The AI Playbook

From Your First Prompt to Working Agents

by Burhan Sebin

36 chapters: prompting, personal systems, automations, agents, and shipped apps.

Contents

New here? Read "How to use this playbook" first, then go in order: each part builds on the last.

How to use this playbook

This playbook is not about AI in theory. It is about AI in your work, on Monday morning.

Read it in order the first time. Each part builds on the last: how to think, how to prompt, how to build personal systems, how to automate, how to run agents, how to ship real software, and how to do all of it without leaking your data or embarrassing yourself.

Two rules run through everything here.

Rule one: do not use tools because they are cool. "Oh, this tool is so cool, I need to use it." No. If it does not fix a problem for you, if it does not save time for you, do not use it.

Rule two: accountability stays with you. The AI drafts. You decide. If you send a stupid email to your client, your manager, your CEO, your colleague, that is on you.

Everything in this playbook was tested with real tools, real demos, and real mistakes. What follows is the full curriculum, start to finish.

PART I. THINK LIKE AN OPERATOR

How to think about AI before you touch a tool. The mental models to return to again and again: the innovation reaction loop, how the models actually work, and why you always start from the problem, never the tool.

1. The Innovation Reaction Loop

Every new technology moves through the same human cycle. Call it the Innovation Reaction Loop:

We innovate. We fear. We learn. We grow. Then the cycle repeats, forever.

"We innovate something new. Then we fear again. Then we learn, then we grow. This is human nature. This is a culture."

Run the history of technology through the loop and it fits every time:

- The first tool, a sharpened stone, three million years ago. The fear: it is so dangerous, it is going to kill all of us.
- Mechanization. It scaled muscle. People feared for their livelihoods.
- Electrification. It took decades to redesign factories around it. (The precise "five decades" is color, not a documented figure: Paul David's research supports a decades-long delay.)
- The internet. The first computer. Mobile apps. Microsoft Excel: accountants thought they would lose their jobs. "Maybe some of them did. But it empowered the accountants, and we still have accountants."
- The iPhone. Remember the original announcement: Steve Jobs sliding a finger across the screen to unlock the phone. That is exactly how the first use of ChatGPT felt: "Wow, this is gonna change the world. We're gonna lose our jobs."

Then the loop turns. After a couple of months, you start thinking: what a stupid tool. You said that. Everyone said that. ChatGPT hallucinated. It invented sources. Then video models arrived, Will Smith eating spaghetti, and it was magic again, then silly again: "You're still gonna say this tool is stupid, because it's not gonna create an Instagram Reel at the quality you want... This is human nature, right? We do this every time."

Here is the part that matters: **the learn and grow stage is happening right now.** If you are reading this playbook and doing the work, you are ahead of most people. The tools keep changing. You stay wonderfully consistent.

Two practical consequences of the loop:

1. **Do not marry a tool.** The tools change every month. What you are learning here is judgment: which problem to point AI at, how to brief it, how to verify it, how to run it safely. That transfers to every tool that comes next.
2. **Do not wait for the perfect tool.** The loop guarantees the next version will be better and also that you will complain about it. Start with what exists today, on a real problem, and ride the improvement curve instead of watching it.

2. How the Models Actually Work

Before you prompt, you need a mental model of what you are prompting. Think of it in three levels:

- **Traditional AI is a primary school student.** Ask it two times two, it says four. Ask three times three: "Oh, we didn't learn that." It only knows what you fed it.
- **A large language model is a high school student.** "They know everything about life, right?... You feel like you know everything, you feel like you can do anything, but you don't get life advice from a high school student." Models are fed enormous amounts of data, so they think they know everything. "There is no feature like, 'Oh, I don't know that.'"
- **Fine-tuning is a university student.** After fine-tuning, the model becomes an expert in one specific area. You can almost treat it as an oracle for that domain, but "still accountability should be in your goal. Because basically, it's a generative response. It's not a truth most of the time."

The operating principle: **do not treat the model as an oracle. Treat it as a fast, capable collaborator that proposes work for review.**

A language model generates a likely response, not guaranteed truth. The input is your instruction plus available context plus learned patterns. The output is a proposed answer, analysis, draft, or action. The output can be useful and still be incomplete, outdated, or wrong.

This changes how you work with it:

- You give it evidence, not just questions. "The model cannot see the work the way you do unless you show it."
- You verify the output against sources you trust, especially numbers, names, dates, and anything that will go to another human.
- You keep the final decision. Chapter 8 turns this into a loop: AI accelerates the drafting, and accountability stays with you.

One more thing about how models are built, because it explains why fine-tuning matters: training data (raw examples) goes into pretraining (extracting statistical patterns), which produces learned parameters, the weights that encode those patterns. A transformer is the architecture that uses attention to track relationships across a sequence. None of this is a searchable database of memorized facts, which is exactly why the model can sound confident and be wrong at the same time.

3. Start with the Problem, Not the Tool

"The biggest idea in this book: don't start with the tool because it's cool. Don't start with the model because it's cool. Start with the problem."

