, a question, a short experience. No "Great post!".
- Never pitch your product or service in someone else's comments.
- One comment per post, never several on the same account in a day.
- Match the language of the post.
- Never DM someone who hasn't contacted you first, unless you approve that exact message.

What to adapt

A local business might focus on replies to its own comments and on local accounts (neighbouring shops, local events). A coach or consultant might comment on posts from the people they want to work with. The batch format stays the same.

Here's how I did it

My Engagement bot runs on weekdays. Its own description sums it up: one numbered batch for me, with outbound comments as exact text that I post myself, and inbound comment and DM replies that it posts only after my yes. It posts nothing without that yes. The role prompt is in `files/09-running-the-team/09-role-prompts/role-engagement.md`.

> You approve
> Every comment and every reply, in one daily batch. You may decide, as I did, to post outbound comments yourself.

Common mistakes: sending you comments one at a time through the day. One batch, at one time, numbered.

MODULE 6

X: scheduled posts, replies, honest scoreboard

Weekly batches, reply drafts and when a standing permission makes sense, a post-score tracker script, and an optional weekly scoreboard.

You'll have: a week of posts approved in one go, and a simple score to learn from.

5 lessons · 5 files

LESSON

Module 6 · X: scheduled posts, replies, honest scoreboard

In this module: 5 lessons, 5 files. Reading time about 20 minutes.

What this module covers

X is where I post most often and where I learned the most about approvals. One bot, my X Channel Builder, runs the account. It drafts a week of posts at a time, posts only what I approved, drafts replies, scores every post, and can post a weekly progress update with real numbers, including the flat ones.

The method works for any brand that uses X: a freelancer sharing their work, a coach posting tips, a small business posting news. If X isn't your channel, the same pattern (weekly batch, one approval, a read-only queue check, an honest score) works for most text-first platforms.

The part people underestimate is the checking. My X bot spends more of its routines checking than posting: a claim check before I see a draft, a queue check twice a day before anything goes out, and a "don't post if the number can't be verified" rule on the scoreboard.

How it runs in my setup

Job	When	Who says yes
Weekly batch of posts	Drafted the week before, one file	You, once per batch
Posting the batch	One routine, one post per slot	Carries out your yes
Replies	Daily	Drafts for your yes; a standing permission only if you choose (Lesson 6.2)
Queue check	Twice a day, 72 hours ahead	Read-only, asks you only if something is wrong
Post scores	Weekly	Nothing to approve, it's a report

Job	When	Who says yes
Weekly scoreboard (optional)	Weekly	You approve the wording once; numbers filled in on the day

Honest notes before you start

- This module teaches the system, not results. I make no promise about followers, reach or sales.
- Only you can do what the platform asks of a person. If X asks you to confirm your identity or shows a warning, that step is yours. The bot's job is to pause every routine that posts and tell you at once.
- Runs can fail. A posting run can fail, for example when the team's usage limit is reached for the day. The bot writes down the failure and moves the post, with the approved text unchanged, after checking it isn't already live.

The lessons

- 6.1 Weekly batches: one file, one approval, posted on time.
- 6.2 Replies: drafts by default, and when a standing permission makes sense.
- 6.3 The post-score tracker: a simple score you can compute every week.
- 6.4 The weekly scoreboard (optional): honest numbers, flat weeks included.

Files in this module

Attach or paste these in your X bot's chat; it keeps its own copies in its workspace.

- `files/06-x/06-role-x-channel-builder.md`
- `files/06-x/06-batch-template.md`
- `files/06-x/06-routine-x-queue-check.md`
- `files/06-x/06-score-posts.py` (a script your bot runs)
- `files/06-x/06-scoreboard-template.md`

> You approve
> The role prompt once, every batch, any standing permission (in writing, in approvals.md), and the scoreboard wording once.

Common mistakes: letting the X bot post "just this once" outside an approved file. Once that happens, the queue check has nothing to compare against, and you lose the one place where every post can be traced back to a yes.

LESSON

6.1 Weekly batches into the scheduler

What it does

Once a week, the X bot drafts the next week's posts into one file: every slot, the exact text, the reply with the link, the image, and where each number comes from. A second bot reviews the batch. You approve it with one line. Then a routine posts one approved post per slot, and a read-only queue check makes sure nothing unapproved slipped in.

One file and one approval a week means you never get 14 separate "can I post this?" messages.

Steps

- Attach `files/06-x/06-role-x-channel-builder.md` in your X bot's chat and ask it to use it as its instructions, with the placeholders filled in from your brand brief. Set the slot times, your niche and your standing permissions (start with "none").
- Pick a batch day. The weekly trend brief (Lesson 3.3) should land the day before, so the batch can use it.
- Attach `files/06-x/06-batch-template.md` and ask the bot to use it for every batch. It fills in every slot and the "numbers in this post" line.
- Ask it to run the claim check (Module 1) on the whole batch before it reaches you.
- Ask a second bot (mine is the CMO) to fill in the review table: KEEP, CHANGE or CUT, with one line of reason each. It compares the batch with the last 30 days of posts.
- Ask the X bot to paste the batch in chat or attach it. Read it and reply with one line: "Yes to all", or "Yes except 4 and 7".
- Posting: either X's own scheduler, a scheduling tool, or one routine that posts one slot per run and reports the link after each. If you post to several channels, a multi-channel scheduler such as Postiz (affiliate link: I earn a commission if you sign up, at no cost to you) is an option; you connect your accounts to it yourself.
- Ask the bot to set up the queue check from `files/06-x/06-routine-x-queue-check.md`. It runs twice a day, looks 72 hours ahead, and stays silent when everything matches.

Here's how I did it

My batches have 14 posts, two a day, and often go through a few revisions. In review, my CMO bot marks each post KEEP, CHANGE or CUT. The most common reason for a cut is a repeat: a post that pushes something already scheduled, or retells a story almost word for word from an earlier post. That is why the template has a "dropped as repeats" line at the top.

The batch also handles a detail people miss: when the clocks change inside a week, the file says which days are on which offset. Numbers that change daily are written as [placeholders] and filled in from a named file on the day.

The queue check earns its place. In one evening run it found a scheduled post with an out-of-date count of what I offer, and two scheduled posts that repeated posts from the week before. It sent all of that as one message with yes/no questions. Nothing was posted until I answered.

My posts keep links out of the post itself: the link goes in the first reply. That is my rule for X; decide your own and write it in the role prompt.

> You approve
> Each batch, with one line. If the posting routine has to change a word you approved, it stops and asks instead.

Common mistakes: approving numbers in advance. "Three new clients this month" can be true on Monday and false on Friday. Keep numbers as placeholders in the batch and have the routine fill them in from a named file on the day it posts.

LESSON

6.2 Replies, and when a standing permission makes sense

What it does

Posting alone gets little attention on X. Replies on other people's posts, and answers to comments on yours, are where conversations start. The X bot finds relevant posts in your niche, writes short replies that add something (a question, an experience, a correction), and answers people who comment on you.

By default these are drafts, sent to you as a batch. In my setup, after a few weeks of approving reply batches, I gave my X bot one standing permission: it can post replies without asking each time. It is the only standing exception in my whole team, and it is written down in the approvals log.