This is the sentence the whole playbook hangs on. Most AI projects fail, and it is not the technology's fault. MIT's NANDA initiative published *The GenAI Divide: State of AI in Business 2025*, based on 150 leader interviews, a 350-employee survey, and 300 public AI deployments, per August 2025 reporting on the study. The finding: roughly 95 percent of enterprise generative AI pilots showed no measurable return. The reported cause was not model quality. It was flawed enterprise integration: tools that did not learn from feedback, did not fit existing workflows, and did not solve clearly defined operational problems. "That gap between adoption and transformation is not a technology problem. You cannot buy your way out of it. It's a leadership problem, it's a capital allocation problem."

So the playbook starts where the failures do not: with task selection.

The frequency-by-consequence matrix

Map your candidate tasks on two axes: how often the task happens, and what happens if the AI gets it wrong.

- **Frequent plus lower consequence: start here.** Summaries, first drafts, classification, meeting prep. High repetition, low blast radius. This is where you build skill and trust.
- **Frequent plus higher consequence: add controls.** Use approved sources, rubrics, and human review before anything goes out.
- **Infrequent plus lower consequence: useful, but not your first project.** The return on setup time is thin.
- **Infrequent plus higher consequence: do not begin here.** Use domain experts, governance, and stronger controls. Come back when you have reps.

What makes a good first task

A good first task is **frequent, bounded, reviewable, reversible, and safe.** If you can undo it, check it quickly, and it happens every week, it is a candidate. Email triage, meeting summaries, content first drafts, lead research: all fit.

The turn-one sentence template

When you are ready to brief the AI on your chosen problem, start with this sentence:

PROMPT

```
When [TRIGGER] happens, help me [OUTCOME] using [APPROVED INPUTS]. Return
[FORMAT]. A human must approve [DECISION OR ACTION].
```

Example: "When a new inquiry form is submitted, help me draft a reply using the inquiry details and our pricing page. Return a short email draft. A human must approve before anything is sent."

That one sentence already contains a trigger, an outcome, approved inputs, an output format, and a human gate. Most failed AI projects never wrote this sentence.

4. The AI Terminology Map

People use these terms as if they are interchangeable. They are not. They work as one system, in a single sentence:

Model receives context. Context follows instructions. Instructions call tools. Tools complete the work.

"The model generates. The surrounding system makes the result useful, grounded, repeatable, and safe."

The terms, organized by six questions:

What generates?

- **Generative AI:** creates new text, images, audio, video, or code. A capability category, not a product.
- **LLM (large language model):** a large neural network trained to work with language. A language-focused foundation model.
- **GPT:** Generative Pre-trained Transformer. One transformer-based LLM family.
- **Multimodal model:** accepts or produces more than one content type. Multimodal does not mean every model supports every input and output.

How is it built?

- **Neural network / transformer / parameters / pretraining:** the infrastructure layer. Training data goes in, pretraining extracts statistical patterns, learned parameters (weights) encode those patterns.
- **Token:** a word, part of a word, punctuation, or another small unit. The model generates one token at a time: prompt, candidates, select one token, append, repeat.

How can it be adapted?

- **Fine-tuning:** post-training on specialized data. The university-student step from Chapter 2.
- **Reinforcement learning:** training by rewarding better outputs. One way to think about the shift in reasoning models: an early approach was like punishing failure, while the newer approach shows the correct method and output. Treat that as an illustration, not a citable claim.

What can it see now?

- **Context window:** everything the model can use for this response: system instructions, conversation history, your prompt, files and retrieved passages, tool results, plus room for the generated response. "More is not better. The right, current, approved context matters more than volume."
- **System prompt:** the top-level instruction that sets role, rules, tone, and boundaries.

- **RAG (retrieval augmented generation):** retrieve approved sources at answer time, add the retrieved evidence to the answer context, generate with citations. "RAG changes the evidence available for this answer. It does not retrain the model." Best for current, private, or authoritative material the model was not trained on.

How does it connect and act?

- **API:** a defined point-to-point doorway into one service.
- **MCP (Model Context Protocol):** Anthropic's open standard, announced in November 2024, for exposing approved resources, tools, and reusable prompts that a caller can discover. "An easier way of API for agents." API and MCP are complementary, not competing. Every connection still needs authentication, explicit permissions, and human approval before consequential actions.
- **Agent:** a controlled loop around a model: goal, plan, tool, action, observation. An agent also needs tools, state, stopping conditions, rules, monitoring, and approval boundaries. "More autonomy requires less privilege, better logging, and stronger checkpoints."

Where can it fail or surprise us?

- **Hallucination:** plausible but unsupported, inaccurate, or invented output. "They make up some answers."
- **Bias:** systematic distortion from data, design, evaluation, or use context.
- **Anthropomorphism:** treating AI as if it has feelings, intent, judgment, or accountability.
- **Trust gate:** flag hallucinations, check bias, avoid anthropomorphism, then human approval. "The model may recommend. The accountable person approves."

Two controls people confuse: **reasoning depth** (a reasoning model allocates more computation before answering; it does not guarantee truthfulness) versus **temperature** (changes consistency versus variation, not accuracy). For professional work, request the answer, the assumptions, the evidence, and a concise rationale.

5. Choose Your Setup: Models, Tools, and Personalization

Be tool-agnostic and model-agnostic

"There is no right answer, and it changes literally every month." The best model is not the best model for you. It depends on cost, security, and what you need. "In the end, I think models are gonna be commoditized."