Steps

- Start with drafts. For two to four weeks, have the bot send a daily reply batch: the post it is replying to, and the exact reply text. You answer "yes", "yes except 3", or "drop 5".
- Watch what you change. If you keep fixing the same thing (too salesy, too long, a link where none belongs), ask the bot to add that to its instructions as a rule.
- Write reply rules into the role prompt: a reply adds one thing; no links unless someone asked; no pitching; no replies on topics you never discuss.
- Decide on a standing permission only when you have stopped changing the drafts. Ask the bot to write it in `approvals.md` with its exact scope: what it covers, how many per day, and what it never covers.

- Keep a log. Every reply the bot posts goes into a replies log with the target post, the text, and whether it was verified as live. Ask to see it now and then.
- Keep an emergency stop. If the platform shows a warning or a limit, the bot stops the routine and tells you immediately.

You don't have to give a standing permission at all. Plenty of people keep replies as a daily batch for good, and that's a perfectly good setup.

Here's how I did it

My standing permission is written in my brand brief and my approvals log: the X bot can post replies without my approval. Everything else still needs my yes: posts, threads, comments on other platforms, and DMs.

My daily reply routine aims for a set number of replies, spaced out through the day, and logs each one. It skips a post when the author limits who can reply. It sends me a one-line count every day, and messages me at once on any warning from X.

The permission did not spread. When I later decided about Instagram comments, I chose the opposite: on Instagram, the bot prepares a list of posts with the exact comment for each, and I post them by hand (Lesson 5.6).

> You approve
> Reply batches until you trust them. A standing permission is a separate, written yes with a clear scope, and you can withdraw it any time.

Common mistakes: treating a standing permission as a general "go ahead". Scope it to one action on one platform, write it down, and have every bot read the approvals log before acting.

LESSON

6.3 The post-score tracker

What it does

Likes are easy to get and tell you little. My X bot scores every original post with one number: (comments + shares) ÷ views. Comments and shares mean someone cared enough to answer or pass it on. A second number, saves rate (bookmarks ÷ views), shows what people want to keep.

Once a week the bot pulls the account's posts, runs the scoring script, and sends you the best and weakest post of the week with one line on why. Over time the table shows which topics and formats work for your audience.

How the score is counted

- Views = impressions.
- Comments = replies under the post, minus your own direct replies (thread parts and link replies), so a thread does not score itself.
- Shares = reposts + quotes. DM shares aren't counted, because X doesn't expose them to the script.
- Posts under 48 hours old are marked "still settling". Metrics refresh for 14 days, then freeze.
- Threads are scored on their first post.

Steps

- Attach `files/06-x/06-score-posts.py` in your X bot's chat and say: "This is a scoring script. Keep it in your workspace, set my handle, user ID and time zone at the top, and run it every week."
- Give it your topics: a short list of what you post about and the words that identify each (the `TOPIC_RULES` list in the script). When a label is wrong, tell the bot, and it overrides that post's label.
- Ask for the weekly routine: the bot fetches your timeline for the last 14 days with its X tools, saves the pages, runs the script, and attaches the result (`post_scores.md`) in chat.
- Ask it to add a short insights note: what worked, what didn't, one idea for next week. It may not invent numbers and must say when the sample is too small.
- The best two topics feed the next batch (Lesson 6.1).

If you're comfortable with a terminal, you can also run the script on your own computer; the instructions are at the top of the file. Either way, the bot has to fetch the timeline pages first, because the script doesn't call X itself.

What to adapt

Pick the score that matches your goal. If you want enquiries, count replies and DMs. If you want readers, count link clicks from your own site's analytics. Keep it to one or two numbers you can compute the same way every week.

Here's how I did it

My bot's first run concluded "sample still too small", and that was the right call. A handful of posts with small view counts is not a pattern. That's what an honest early tracker looks like, and most early accounts look like that. The value is the record you build up over months, not the first week's table.

Two technical lessons from building it: X's search only found posts under the current handle, so the script uses the timeline, not search, to find posts; and some timeline pages came back without an author field, so the script fills it in for your own timeline pages.

> You approve
> Nothing. It is a read-only report. You decide what to do with it when you approve the next batch.

Common mistakes: reading a winner into tiny numbers. Until your posts get a few hundred views each, treat the tracker as a record, not a verdict. Let it run for a month before you change your topics because of it.

LESSON

6.4 The weekly scoreboard, flat weeks included (optional)

What it does

This lesson is for you if you build in public: you share your progress openly as part of your brand. If you don't, skip it.

Once a week, the X bot posts a short scoreboard: two or three numbers you chose, what is live, and one thing planned for next week. The wording is approved once. The numbers are read on the day from named sources (the live profile, a stats file). If any number can't be checked that day, the post doesn't go out.

Why post flat numbers? Because building in public only works if the numbers are real. A scoreboard that only appears in good weeks is an advert. A scoreboard that also shows the weeks where nothing moved is a record people can trust, and it keeps your whole team honest.

Pick your numbers

Choose numbers your bots can read from a source, not from memory. Examples:

- Followers on the channel itself (read from the live profile).
- Site visits or enquiries (read from your own stats, if you keep them).
- For a service business: enquiries received, read from your enquiries log.

Share only what you're comfortable sharing in public. A scoreboard with one honest number is better than one with five you'd rather hide.

Steps

- Attach `files/06-x/06-scoreboard-template.md` in your X bot's chat. Write Part A in your own words and keep the {CURLY} fields for numbers.
- Approve the wording once. Ask the bot to log it in `approvals.md`.
- Make sure each number has a named source that's kept up to date (for example, a read-only routine that writes a stats file daily). The scoreboard reads from it; it never logs into your tools itself.
- Ask the bot to create a weekly routine from Part B. The key line: "If any number cannot be read and checked today, do not post."
- Add a duplicate check: search your recent posts for the first line before posting.
- After posting, the bot sends you the link and the numbers it used.

Here's how I did it

My scoreboard wording is approved once and logged. My first version had been typed into X's scheduler with fixed numbers. Fixed numbers go stale, so I replaced the scheduled copy with a routine that reads them fresh on the day.

One wording rule I added after review: no goal amounts next to the results. A target is not a number you have reached, so it does not belong on a scoreboard. Put it in a separate post about your plan, if anywhere.

> You approve
> The scoreboard wording once. Each week's numbers are not approved one by one; they are read from named sources, and the routine refuses to post if it can't verify them.

Common mistakes: skipping the flat weeks. The week nothing moves is the week your readers learn the scoreboard is honest. Post it, in one line if you have to: "No change this week. Here's what I'm trying next."

MODULE 7

YouTube without a camera

Script and claim check, narration in your own voice (recorded or cloned), screen captures done by the bot, the edit in Remotion, music mix and captions, thumbnails and upload. An optional bonus lesson covers an avatar presenter.

You'll have: a full pipeline for a tutorial video with no filming, and your yes at the script, the cut, the thumbnail and the publish button.

8 lessons · 19 files

LESSON

Module 7 · YouTube without a camera

In this module: 8 lessons (one of them an optional bonus), 19 files (including a Remotion starter project and two scripts). Reading time about 45 minutes. This is the centrepiece of the free guide.

What this module covers

You can make a full YouTube tutorial without filming anything. My first tutorial video was made this way: the team wrote the script, checked every claim, generated the narration in my own cloned voice, recorded the screen, edited the whole thing in code, mixed the music, built captions and thumbnails, and prepared the upload. I approved the script, the fixes, the final cut and the thumbnail.