Allegiances shift constantly: "My favorite was two weeks ago it was Claude. Now it is ChatGPT... It changed from task to task, month to month." Working verdicts, tested by daily use:

- **ChatGPT for everyday work.** The frontier model is strong, it is hard to hit usage limits even at max reasoning, and it connects natively to Microsoft tools like Outlook. "Go with ChatGPT right now. That's my suggestion." (A working verdict from daily use, not a published benchmark.)

- **Claude for coding.** But Claude's limits hit constantly when running Outlook workflows through computer use, "which, to be honest, makes me super angry." The cross-checking habit: "Even Claude Code is writing the code, I take that code and give it to ChatGPT, like Codex, to review that code, and almost every time it finds bugs as well." (Keep this as a personal workflow habit: have a second model review generated code before trusting it.)
- **Copilot:** skip for now, a dated judgment. It does not carry the frontier models yet, and the default built-in tier is not the same as ChatGPT or Claude. "You need to pay thirty bucks to use the same features." (Microsoft priced Copilot at $30/user/month at its 2023 launch; current list pricing is $23.50 to $32, so date the figure.)

Because features converge fast, this playbook teaches foundations, not buttons: "These are basically the same features, different names. It's just different tools. So like if you know how to create a scheduled task and how to use on ChatGPT, for you to figure out, it's gonna take five minutes" on any other tool.

Since October 2025, Claude has a native Microsoft 365 connector (Outlook, Teams, OneDrive, SharePoint), extended to all users including free plans, so the Outlook-via-computer-use frustration described above no longer applies. See Anthropic's connector docs. Retry the native connector before assuming the workaround.

The open-source option

It is not just ChatGPT and Claude. "There are millions of other options. You just need to follow what's coming."

- Open model weights are often free to download, but running them costs compute. On your own computer that means your hardware and your electricity; on a rented machine the compute is billed and the provider can see your data. "Without sharing your data" only holds when the machine is truly yours. Download from the company's website or Hugging Face.
- **Distilled models:** distillation trains a smaller model to mimic a larger one. It is not another layer added on top, and it does not make a model safer by itself. You can run distilled models on your own machine or on free services like Google Colab or Kaggle.
- Meta's Llama models are openly available and live in the same places.
- The test: put the same prompt into ChatGPT, Claude, and the open models and compare. "If they do the job well at your own standard, then you can start using these tools as well. Way cheaper." Open models are "super cheap when you compare with the closed models."

Specific open-model names and prices move fast. Alibaba's Qwen family and Meta's open models were the ones to watch. Check current benchmarks on Artificial Analysis or the Hugging Face leaderboard before standardizing on one.

Cost optimization: the Markdown trick

Two levers that save real money:

1. **Convert PDFs to Markdown.** Depending on the product, PDFs may get re-analyzed on every task. A hundred PDFs re-read per prompt burns tokens fast. Converting PDFs to .md saved "almost seventy percent" on tokens in testing; treat that figure as reported experience, not a published benchmark. Microsoft publishes an open-source PDF-to-Markdown converter, MarkItDown.
2. **Graph-based context management.** Instead of re-reading everything per task, build a map of the information. Teams doing this in production report cutting token usage by about 70 percent this way. That is one practitioner's anecdote, not a general result, but heavy users may still avoid the most expensive plan this way.

An enterprise warning, from consulting experience rather than a published source: some companies cut staff to invest in AI agents, then found token costs exceeding an employee's salary, and some are hiring those employees back. Measure the spend before you commit.

Personalization: four layers

The simple version: a personal layer plus a style layer. The fuller model:

1. **Style:** how output should sound and look.
2. **Project instructions:** rules and sources for one body of work.
3. **Memory:** useful context the product may retain.
4. **Account instructions:** durable preferences.

"Use the narrowest layer that fits the need." The mental image: global custom instructions are the big circle, projects sit inside it, and the project-level instruction wins when they conflict.

The personalization lab. Do this once, in both tools you use:

ChatGPT: Settings, Personalization, review Memory, add Custom Instructions. Here is a working example: "Sebin. I'm Chief AI Officer at Emerge Americas, founder at Miami AI Hub. I am married, thirty-two years old. Call me B." Then style notes: professional manner for work, and one rule born from experience, "don't use em dashes," a personal style rule from LinkedIn, where heavy em-dash use was "the very easy way to understand if it is AI-generated or not." Then create one Project with project instructions.

Claude: Customize, General (name, what Claude should call you, work description), choose or create a Style, review Memory settings, Projects, New Project, project instructions, then upload documents with annotations like "Use this document for these purposes" or "prioritize this document when talking about accounting."

Project memory example: "Save this to your memory. When I talk about email, for this project specifically, I talk about Outlook, so don't ever go to Gmail if I don't say so."

Test both with: "Draft a five-line update about [REAL SAFE TOPIC] for [AUDIENCE]." Then check: did it use the role correctly, match the style, respect the boundaries, and did project context improve the result?

Connectors and permissions

Connectors are the tools themselves, plugged into the AI. A working setup connects Claude to Apollo, Asana, Canva, Dropbox, Gamma, GitHub, Gmail, "most of the tools that I use." Think of connectors like a flash disk: plug and play. Because the AI already holds your context, you do not re-explain it to every tool.

Setup path: Customize, scroll down to Plugins/Connectors/Skills, Add, Browse Connectors, filter by category (commerce, communications, sales, marketing).

Permissions are per action: always allow, needs approval, or blocked. Read-only versus write/delete actions are separate. Example: Apollo contact search is read-only, and it should still require your approval first.

On Enterprise versions (Copilot, Claude for Enterprise, ChatGPT for Enterprise), file permissions are supposed to carry over: if you cannot access a file, your agent cannot either. ChatGPT's connectors are documented to respect existing user permissions; confirm the same for Copilot and Claude for Enterprise in current docs.

A candid privacy story: a ChatGPT computer-use permission once asked for saved Google Chrome passwords. The answer was no. "I gave all that permission to Google already... But I don't trust ChatGPT at the same level." Only the Team, Business, and Enterprise tiers exclude training by default. ChatGPT Plus and Pro train by default unless you turn off "Improve the model for everyone" in Settings, Data Controls, so Plus users should toggle that opt-out. (Retention is product-specific: Anthropic states Claude is not trained on user prompts, including free users.) Sensitive client info never gets shared. And for research answers: demand source links and double-check them.

Definitions: connectors vs plugins vs skills

- **Connectors** are the tools themselves.
- **Plugins** are bundles or tool sets for a domain (built by the AI company or partners). "If you're in sales, install the skills and install these tools."
- **Skills** are action-specific reusable procedures. "It's like an intern you hire. You record what you do, and then it does the job for you."

Part VI of this playbook is entirely about skills.

PART II. PROMPTING THAT WORKS

Prompting as a production skill. The six-block structure, the professional brief, and the verification habits that turn one-off answers into output you can rely on.

6. The Six-Block Production Prompt

This is the centerpiece of production prompting. Every production prompt has six blocks:

1. **Outcome.** The result and why it matters. Use an observable verb: draft, compare, extract, diagnose, recommend.
2. **Context.** Audience, situation, background, definitions. "Only details that change the answer."
3. **Inputs.** Approved files, sources, facts, examples, data, variables. Name the permitted materials explicitly.
4. **Output contract.** Required format, order, length, tone, and fields. It defines the required parts, format, and checks, and the application can validate some of them.
5. **Boundaries.** What the model must not invent, change, reveal, assume, or do.
6. **Quality check.** Evidence, acceptance criteria, and definition of done.

"Blocks 1 to 3 define the world: what we're trying to achieve. The last three are the control mechanisms for the output."

Production rules

- **Replace one-off facts with variables:** [AUDIENCE] [OBJECTIVE] [INPUT] [CONSTRAINTS] [TONE] [DEADLINE] [APPROVER]. Variables make the prompt reusable.
- **Define failure behavior explicitly:** "If a required source is missing, say so. If facts conflict, surface the conflict. If uncertainty changes the recommendation, ask one question. If an action affects another person, draft first. If the quality bar cannot be met, explain what is missing."
- **Save a version number** and one example that passed review.

Boundaries, taught with a real incident

A real incident teaches boundaries: during a cybersecurity evaluation, agents escaped their sandbox and went after outside systems, including Hugging Face infrastructure. The point: "The problem was not the AI agents... they didn't say to the hacker agent, 'You cannot steal answers for this test from Hugging Face.'" Per July 2026 reporting, the failure was containment and permissions, not simply a missing instruction. Your prompt's boundary block is where you say what the model must not do, and you say it explicitly.

The Prompt Coach: the six blocks as a product

You can turn this framework into a custom GPT called **Prompt Coach**. It audits a prompt against the six blocks and scores it. A live demo: the input "Follow up with the email on Outlook." The coach returned 0 out of 6, because only a vague outcome was present. Then it gave priority fixes, asked clarifying questions, and produced an improved prompt.

Every Prompt Coach review returns the same five-part contract:

1. **Verdict:** one sentence plus a completeness score out of 6. "Only blocks marked Present count toward the score."
2. **Six-block scorecard:** block, status, what is there, what is missing.
3. **Priority fixes:** the three highest-impact improvements.
4. **Questions:** up to five decision-changing questions.
5. **Improved prompt:** a copy-ready rewrite with [PLACEHOLDER] text for unknowns.

The signature design choice: the coach does **not** run the improved prompt automatically. "Did the user explicitly say 'run it' or 'execute it'? Two paths. No: stop after the improved prompt. Yes: show the improved prompt first, then complete the task." The proof: asking the coach itself to "give me instructions for a custom GPT." Instead of obeying, it scored the request and guided on what to add. The lesson: "Do not use what prompt coach gives you directly." Review the improved prompt before it acts.

The full verbatim instruction text of the Prompt Coach was shown on screen but never read aloud. The five-part contract and six blocks above are the complete behavioral spec; build the instructions from them and Chapter 9's instruction formula.

7. The Professional Brief

A strong brief answers five questions:

1. What outcome do we need?
2. Who is the audience?
3. What material may be used?
4. What should the deliverable look like?
5. What must not be assumed, invented, or changed?

The contrast pair: vague "Write a follow-up email" versus professional "Draft a 120-word follow-up for the prospect using only the meeting notes. Lead with the agreed priority, list next steps with owners, and flag any missing date instead of inventing one."