The core path needs no avatar and no face on screen. It works with three ingredients:

- Your voice: recorded by you, or a clone of your own voice.
- Screen captures: the tool, website or process you're explaining, recorded by a bot.
- Visuals that aren't footage of you: title and chapter cards, text animations, diagrams, stock clips or AI-generated b-roll.

If you made an avatar in the bonus lesson (2.3), you can add it as a presenter (Lesson 7.3, optional). You can also use your own likeness or no presenter at all. It's your choice; just check each platform's rules for labelling AI-generated or altered content.

The method suits any brand that explains things: a software walkthrough, a recipe breakdown with photos and cards, a "how I price a commission" explainer, a tour of a service.

The pipeline

Stage	Lesson	Who does it
Script and claim check	7.1	Script bot + claim check; you approve
Narration in your voice	7.2	You record, or a voice tool from your own clone

Stage	Lesson	Who does it
Bonus: avatar render (optional)	7.3	Avatar tool; only if you chose the bonus avatar
Screen captures and other visuals, privacy masks	7.4	Capture bot on its own machine
The edit in Remotion	7.5	Editor bot, in code
Music, mix, captions	7.6	Editor bot
Thumbnails, metadata, upload	7.7	Editor bot drafts; you pick, approve and do the verification steps

What stays with you

- Your voice. Either you record the narration, or you create a voice clone from your own recordings and accept the tool's terms. Only clone your own voice.
- Approvals. The script, every claim fix, the edit plan, the final cut, the thumbnail and the publish button.
- Platform checks. YouTube asks a person to verify the channel before some features work (for example, custom thumbnails, longer videos, pinned comments and clickable links). At the time you read this, check YouTube Help and your channel's feature eligibility page for what each level unlocks. No bot can do this for you.
- Taste. The music mood and the thumbnail are yours to pick from options.

Honest notes

- My first video took five cuts. Plan for revisions on your first one.
- I don't show tool prices or my costs in this module. Prices change; check each tool's current pricing before you sign up, and approve every subscription yourself.
- I make no promises about views or subscribers. This module teaches production, not growth.

Files in this module

Attach these in your editor bot's chat; it keeps its own copies and runs the scripts in its workspace.

- `files/07-youtube/07-script-template.md`
- `files/07-youtube/07-claim-check.md`
- `files/07-youtube/07-edit-plan-template.md`
- `files/07-youtube/07-remotion-starter/` (generic template, type-checked)
- `files/07-youtube/07-mix.py` (a script your bot runs)
- `files/07-youtube/07-qc-tools/` (format, loudness, freeze, black, sync, privacy sheet; your bot runs them)
- `files/07-youtube/07-yt-metadata-template.md`
- `files/07-youtube/07-upload-checklist.md`

> You approve
> Script, claim fixes, edit plan, final cut, thumbnail, metadata, and "publish". Six yeses for one video, each a quick read because the bots did the checking first.

Common mistakes: starting with the edit or the visuals because they are the fun part. The script and the claim check decide whether the video is any good. Everything after them is execution.

LESSON

7.1 Script and claim check

What it does

The script bot writes a narration script from material you already approved, such as a blog post and its notes. Only lines starting with `>` are spoken. Visual notes go in `[SCREEN: ...]` brackets for the editor. Then a claim check reads every spoken line and the description, finds the source for each fact, and proposes exact replacement lines for anything that fails. You approve the fixes before any narration is recorded or generated.

Fixing a line in text costs a minute. Fixing it after the narration and edit exist costs a re-render.

Steps

- Pick one question the video answers. That question, or its answer, is the title.
- Give the script bot only approved sources: a blog post, your notes, the page about your product or service. Attach `files/07-youtube/07-script-template.md` and ask it to list the sources at the top.
- Chapter structure: a short answer first (under 40 seconds), then one chapter per question people search for, then a short FAQ.
- Write for the ear: short sentences, numbers spoken as words, no URLs read aloud ("The link is in the description").
- If facts can change (prices, features), say so in the script without a date: "Check the current details on the tool's own page."
- Attach `files/07-youtube/07-claim-check.md` and ask the bot to run it. It marks each claim Pass, Fix or Unverified, with the source quoted, and pastes the report in chat.
- Approve or edit the fixes. Then, and only then, record or generate the voice.

What to adapt

A cooking channel's claims are about ingredients, times and temperatures. A consultant's are about methods and what a service includes. The check is the same: every fact has a source, and the bot proposes an exact fix for anything that hasn't.

Here's how I did it

The script for my first video had nine chapters. I approved it with one small fix.

The claim check then found one more problem, and it was not a wrong fact. A line said the test report was one "that I pasted in". In reality, the test was a run made by my team's test pass, not by me personally. My team rule is that team-made work is described as "made with this setup". The proposed line became: "This is a real test run of the prompt, made with this setup. But it ran on a made-up report pasted into the chat, not on a live account. So this is a fictional list, and these are not real numbers."

I approved it. Only that one line was regenerated in the same voice and spliced into the narration at its pauses. A speech-to-text pass confirmed the new wording. Every other claim, from what the connected tool can and can't do to the setup steps, passed with a quoted source.

> You approve
> The script, then each claim fix as an exact line. No narration is made before both.

Common mistakes: checking only facts about tools. Check who did the work and whether example data is real. "I tested this" said over a bot's test run is a claim too, and viewers can't tell the difference unless you tell them.

LESSON

7.2 Narration in your own voice: recorded or cloned

What it does

The narration carries the whole video, so it should be your voice. There are two honest ways to get it:

- Record it yourself. Read the spoken lines into a decent microphone in a quiet room, one file per chapter. The bot does the rest: cleaning, levelling, joining and checking.
- Use a clone of your own voice. You create the clone once in a voice tool from your own recordings. The bot then generates the narration from the approved script, one chapter at a time.

I use a clone (made with ElevenLabs) because it lets the team fix one line without me re-recording. Either way, the processing and checks below are the same.

Disclosure

If the narration is a clone, say so. My descriptions carry a line like: "Voiceover: my own voice, AI-generated with {{VOICE_TOOL}}." On YouTube, set the altered or synthetic content setting as the platform asks; on Instagram, switch on the AI label. At the time you read this, check each platform's current rules on labelling AI-generated content.

Steps

- You: either record each chapter yourself, or create the clone in your voice tool from your own recordings and accept its terms. Never clone anyone else's voice.
- If you use a clone, ask the bot to save the voice settings in a notes file next to the project (model, stability, similarity, style) and use the same settings for every video, so the voice stays the same.
- The bot extracts the spoken lines (every line starting with `>`) and splits them by chapter.
- It generates (or takes your recordings as) one file per chapter, uncompressed, at 48 kHz.
- It processes each file the same way: trim silence, gentle limiting, level to -16 LUFS, then add exactly 300 ms of silence at the start and end.
- It joins the chapters with a fixed pause between them. Mine is 1.2 seconds.
- It runs speech-to-text on the result and compares it with the script, listing any wrong words, odd stress or mispronounced names.
- It regenerates (or asks you to re-record) only the bad lines, and splices them in at silences.
- It writes the disclosure line into the metadata file (Lesson 7.7), if you used a clone.

Ask the bot to attach the full narration in chat when it's done. You listen once.

Here's how I did it

My first video's narration was nine chapter files. Each was levelled to -16 LUFS with exactly 300 ms of silence at each end, and the narration notes record every measurement per file.

After the claim-check fix (Lesson 7.1), only one line was regenerated. The bot cut the old line out at the silent dips before and after it, kept the original breath that leads into it, and matched the pauses to the original take. It compared loudness, brightness and pitch of the new line against the old one before splicing, instead of regenerating the whole chapter.

A second change trimmed a spoken URL off the end of the last chapter. The notes say: "Worth one listen at the very end." The bot measured everything it could, then asked for one human listen where measurement wasn't enough.

> You approve
> Your recordings or the clone (only you can make either), the disclosure wording, and one listen to the full narration before the edit is built on it.

Common mistakes: changing voice settings (or microphone and room) between videos or between fixed lines. The splice will be audible. Keep the setup the same, and treat it like your brand colours.

LESSON

7.3 Bonus: avatar render (optional, if you chose the bonus avatar)

What it does

This lesson is optional. Skip it if you didn't make an avatar in the bonus lesson (2.3), or if you'd rather your videos show only the screen, cards and visuals. Nothing else in this module depends on it.

If you have an avatar, you can animate it so it speaks your narration: a consistent character, a stylised look, or your own likeness, whatever you chose in Lesson 2.3. It's your choice. Check each platform's AI-label rules and disclose an animated presenter in the description.

In my setup, the avatar is animated with Hedra from one still image and the full narration file, so the lips follow the words. It appears full-size on the title and end shots and as a small corner card during explanations. It never covers a screen recording.

Steps

- Pick an image of your avatar that suits a presenter: open hands, facing the camera, enough room above the head.
- Ask the bot to make a short test first: about 10 seconds of narration with that image. Check the mouth, the hands and the eyes before you approve credits for the full length.
- Render the full narration in one clip, so the avatar's timing equals the narration's timing.
- Convert the clip to your timeline's frame rate if needed. Mine came out at 25 fps and was converted to 30 fps by repeating frames, which keeps timing unchanged.
- In the edit, always play the avatar at the narration's own time: timeline time minus the lead before the first word. Then lip sync holds in every shot, even after cuts.
- Plan placement in the edit plan: full avatar on the right third with text on the left; corner card about 320 px tall with a 48 px margin; never over a screen take.
- Check lip sync on the final file at several points (Lesson 7.6).

Here's how I did it

When my edit plan was written, the avatar existed only as still images. The plan said so, proposed gentle motion on stills as a fallback, and listed "who makes an animated version, and how" as an open question for me. The animated version was then made with Hedra: a 10-second test first, then the full clip.

In the final cut, the mouth area of the avatar was compared with the source clip at 10 points across the video, both full-size and corner. The best match was at 0 frames offset at every point. One honest detail from the check: the mouth leads the sound by a few frames in the tool's own output. That comes from the tool, not the edit, and it is within what viewers accept.

I don't list what the render cost. At the time you read this, check the tool's current pricing, and approve the spend yourself.

> You approve
> The avatar look (once, in Lesson 2.3), the 10-second test, and any spend on the full render.

Common mistakes: rendering the avatar per shot. Each render drifts a little. One clip for the whole narration, played at the narration's own time, keeps sync simple.

LESSON

7.4 Screen captures and other visuals, done by the bot

What it does

Most of a camera-free video is what's on the screen. The capture bot records the screen on its own computer (not yours): the browser, the tool or website being explained, the steps in order. It follows the edit plan's shot list, records each shot as a separate take, and never touches anything that would connect an account, spend money or show private data. Every take gets a privacy review before the editor can use it.

Not everything has to be a screen recording. The rest of the picture comes from:

- Cards and text animations: titles, chapter cards, key points, lists, built in the edit (Lesson 7.5).
- Diagrams and simple animations: a process drawn step by step.
- Stock clips or photos with a clear licence, or AI-generated b-roll in your colours.
- Your own photos: your products, your workspace, your work.
- An avatar, only if you chose the bonus (Lesson 7.3).

Steps

- Let the bot capture on its own computer, with a separate browser profile for capture, never your everyday one.
- Attach `files/07-youtube/07-edit-plan-template.md` and ask the bot to fill it in from the approved script, including the data safety rules: what may never appear in a frame, and which buttons are never clicked. You approve the plan before any recording.
- Use fictional example data only, and keep a "fictional" label on screen whenever it shows.
- Capture the page area only: no address bar, tabs, desktop, clock or notifications.
- Clean the screen before each take: one tab, no banners, no pop-ups, sidebar collapsed.
- Move the cursor slowly and deliberately. Paste long text from a file rather than typing it.
- Record more than you need: a couple of seconds of still handles at the start and end of each take.

- Privacy review: the bot runs the contact-sheet check from the QC tools on every take and attaches the sheet for you, and searches the frames for your name or email if your tools greet you by name.
- Anything private that can't be avoided is masked, tracked frame by frame, and re-checked after masking.
- For stock or AI visuals, the bot saves the licence or the generation log next to the project.

If a step needs a sign-in on the bot's computer, you do that one step yourself. You never share passwords with the bot.

Here's how I did it

For my first video, several steps would have needed a real connection to one of my accounts: the connect flow, a test question, the connector menu. The edit plan chose not to show them. The bot recorded the real public directory page up to, but not including, the "Connect" button, and showed the remaining steps as text cards that are clearly graphics. No mock interfaces were made.

The plan listed everything that must never appear: the chat history sidebar, the account chip, the list of connected tools, any email address, the address bar and tabs. One thing slipped into the raw takes: the chat app greets the user by name above the input box. The editor masked it frame by frame in three shots, then checked every frame of those shots with an image search for the name. No match was found. The "fictional list, not real numbers" label was checked on every frame where example data appears: none missing.

> You approve
> The edit plan with its safety rules, and one test frame before the first take. You also do any sign-in the capture needs, on the bot's computer.

Common mistakes: trusting a blur. Blur fails on moving frames and some zooms. Prefer not filming private screens at all; use a mask only when you must, and check every frame after.

LESSON

7.5 The Remotion edit

What it does

The edit is code. Remotion is a framework that renders video from React components, so your editor bot can build the timeline, the cards, the zooms and the lower thirds the same way it builds a web page, and re-render any shot after a change. One data file holds the shot list with in and out times. The components read it and draw each shot.

This is what makes an AI editor practical: it can't drag clips around in a timeline app, but it can write and change code precisely, measure the result, and render again. You never open the code. You watch what the bot attaches and say what to change.

Steps

- Attach the files in `files/07-youtube/07-remotion-starter/` in your editor bot's chat and say: "This is a Remotion starter. Set it up in your workspace, put my brand colours and font in theme.ts, render a still of the title card and attach it."
- Ask the bot to turn the approved edit plan's shot list into `src/timeline.json`: one entry per shot with kind, in, out, and its text or take file.
- It times the cues from the narration: word timings from speech recognition aligned to the script text, with each cue firing just before its word.
- It puts the cut, masked screen takes and other visuals into the project.
- It uses a slow push-in (3 to 5%) on any screen shot that holds still for more than two seconds, and a ring to point at a button. It never shows a click on Connect or Pay.
- It keeps the "fictional example" pill on every frame with made-up data.
- It renders the picture without sound, mixes the audio separately (Lesson 7.6), and joins them. The commands are in the starter's README, for the bot.
- It makes a contact sheet of one frame per shot and attaches it. Review that before watching the full cut, then watch the cut it attaches.