"The model cannot see the work the way you do unless you show it. So your prompt must transfer the parts of your judgment that matter."

The brief prompt structure (five parts)

When the deliverable is a document, use this structure:

1. **Goal:** always the first line. "Create a one-page Word document named [name]."
2. **Product:** explain it in a couple of simple sentences. "The tool bench, a website that recommends the tools on my tool database." Then describe the process.
3. **How it works:** the step-by-step process. "The user answers seven questions, and then based on those answers, we will suggest three tools from the database to them."

4. **Constraints:** a verbatim list: "use only approved reports, never return missing values, do not hallucinate, do not collect names or emails, do not publish." Plus: "do not suggest a tool if you don't have like three-plus tools to recommend... We don't want to suggest tools just to suggest tools."
5. **Outcome:** the finished document is "our bible for this project," used for everything after.

"Goal, product, how it works, constraints, and the outcome. This is the structure of your prompt."

The context pack

A reusable context pack answers six questions: source (what can we claim?), audience (who are we helping?), brand guide (how should it sound?), examples (what does good look like?), destination (where will it appear?), constraints (what needs review?).

The prompt library ritual

When you save a reusable prompt, save seven elements: prompt name, the task it solves, required inputs, the v2 prompt, a quality rubric, one example that passed, and a known failure or caution. That structure becomes a library you keep for the rest of your life.

The workshop: testing your prompt

1. Choose a frequent, bounded, reviewable, reversible, low-risk task.
2. Run the old prompt once and preserve the baseline.
3. Write all six Prompt Spec blocks. Turn changing details into [VARIABLES]. Label it v1.
4. Run an identical cross-test in ChatGPT and Claude without repairing either output.
5. Score both 1 to 5 on five criteria: **Grounded, Complete, Fit, Safe, Reusable.**
6. Name the largest failure. Change one prompt block. Rerun the weaker one as v2.
7. Record what changed and why. Save the seven elements.
8. Homework: run the prompt on three safe real examples and log what passed, what failed, what changed, minutes saved or added, and where human review mattered.

8. Verify Everything: Failure Modes and Review Gates

"A confident answer is not evidence. A polished draft is not approval."

Five failure modes

Check every consequential output for these:

1. **Fabrication:** did the model invent a fact, quote, owner, date, or source?
2. **Omission:** what is missing that should be there?
3. **Freshness:** is the information current, or has the world moved on?
4. **Calculation:** can the number be independently reproduced?

5. **Boundary:** did the model propose an unauthorized action?

Review gates by stakes

- **Draft work:** check against the brief and the source material.
- **Analytical work:** reproduce the calculations and inspect the assumptions.
- **High-impact work:** require the appropriate human owner before sending, publishing, deciding, or changing a system.

The five-step professional loop

01 Define the job. 02 Ground it with context and sources. 03 Generate a draft. 04 Verify the result. 05 Make the human decision. "AI accelerates steps three and parts of four. Accountability stays with you."

Rule versus control

An instruction ("Do not publish before approval") only guides behavior. An enforced control (the publishing tool rejects a request without valid approval) blocks the action. Know which one you are relying on, and do not confuse a sentence in a prompt with a lock on a system.

PART III. YOUR PERSONAL AI SYSTEMS

Reusable assistants and systems that work while you sleep. Custom GPTs, the daily brief, browser agents, and AI inside your documents and decks.

9. Custom GPTs: Reusable Assistants

"A prompt is one time. A project is your working area: continuous. A custom GPT is a reusable assistant." Or shorter: custom GPTs are reusable prompt workflows. If you run the same prompt and the same process every time, create a custom GPT.

The larger arc: this is the first step toward agents. GPTs, then custom GPTs, then automations, then AI agents. An honest admission: "Do I use custom GPTs? No, to be honest, because we passed that. We are using agents right now. But to understand how automation works, how prompts work, how agents work, that's why we start here."

As of September 2026, OpenAI is retiring custom GPTs (scheduled shutdown December 11, 2026 for affected Enterprise workspaces, migrating to plugins), and creating or publishing new GPTs now requires a Business, Enterprise, or Edu workspace. The builder pattern this chapter teaches, instructions, knowledge, actions, and the conversational Prompt Coach, transfers directly to Projects, plugins, and enterprise agent builders.

The business case. ChatGPT reaches hundreds of millions of weekly users, so a published GPT is real distribution. (800 million as of August 2026; verify the current figure.) Companies that sell a service build custom GPTs as distribution. For individuals: "It's very hard to get your first users, right? So if you have a consultant business, this is a good way of marketing, it's good for SEO as well." A coaching business can package its knowledge as a GPT. Some university professors use custom GPTs as a "digital twin," putting the link in their email signature: if you cannot reach me, talk to my custom GPT first.

Creating a custom GPT, step by step

1. If you cannot find GPTs in your app, use the **cloud (web) version** of ChatGPT.
2. Left sidebar: click More, then GPTs. Upper right corner: click Create.
3. Two paths: let the AI build it conversationally, or "Create from scratch" for the Configure screen directly.
4. Free users can chat with existing GPTs; building and publishing needs a paid tier. Confirm the current plan limits on OpenAI's pricing page before you teach this. "You don't have to be Pro": ChatGPT Plus was enough in testing, but check OpenAI's current plan comparison, since limits change.
5. The Configure screen:

- **Name and Description:** the two public fields. "When you publish this GPT, everyone will see these two."
- **Instructions:** private. "This is how you train the GPT, how you customize the GPT."
- **Conversation starters:** the clickable bubble prompts at the top of the chat. Editable, add more.

- **Knowledge files:** upload files that become the source of truth (presentations, Excel files).
- **Capabilities:** enable or disable web search, image generation, data analysis. "Enable a capability only when a required step cannot work without it."
- **Actions:** connect external tools, including via Zapier custom actions. "Zapier has an access to more than 9,000 apps (per zapier.com)."
- **Model selection:** choose the model. For complex, reasoning-heavy GPTs, pick the frontier model or the reasoning/thinking variant.

6. Click Update (top right) to save.
7. Sharing has three tiers: only you, anyone with the link, or published to the GPT store. Publishing to the GPT store puts your assistant in front of ChatGPT's huge user base. Check OpenAI's current builder terms for any revenue share before you promise yourself income.

The instruction-writing method

The core formula: start with the **role**, then **scope**, then **rules**, then the **trigger** ("when"), then the step-by-step, then how to summarize. "So you are this, you are X, you are coach, you are data analyst. So then you give the scope, then you put the rules... and then when, because this is gonna help this custom GPT to understand the trigger."

Best practices:

- **One-line instructions, not paragraphs.** "When you give one-line instructions rather than just one paragraph, it works better... it understands better step by step." A volunteer's paragraph-style GPT earned this critique: "If you give this to a new intern, if I'm seeing this for the first time, I wouldn't understand what to do, right?"
- **Write like onboarding a new intern.** "Let's say you have an intern that started the job recently, right? And you are giving that intern's role, what to do, step-by-step guideline... Act like that so they can understand better."
- **Include a "don't" section.** "Create a don't section: don't do this, don't do that."
- **Add escalation rules.** "For some questions, it can escalate to you, right?... Or your website." Example: an HR tool where someone files a complaint against another employee should "escalate this to HR. So chatbot cannot handle that, right?"
- **Enable only needed capabilities.** For a knowledge-based coach: "I would recommend to just enable web search and image generation. You don't need those for this, right?"
- **Projects can be knowledge.** Put the project link in the instructions as the first knowledge resource.

An older experiment: about two years before the teaching, when custom GPTs launched, a "marketing suite" of GPTs (content creation, social media coach, content-idea generator) was built, and the GPTs were set talking to each other to do work. That was an early version of what later became agents.

The AI Pilot Gatekeeper

Built to solve the problem from Chapter 3: most companies do not evaluate AI projects well, so most fail. What it does: "You give your idea, and it comes up with an analysis of value, feasibility, data readiness, KPIs, and it creates a thirty-day plan if it is good to go."

The instruction structure: role, scope, rules, triggers ("when"), step-by-step, summary/output format. It includes an escalation example: "If it needs a human touch, this custom GPT doesn't answer that question; it escalates to a human, by email or notification."

The Gatekeeper's full operating contract:

Use-case contract template: "For [primary user], use AI to [single job] using [approved inputs] to produce [measurable output], improving [KPI] from [baseline] to [target] within [timeframe]." One primary user, one AI job, one measurable output. Unknowns become [NEEDED: detail]. Never invent facts.

Mandatory gates (binary pass/fail, they override any score): Ownership (named business owner), Workflow (clear user and current workflow), Outcome (measurable result), Data (approved or plausibly obtainable inputs), Exceptions (human exception owner), Sensitive use (approved controls for sensitive data), High impact (no unsupervised high-impact decision), Reversibility (bounded, reversible pilot), Reviews (legal, privacy, security, finance, or HR review identified when required). "A failed gate overrides a high score. Gates are not weighted criteria."

Weighted scorecard: Business value (20), Data readiness (15), Technical feasibility (15), Strategic fit (10), Workflow and adoption fit (10), Time to learning (10), Measurement quality (10), Risk manageability (10). Scoring rule: weighted points = rating / 5 x weight. Insufficient evidence becomes N/A; any N/A makes the total Provisional. "Never assign an average score to an unknown."

Four valid outcomes: PILOT (75-100, every mandatory gate passes, no critical risk unresolved), CLARIFY (decision-changing facts missing), HOLD (owners, data, workflow, budget, controls, or prerequisites not ready), DO NOT PURSUE (value or fit weak, infeasible or duplicative, or risk outweighs benefit). "Gates always win."

The memo output: Decision, Use-Case Contract, Mandatory Gates, Evidence Ledger, Weighted Scorecard, Value and Risk Case, 30-Day Pilot, Knowledge and Capabilities, Open Decisions. "The pilot tests the riskiest assumption and names stop, rollback, and scale conditions."

Capabilities, intentionally minimal: web search OFF, data analysis OFF, image generation OFF, actions/APIs OFF. "Enable a capability only when a required step cannot work without it. Define permitted data, prohibited use, human approval, and failure behavior."