If you're comfortable with Node.js, you can also run the starter on your own computer; the README explains how. You don't need to.

Here's how I did it

My first video's timeline had about fifty shots across nine chapters, at 30 frames per second, with a 2-second lead before the first word and an 8-second end card.

It took five cuts. The freeze check (Lesson 7.6) found three screen shots that looked frozen on long holds, so the bot added slow push-ins and re-rendered. Then I asked for my avatar (I use the bonus one) to be mirrored in the full-size shots so it gestures toward the text, and for a link to come off the end card, leaving it only in the description. Each change was a small code edit and a re-render of the affected shots, not a rebuild. I approved the fifth cut.

The starter in this module is a stripped-down, generic version of that project: same ideas, far fewer moving parts. It was type-checked and a still was rendered from it. It is a starting point for your own template, not a finished video.

> You approve
> The edit plan before the build, then each cut you watch. Say exactly what to change, and the bot changes only that.

Common mistakes: building a new project for every video. Keep one template with your cards and rules, and change only the timeline file per video. Check Remotion's licence too: some companies need a paid company licence. At the time you read this, read the current terms at remotion.dev/license yourself before you use it.

LESSON

7.6 Music, mix to -14 LUFS, captions

What it does

The editor bot picks a music track you're allowed to use, loops it to the video's length, ducks it well under the voice, fades it in and out, and masters the mix to the loudness YouTube normalises to. It builds captions from the exact script text timed to the narration. Then it runs a set of checks on the final file and writes the results into a delivery-check file before you watch anything.

Steps

- Music: you pick the mood (ask for three options). The bot finds a track with a clear licence (royalty-free or public domain), instrumental, at least 3 minutes or cleanly loopable. It saves the track page and licence next to the project.
- Mix: attach `files/07-youtube/07-mix.py` in the editor bot's chat and ask it to run it with your narration and the music, and attach `mix_report.json` and a short listen. Defaults: music 19 dB under the voice while it speaks, up a little in the lead and end card, mastered to -14 LUFS with a true peak below -1 dBTP.
- Captions: the bot builds an SRT from the script's spoken lines, timed with word timings from speech recognition. Max 42 characters per line, 2 lines, 1 to 7 seconds per cue. Numbers and names are written the way they should read on screen. The file is uploaded, not burned into the video.
- Checks: attach `files/07-youtube/07-qc-tools/` and ask the bot to run every check on the final file: format, loudness, freezes, black frames and narration sync, plus a privacy contact sheet. If you used the bonus avatar, it also checks lip sync. It checks the "fictional" labels for your layout.
- Ask it to paste every result into a delivery-check file with PASS or CHECK per line, and to attach that file with the cut.

Here's how I did it

I picked "warm lo-fi, royalty-free instrumental" from three moods. The bot found a public-domain (CC0) track and saved its page and licence.

The delivery check on my final file measured -14.0 LUFS integrated, a true peak safely below -1 dBTP, music 19 dB under the voice during narration, zero offset between the narration and the final audio at five points, and every caption cue in place. The bot looped the track on bar lines and measured the joins to make sure none sounded like an extra hit. Its note on that is honest: "measured, not listened to."

One check came back below target: the audio bitrate measured lower on average than the encoder setting asked for. The bot listed it as an open item instead of calling it a pass.

> You approve
> The music mood and the specific track. Everything else is measured; you read the delivery check, then watch the cut.

Common mistakes: mixing by ear on laptop speakers. Measure loudness and ducking with numbers, then listen once on headphones for anything numbers miss, like a loop seam or a clipped breath.

LESSON

7.7 Thumbnails, metadata, upload, pinned comment

What it does

The last stage turns an approved cut into a published video. The bot makes three thumbnail options, writes the title, description, chapters and tags, sets the altered or synthetic content setting when it applies, uploads the video as Private with captions, and prepares a pinned comment. You pick the thumbnail, approve the metadata, and say "publish".

Steps

- Thumbnails: three options in your brand style, each tested at 320 px wide (the size most people see). The bot attaches a preview sheet so you can compare them side by side.
- Metadata: attach `files/07-youtube/07-yt-metadata-template.md` and ask the bot to fill it in. Chapters come from the chapter card times in the final cut. The description says what the tools can and cannot do, labels any fictional example, and discloses a cloned voice or an animated avatar if you used them.
- Channel setup (you): create or choose the channel, and sign in where the bot will upload (in my setup, in the bot's own browser). Do the verification yourself. YouTube turns on some features (for example custom thumbnails, longer videos, pinned comments and clickable links) only after phone, ID or video verification or enough channel history. At the time you read this, check YouTube Help and Studio > Settings > Channel > Feature eligibility for the current rules.
- Upload: the bot follows `files/07-youtube/07-upload-checklist.md`. Upload as Private, set altered or synthetic content to Yes if the voice or presenter is AI-generated, upload the caption file, set the thumbnail.
- Publish: only after your "publish". Then the bot checks the live page field by field and reports.
- Pinned comment: one line pointing to the free resource or the related post. Pinning a comment and making its link clickable may need the verified features from step 3.
- The bot hands the link to the other bots: blog embed, X post.

Here's how I did it

After I approved the fifth cut, the bot uploaded the video as Private with altered content set to Yes. Two human steps showed up on the way. First, the account I planned to use wasn't eligible for YouTube, so I chose a different account to own the channel. Second, the captions were still

processing at upload, so the bot set a short routine to retry them. After I checked the page and said "publish", the bot set it to Public.

My description lists the chapters, the free download link, "The example in this video is a fictional list, not real numbers", the voice disclosure ("my own voice, AI-generated with ElevenLabs") and my one-line bio.

The verification step for pinned comments and clickable links is mine to do. A bot can remind me; it can't do it.

> You approve
> The thumbnail, the metadata file, and "publish". You also do the channel setup and any verification step yourself.

Common mistakes: publishing straight from the upload. Upload as Private, check every field on the real page, then publish. A typo in a title is easy to fix; a wrong claim seen by the first hundred viewers is not.

MODULE 8

Reels and short video

A monthly credit plan so nothing goes unused or over budget, and a Reels house style.

You'll have: one bot that owns every AI video spend, with a plan you approve.

3 lessons · 4 files

LESSON

Module 8 · Reels and short video

In this module: 3 lessons, 4 files. Reading time about 15 minutes.

What this module covers

Short vertical video is where AI video tools are most useful, and where it is easiest to waste money. This module shows a way to handle it for any brand: one bot owns every AI video job and runs a monthly credit plan, so nothing expires unused and nothing goes over budget without a yes. A style study turns into a house style.

A bakery might make Reels from photos of the day's bake with short captions. A coach might do one tip per Reel with screen-recorded slides. A software consultant might show one feature per Reel. The roles, the credit plan and the checks are the same.

In my setup two bots share the work:

Bot	Owns
Short Video Creator	AI video and image generation, the credit plan, narration, Reels from idea to export
Video Editor	The editing templates (long and vertical), long-form video (Module 7)

On a small team, one bot can do both jobs.

Rules I set for my Reels

These are my standing rules. They are taste decisions, not data, and they show how your calls shape a format:

- Every Reel has narration. A silent Reel the team made didn't get my yes; I want my voice on every one. If the narration is a clone of your voice, the platform's AI label goes on.
- Links only by DM after a keyword comment (Lesson 5.5): never in the caption, the voiceover or on the end card. The end card shows the keyword instead.

Your team will need your own calls. Write them in the style sheet and the approvals log.

The lessons

- 8.1 A monthly credit plan: use every credit, never overspend.
- 8.2 Reels style: what polished accounts do, and a house style sheet.

Files in this module

Attach these in the matching bot's chat; it keeps its own copies.

- `files/08-short-video/08-role-short-video-creator.md`
- `files/08-short-video/08-role-video-editor.md`
- `files/08-short-video/08-credit-plan-template.md`
- `files/08-short-video/08-reels-style-guide.md`

> You approve
> The monthly credit split (one yes covers spending inside it), any test with its own cap, the house style once, and every Reel before it's posted.

Common mistakes: letting several bots generate video "when they need it". Credits disappear and nobody knows where. One owner, one log, one plan.

LESSON

8.1 A monthly credit plan, so nothing goes unused

What it does

Most AI video tools sell monthly credits, and with many plans unused credits expire when the month renews. That creates two ways to waste money: credits left over at the end of the month, and credits burned early on experiments nobody planned. The credit plan fixes both. Once a month, the short-video bot splits the month's credits into buckets, you approve the split once, and that yes covers every job inside it.

The tool I use for this is Higgsfield: higgsfield.ai (affiliate link: I earn a commission if you sign up, at no cost to you). The method works with any credit-based tool.

Steps

- Give one bot ownership of the tool. Every other bot asks it for clips and images through the shared image request file.
- On reset day, the bot reads the real balance and the rules: does anything roll over, do failed jobs give the credits back? It writes where each fact came from.
- It prices each job type with the tool's free cost preview, if there is one, and writes a cost table in credits.
- Attach `files/08-short-video/08-credit-plan-template.md` and ask the bot to fill it in: buckets, unit cost, volume, total. It pastes the plan in chat.

- Leave a buffer for retries (I use about one retry in six jobs) and a month-end sink: evergreen b-roll clips you'll reuse for months, made in the last few days with whatever is left.
- You approve the split. The bot logs it as a standing permission with its scope ("spend per the monthly split").
- Any one-off test gets its own cap and its own yes.
- The bot logs every job and checks the balance before and after. The log and the balance must agree.

Here's how I did it

My plan splits the month into buckets: Reel b-roll, avatar animation (only because I use the bonus avatar; leave it out if you don't), approved tests, images for other bots, a capped test Reel, retries, and the month-end sink. One standing OK covers spending inside that split.

A test Reel (made to try the tool, not posted) shows how a cap works. I said yes to an estimate with a hard cap above it. Before every job the bot ran the free cost preview. It stayed under the estimate, and the balance before and after matched the log exactly. In review, one shot had an extra finger on a pointing hand, so that shot was generated again for the next version. The bot also noted what it skipped: a screen recording that would have needed a browser login on its computer.

The plan is also honest about what it can't confirm. Some facts are marked "unverified", such as the exact reset date, because the bot couldn't check them without account pages it isn't allowed to open.

I don't show what the subscription costs. At the time you read this, check current prices on the tool's own page, and decide on any subscription yourself.

> You approve
> The monthly split (one yes), each test with its own cap, and any plan change or extra purchase. The bot never buys credits.

Common mistakes: spending a whole month's credits in the first week because a new model came out. Keep the buckets. If something new is worth testing, give it a capped test bucket and approve it separately.

LESSON

8.2 Reels style: what the polished accounts do

What it does

Before making Reels, my Short Video Creator studied how popular accounts in my niche edit theirs: captions, b-roll, pacing, hooks, colour, motion graphics. It wrote down where each observation came from, and marked which ones it confirmed from a source and which came from general knowledge of the format. From that, it wrote a house style for my brand. The style sheet is a one-page file every Reel is checked against.

Steps

- Ask your short-video bot for a style study of 10 to 12 accounts in your niche. Require a source for each claim, and a flag for anything not confirmed.
- Ask for the patterns they share, not a copy of any one account.
- Attach `files/08-short-video/08-reels-style-guide.md` and ask the bot to turn the patterns into your house style in Part B: your brandbook colours, your font, and who (if anyone) appears on screen.
- Add your own rules: disclosure, never-mention topics, how links are given.
- Make one test Reel in the style before committing. Ask the bot to attach it with a contact sheet, and review it frame by frame.
- Keep one template in code (Lesson 7.5) for captions, cards, zooms and the end card, so every Reel looks the same with little effort.

Who appears on screen

That's your choice: you on camera, your own likeness, a character or stylised avatar (Lesson 2.3, bonus), your products and hands, or nobody at all, just screens, photos and text. Each works. If anything on screen or in the audio is AI-generated, check each platform's current AI-label rules and follow them.

What to adapt

A local business's "polished accounts" might be other cafés, salons or shops, not tech creators. Study the accounts your audience already watches. The patterns (fast payoff, motion in every frame, one caption system) tend to carry over; the look should be yours.

Here's how I did it

The study found eight shared patterns. The ones I use most: the payoff arrives in the first 1 to 2 seconds; something moves in every frame, even a slow 3–5% push-in on a screenshot; one caption system used every time; and a rehook every 5 to 7 seconds.

My house style that came out of it: 20 to 35 seconds, one thing per Reel; a short headline in my text colour with one accent word, over the most striking visual; my bonus avatar in a corner; screen recordings in a dark frame with zoom and a ring on the click; captions of 1 to 4 words in the upper middle; a cut every 1.5 to 2.5 seconds; my cloned voice (music under it is optional), so the AI label goes on.

My feedback on my test Reels added two rules the research didn't: every Reel has narration, and links go only by DM after a keyword comment (never in the caption, voiceover or end card). The style sheet now includes both.

The research also says plainly what to ignore: the "I built X in 10 minutes" hook doesn't fit my rule against unmeasured time claims.

> You approve
> The house style once, the test Reel, and every Reel before posting. Taste rules you add later go into the style sheet and the approvals log.

Common mistakes: copying one big account's look. You'll be compared with the original. Take the shared patterns, then make the colours, the visuals and the voice your own.

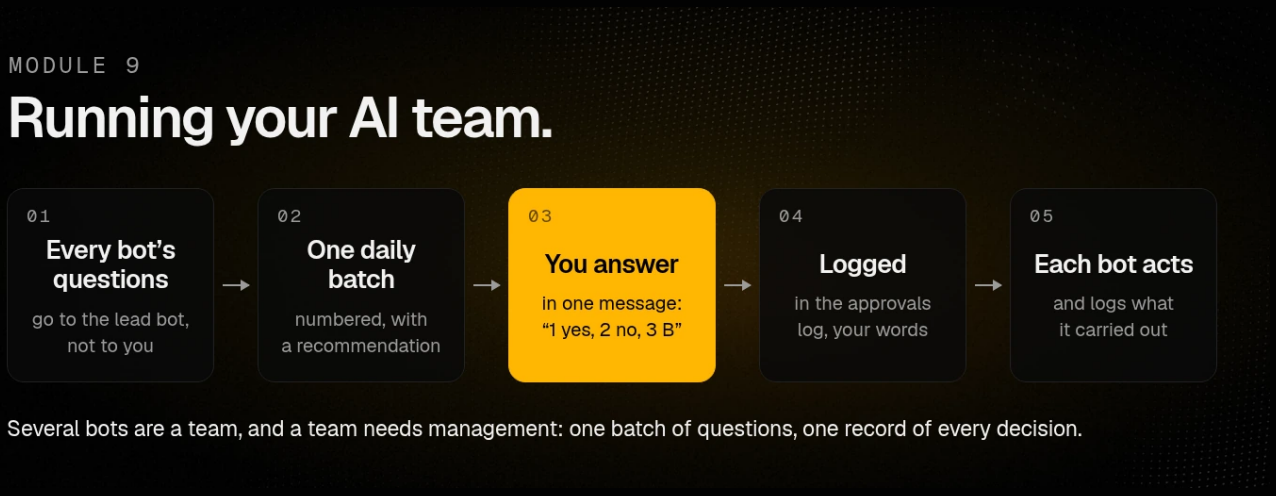
MODULE 9

Running your AI team

A lead bot's daily summary and batched decisions, daily self-review per bot, a maintenance bot, handoffs through shared files, and what to switch off.