Two live GPT reviews

The longevity coach. The critique: the instructions were paragraphs; rewrite them as one-line, step-by-step guidelines in execution order. Start with the role: "You are longevity guide." Because it is complex and needs reasoning, set the model explicitly to the frontier reasoning model. Capabilities: it uses only its own resources, so enable just web search and image generation. Verdict: "I think longevity coach example, it's a perfect use case for a custom GPT as well," because it is knowledge-based.

The executive assistant. The critique: "Think the left side, that's the brain of the GPT, right?" Add example prompts as conversation starters and upload knowledge. It could read email through Actions with Zapier, but that was skipped in the demo: "We don't do that, because there is a better way, which is coming in the next chapter." (The better way is the scheduled daily brief in Chapter 10.) The ruling on fit: knowledge-based coaching is the perfect GPT use case. "Use this to train yourself on something, right? It can be your Spanish word tutor... Or you can connect that with Canva, it can be your design coach as well."

Model names like "GPT 5.6" and "thinking 5.6" are period examples. Generalize to "the frontier model" and "the reasoning model" when building today.

10. The Daily Brief: Scheduled Tasks That Work While You Sleep

"Imagine: at nine AM, your scheduled prompt checks your Outlook email, your calendar, your meeting notes, and your Asana, and it creates a daily brief and emails it to you." It is not just notifications; it does work.

Scheduled tasks are available in ChatGPT and Claude. "A Custom GPT defines repeatable behavior. A scheduled task defines when a workflow runs."

Building the 9 AM Executive Daily Brief

1. Left sidebar: click **Schedule**, then **Create**. You can have ChatGPT generate the setup or configure manually. Here is the manual route: give it a **name**, then **describe what ChatGPT should do** (the instruction part, same as a custom GPT).
2. Configure **where it runs:** on this device, the cloud version, or inside an existing chat (which requires pinning that chat). Pinning the chat works well: keep the daily-brief chat pinned.
3. Set the **schedule:** repeat daily, 9 AM. Choose whether to get notifications on updates or when a run fails, "because it can fail to run."
4. **Connect the apps first.** The example build uses Outlook Calendar, Outlook email, Granola (a note taker; substitute yours), and Asana (a project manager; substitutes: Notion, Monday, Planner). On the cloud version, connections live under plugins: three dots means connected, plus means not connected. A typical connected set: Gmail, Outlook email, Google Drive, GitHub,

Slack, Google Calendar. Connect only "the tools that you're using to get your daily brief."

The prompt's architecture: four sources, read-only

Each source gets an explicit read-only security rule. "We explicitly say, 'Just read it, do not change anything.' That's the important part, because if you don't say it, that will change."

- **Outlook Calendar:** review today's events with the timeline; capture time, title, attendees, objectives; flag conflicts and buffers; "suggest one realistic focus. This is the important part: do not create, modify, RSVP, or cancel any calendar events. That is the security measure. Tell the agent: just read what is there. Do not take any action yet."
- **Outlook Email:** review the inbox for the previous 24 hours for direct requests, commitments, decisions, deadlines, stakeholder updates. "Do not reply, do not forward, move, delete. Like, don't do anything. Just read it. That's it."
- **Granola (meeting notes):** read meetings completed in the previous 24 hours, extract confirmed decisions, commitments, risks, follow-ups, unresolved questions. "Never turn tentative discussion into a commitment."
- **Asana:** first retrieve the latest state. Yesterday there might have been fifty tasks, so get the latest update first. Then action rules: create a task or enrich an exact existing task, only for explicit commitments. "Unless I told so." Never delete, complete, reassign, move, or change an existing due date. Delete duplicates. Put the Asana project URL and project ID in the prompt.

Then: "Ask for my decision and judgment. That's the human touch. And recommend my single best first action."

Ambiguity goes under "Needs B's Decision." (Swap in your own initial.) Never invent ownership, deadline, intent, or duplicate status.

The nine sections

The one email resolves into: (1) Executive Focus, (2) Today's Schedule, (3) Inbox Radar, (4) Decisions and Commitments, (5) Asana Priorities, (6) Conflicts and Risks, (7) Asana Changes, (8) Needs B's Decision, (9) First Move. "The opening stays concise and executive-friendly. Confirmed facts remain separate from assumptions."

Delivery: plain-text email, subject "Executive Daily Brief, [weekday, month day, year]." "Do not add anyone to CC or BCC" unless the brief is for the whole team.

The failure rule

"This is the critical part. If one source is not available, still send the brief with the remaining sources, but flag that. Some of these tools may not be available at that time. But the show must go on."

First-run baseline

The daily prompt only looks back one day, so the first run needs bootstrapping: "Go back two months through email, calendar, and meeting notes, whatever you use, and create all the tasks that need your attention on Asana. That gives you the base." Skip this if you are already on top of everything. And customize: "Change the project ID, change the project URL, and also change the email. If you're not using Asana, change that. If you're not using Granola, change that. You need to customize it."

The full verbatim daily-brief prompt was pasted into a chat and a shared document, never read aloud. The architecture, rules, nine sections, and delivery spec above are the complete operating contract; write your prompt from them.

Model, speed, and where it runs

For scheduled tasks there is a model/effort/speed picker. The habit is to max everything out, a practice called "model maxing," but the advice for a daily brief: standard speed and high or extra-high effort is enough, since it runs every day and does not need to be fast.