You'll have: one daily batch of decisions you answer in a single message.

6 lessons · 16 files



The workflow from Module 9.

LESSON

Module 9 · Running your AI team

In this module: 6 lessons, 16 files. Reading time about 25 minutes. The lesson count includes this overview.

What this module covers

One bot is a helper. Several bots are a team, and a team needs management: someone who collects the questions, a record of every decision, a way for each bot to learn from its own work, and someone who notices when routines are wasting time and money. Without that, you spend your day answering the same question from three bots.

This module is how I run mine. Here is my team, as an example:

Role	Main job
CEO	Daily summary, decision batches, the brand brief
CMO	Instagram, website, brand visuals
Research	Daily trend feed, weekly trend brief
X Channel Builder	X posts, replies, scores, scoreboard
Short Video Creator	AI video, credit plan, narration, Reels

Role	Main job
Video Editor	Long videos, screen recordings, templates
Digital Products (optional)	Offers and listings, only if you have them
Business Development Manager	Partnerships, affiliates, leads
SEO & AEO Desk	Search research, page metadata, reports
Engagement	Comments and DMs, one daily approval batch
Maintenance bot	Audits routines and chats, proposes fixes

Yours will be smaller, especially at first. Three roles (a lead, a content bot, research) are enough to start (Lesson 0.2). Everything in this module works the same with three bots as with eleven.

I added the Engagement role later, when comments and DMs moved off my CMO's list. Lesson 9.4 shows how that handoff worked.

What stays with you

Every decision in the daily batch. Which roles exist. Whether a fix the maintenance bot proposes gets made. And the things only a person can do: sign-ins, verifications and payments (Lesson 0.4).

How much of your time it takes

My clearest example: once, I answered a batch of more than twenty decisions from across the team in one message. Most answers were a word or two. The bots had done the research, written the options and made a recommendation for each. That is the point of the CEO bot.

The lessons

- 9.1 The CEO bot: daily summary, decisions batched into one message.
- 9.2 Daily self-review per bot: three lines a day.
- 9.3 The maintenance bot: routine and chat healthchecks.
- 9.4 Handoffs through shared files, not memory.
- 9.5 What to switch off: routine waste.

Files in this module

Attach these in the matching bot's chat; each bot keeps its own copy.

- `files/09-running-the-team/09-role-ceo.md`
- `files/09-running-the-team/09-routine-daily-summary.md`
- `files/09-running-the-team/09-self-review-template.md`
- `files/09-running-the-team/09-routine-healthcheck.md` (routine and chat healthchecks)
- `files/09-running-the-team/09-role-prompts/` (all 11 role prompts and an index)

> You approve
> The decision batch, every change to roles and rules, and every fix the maintenance bot proposes.

Common mistakes: adding bots before you have a lead bot and an approvals log. More bots means more questions. Without batching, you become the bottleneck you were trying to remove.

LESSON

9.1 The CEO bot: daily summary and batched decisions

What it does

The CEO bot (your lead bot) doesn't make content. It does two things for you every day. First, a short summary at a fixed time: what shipped, what failed, what is waiting on you, and any real numbers you track. Second, a decision batch: every open question from every bot, collected into one numbered message with options and a recommendation, so you can answer them all at once.

When you answer, the CEO bot writes each decision into the approvals log with your words quoted, and tells each bot what was decided for it.

Steps

- Attach `files/09-running-the-team/09-role-ceo.md` in your lead bot's chat and ask it to use it as its instructions, with the placeholders filled in from your brand brief. Give it ownership of the brand brief and the approvals log.
- Attach `files/09-running-the-team/09-routine-daily-summary.md` and ask it to create the daily summary routine. Pick a time when you usually have five minutes.
- Make "read approvals.md first" the first step of every summary and batch. Nothing already decided may appear as waiting.
- Define "waiting on you" narrowly: only steps no bot can do (sign-ins, verifications, payments, taste calls, decisions).
- Tell every other bot: questions for you go to the CEO bot's batch, not to you directly, unless it's urgent (a platform warning, a failed payment, a verification request).
- Answer the batch in one message. Numbers and short words are enough: "1 yes, 2 no, 3 B".
- Once a week, ask the CEO bot to show you the approvals log. Every yes should be there, in your words.

Here's how I did it

My summary runs every evening and posts to the team's group chat: what shipped, what's open in the decision batch, and the numbers from the files that track them.

I added the rule "read approvals.md first" to the routine so that nothing already approved is ever listed as waiting. One wrong "waiting on you" line costs your trust in the whole summary, so the rule lives in the routine itself, not in anyone's memory.

My big batch shows the format. Each item was one line with the options. I answered all of them in one message, and the CEO bot logged each answer word for word in the approvals log: a yes to a project, a no to an affiliate network, a pick for the website direction, and, for Instagram comments, the decision that the bots prepare exact comments and I post them by hand. Each bot then acted on its own items, and each action was logged again by the bot that carried it out.

> You approve
> Everything in the batch. The summary itself needs no approval; it's a report.

Common mistakes: a summary that's a list of everything every bot did. Keep it under 200 words. If you have to scroll, it's not a summary, and you'll stop reading it within a week.

LESSON

9.2 Daily self-review per bot

What it does

At the end of its main daily job, each bot writes three lines into its own self-review file: what worked (with a real number if it has one), what to change tomorrow, and one idea. It takes the bot seconds. Over a week, the file shows you what each bot is learning, where it lacks data, and which changes are worth approving.

I asked for this because I want every bot to evaluate its own work every day, question its own decisions, and get better. The three-line format keeps that from turning into long essays nobody reads.

Steps

- Attach `files/09-running-the-team/09-self-review-template.md` in each bot's chat (or ask your lead bot to pass it on to every bot in the same workspace).
- Ask each bot to add Part A as the last step of its main daily routine, and to keep one self-review file of its own.
- Require a source for "what worked". If there's no data, the bot says so and says what data it would need.
- Changes to the bot's own work (tomorrow's drafts) it can just make. Changes to rules, roles or other bots are drafts for you.
- Optional: once a week, the CEO bot reads all self-review files and sends you at most three changes worth approving (Part C).
- Once a week, ask one bot to show you its self-review file. It's the fastest way to see how a bot thinks.

Here's how I did it

My CMO bot's first self-review entry is a good example of an honest one:

- What worked: no reach data. The tool it publishes through had no analytics, and there was no Instagram insights export available to it. The only signal it had was the comments on the first post, too few to mean anything on day one.
- What to change: fewer of my own product how-tos in Stories, more news people are already searching for, explained with one concrete example. Each frame asks for one easy reply.
- One idea: get Instagram insights into the shared files daily, so tomorrow's review can rank frames by real numbers instead of guesses.

Notice what it didn't do: invent a reason why something "performed well". It said there was no data, and its idea was how to get data. That third line is often the most valuable one, because missing data is the most common problem in a young setup.

> You approve
> Any change a self-review proposes to rules, roles, budgets or other bots. Changes to a bot's own next drafts don't need a separate yes; you'll see them in the drafts.

Common mistakes: letting self-reviews grade themselves ("Great day, everything went well"). Require a number or an honest "no data". A review without evidence is just a mood.

LESSON

9.3 The maintenance bot: chat and routine healthchecks

What it does

The maintenance bot doesn't make content. It watches the team. Once a week it reads every bot's routines file and flags waste: routines that run too often, message you when nothing changed, are dead, or duplicate another. On weekdays it scans the bots' recent chats for friction: you correcting the same thing twice, a bot asking something already decided, the same step failing again. It proposes one fix per problem and changes nothing until you pick.

Steps

- Make sure every bot keeps a routines file (Lesson 1.4). If your bots share a workspace, the maintenance bot can read them all. If not, ask each bot to attach its routines file in the maintenance bot's chat, or paste them in yourself once a week.
- Create the maintenance bot: attach `files/09-running-the-team/09-role-prompts/role-maintenance.md` in a new bot's chat and ask it to use it as its instructions.
- Attach `files/09-running-the-team/09-routine-healthcheck.md` and ask it to create both routines: weekly for routines, weekdays for chats.

- Keep both quiet when nothing is flagged.
- Answer its list by number: "Do 1 and 3, skip 2." The owning bot (or the maintenance bot, with your yes) makes the change and updates its routines file.
- If a fix keeps coming back, turn it into a rule in the brand brief or a role prompt, so it stops recurring.

What it looks for

- Dead: a one-off routine that already fired but is still marked enabled.
- Paused and never run: a routine paused before its first run and left that way. It should be deleted or restarted, not kept forever.
- Noisy: a routine that messages you when nothing changed.
- Too frequent: a weekly job checked daily.
- Duplicate: two bots checking the same thing.
- No report: a bot whose routines file is missing or says only "none". That's allowed, but the healthcheck confirms it's true.

Here's how I did it

My maintenance bot runs the routine healthcheck once a week and the chat healthcheck on weekday mornings. Both stay quiet when there's nothing to report. When they do report, I get one short list, and I decide. The maintenance bot doesn't fix anything by itself.

> You approve
> Every fix it proposes. It has no permission to edit another bot's routines, prompts or files on its own.

Common mistakes: letting the maintenance bot "just clean up". A routine that looks dead might be a yearly reminder you still want. Proposals only; you pick.

LESSON

9.4 Handoffs through shared files, not memory

What it does

Bots forget. Chats get long, get summarised, or start fresh. So in my team, nothing important lives only in a chat. Work moves between bots through shared files: the brand brief says who does what, the approvals log says what you decided, the routines files say what runs when, and each piece of work is a named file another bot can open.

When one bot hands work to another, it sends a short message naming the file, not the work itself. The other bot opens the file.

Where the shared files live

Your bots keep these files in their own workspace, as set up in Module 1. Bots in the same workspace all read the same files, so a handoff is just "it's in research/latest.md". For a bot that doesn't share the workspace, the handing-over bot (or you) attaches the file in that bot's chat. Either way, you see any file by asking a bot to show it or attach it.

Steps

- Ask your lead bot to keep the shared files from Module 1 where every bot can read them: brand brief, approvals log and routines files.
- Put "who does what" in the brand brief (I call it Part 2). One line per role: what it owns, and what it doesn't.
- Every handoff message has three parts: what's ready, the file name, and what the receiving bot should do with it.
- Shared outputs live in fixed places with a `latest.md` where useful: the research feed, the scoreboards.
- When a role changes, the lead bot updates Part 2 the same day, and the routines move: the old owner removes them from its file, the new owner adds them to its own.
- When you approve something in one bot's chat, that bot writes it into the approvals log. Other bots read the log, not each other's chats.

Here's how I did it

When I turned an idle bot into a new role, Engagement (one daily approval batch for comments and DMs), the handoff took four file changes:

- The approvals log got the decision.
- The brand brief's Part 2 got a new line for Engagement: what it owns, and what it doesn't (not X, not feed posts or Reels, not leads or affiliates).
- Engagement's routines file got its daily approval batch.
- The CMO bot recorded that its engagement work had moved to Engagement.

No bot had to remember the change. From then on, any bot that read the brief knew who to send comment work to.

Small daily handoffs work the same way. The research bot writes its feed to `research/latest.md` and pings the lead bot with the file name and top three ideas. The CMO's daily drafts read that same file. Any post that quotes a number reads it from the file that tracks it. Nobody copies numbers from a chat.

> You approve
> Role changes (they're decisions like any other, so they go in the approvals log). Day-to-day handoffs between bots need no approval; the work they hand over still does.

Common mistakes: "I told the CMO bot last week." If it isn't in a shared file, assume the other bots don't know it, and assume the CMO bot may have forgotten. Write it down where every bot reads.

LESSON

9.5 What to switch off

What it does

Every routine costs something each time it runs: usage, attention, and a little of your trust if it messages you for nothing. A team of bots builds up routines fast. This lesson is about cutting them back: what to pause, what to delete, what to make quieter, and what to do when the whole team has to stop at once.

Steps

- Delete one-offs after they fire. Make "deletes itself after running" part of every one-off routine you create.
- Match frequency to change. A weekly job checked hourly is waste. Ask: how often does the thing I'm checking actually change?
- Add a quiet rule to everything. Routines message you only when something happened or something failed. A daily "nothing new" message trains you to ignore the bot.
- Pause, then decide. A paused routine that never ran is a decision you haven't made. Delete it or restart it within a few weeks.
- Merge duplicates. Two bots checking the same inbox or the same numbers is one routine too many.
- Plan for an emergency stop. Ask each bot to write down which of its routines post or send. If a platform shows a warning or asks for a verification step, those pause first, and the bot tells you at once. After you've dealt with it, the bot checks what already went live before resuming anything.
- Plan for running out of usage. Runs can fail when a team's usage limit is reached. Each bot should log the failure and reschedule with the approved text unchanged, after checking the post isn't already live.
- Let the maintenance bot (Lesson 9.3) find candidates every week. You decide.

Here's how I did it

Running out. One day several scheduled runs across my team failed because the team's usage was used up for the day. A carousel, a set of Stories and two X posts were moved to new times with their approved text unchanged, and each move was written into the routines files. Nothing was posted twice, because every posting run checks whether its post is already live before it publishes (Lesson 5.3).

> You approve
> Deleting or pausing any routine you created with a yes, and resuming after an emergency stop. Any verification step a platform asks for is yours alone.

Common mistakes: adding routines and never removing any. Once a month, ask each bot to show you its routines file (or ask the maintenance bot for a summary) and ask of each routine: would I create this today? If not, switch it off.

YOUR TEAM DRAFTS. YOU DECIDE WHAT GOES OUT.

Start with step one

Import dr eggbot, tell it what you want to build, and follow the modules in this guide. My own setup is the example.

viktorszabo.com/course/