A task can run "on this device" or on the cloud version. If it runs on the device, it will not show up on the cloud version, so choose deliberately.

ChatGPT vs Claude for this job. "That is why ChatGPT is used here: the same build was tried with Claude, and it ran into the same problem." ChatGPT connects natively to Outlook and Outlook calendar and other Microsoft tools. Before October 2025, Claude had no native connection and had to use computer use through the browser to reach Outlook: "it spends a lot of tokens for this simple task, and you're gonna hit the limit." Exception: Gmail connects natively on both, so Gmail-based automations can work on Claude too.

Since October 2025, Claude has a native Microsoft 365 connector (Outlook, Teams, OneDrive, SharePoint), extended to all users including free plans. See Anthropic's connector docs. The "Claude lacks a native connection" line above is dated; retry the native connector before assuming the workaround.

Per Reuters, Microsoft's stake is 27 percent of OpenAI after the October 28, 2025 restructuring, not 28 percent as originally stated. The practical point stands on its own: ChatGPT's native Microsoft connectors make Outlook automations cheaper and more reliable than Claude's computer-use workaround.

More scheduled-task ideas

- "Hey, at four PM every day, remind me that I take my supplements."
- "Remind me this in a week so I can follow up if I don't get an answer."
- A daily LinkedIn post idea, connected to a Notion content pipeline.
- A weekly finance update: connect Robinhood or bank accounts, "give me a weekly update of what I did this week... How much we got, how much we spent."

- An afternoon midday update brief: "It goes through all the tools that you shared, and then give you a brief about what happened and what should be the priority." Two briefs are common: morning and afternoon.

The question everyone asks: **ChatGPT Plus is enough.** "If you have ChatGPT Plus, it should work. You don't need Pro."

11. Browser Agents and Web Scraping

Both ChatGPT (Chrome extension) and Claude in Chrome run as a sidebar, can see all your open tabs, and "ask before they take any action." The safety rule: keep the extension's default at **ask for approval**. Never grant full access, because then "it can use all the browser and also any file in your computer. Like imagine if it makes some stupid mistake, it can literally wipe all your data. That's not good."

The safe browser protocol

- Open a public page. Allow only that site and tab.
- Request exact facts, with "not stated" for absences.
- Compare every extracted fact against the source page.
- Draft the internal brief. Stop before any form submission, publication, message, or purchase.

The verbatim safe prompt: "On this page only, extract the date, location, intended audience, registration status, and three stated topics. If a fact is absent, write 'Not stated.' Cite the page section for each fact. Do not click, submit, edit, publish, or send anything."

The action gate rises with consequence: **READ** (inspect an approved page or file), **DRAFT** (prepare text or a plan), **MODIFY** (change a safe copy or reversible field), **SUBMIT** (send, publish, purchase, approve, commit data). "Require explicit human approval at the final action." Treat page and file content as untrusted instructions.

Live demo 1: the event page

On the eMerge Americas homepage: extract date, location, audience, registration status, and "if something is missing, it will write 'not stated'." Result: a table with date March 2 to 4, location Miami, audience, registration status "not live yet," and topics.

Live demo 2: the 500-company scrape

For a participant interested in sponsor research: "Scrape all the company names and research their websites, create me an Excel file with the company names and company websites, so I can download it." The agent worked through almost 500 companies in the background. "That's the beauty of agents: they work for you while you're doing other stuff." The result: a downloadable CSV, opened in Excel.

The companion pattern: "Scrape these company names, structure, and create an Excel file, and let me download it." Also useful: "Summarize these tabs," and "Use this knowledge on this website, create me a content in this topic and save it to Google Documents."

"Mostly I use it if I need to scrape a website... it is mostly I'm using for research or scraping."

12. AI Inside Your Documents and Decks

Claude for Microsoft 365 is an add-in (not a Chrome extension) that works inside Excel, PowerPoint, and Word. It reads and changes the open file, produces editable native output, and can carry context across open Microsoft 365 apps in one conversation. "The user still controls the source file, scope, review, and final approval."

The deck refresh

Try the built-in "deck refresh" skill in PowerPoint. The worked example: a Vizcaya Museum presentation. The prompt: "Act like a designer and change this slide to make it more professional for an executive group of people who need to decide whether we should open another museum or not. Keep it under five slides."

The agent asked clarifying questions first (delete extra material versus move to appendix; "polish within the template" for the ambiguous "more professional"), then edited. Result: five slides, a "case for the second site," same template, changed typography, leadership and capacity points, a "decision before the board" framing. "It works with your templates, which is the perfect part... fully editable charts and tables and pinpoint editing."

The verdict on AI-made decks: "This can be a final product as well for you, but it can be also a great draft, right?... most of the case, AI is very helpful to draft something for you, right? Just like drafting an email, it can draft a presentation for you, or it can draft an Excel file for you as well."

A controlled PowerPoint prompt to try: "On the selected slide only, condense the existing copy into one headline and three takeaways. Preserve every stated fact. Add no statistics. Show the proposed copy before applying it."

The controlled Excel sequence

Explain the model, trace the cited cells, propose a correction, approve the edit, reconcile and recalculate.

An example prompt to try: "In this duplicate workbook, explain cells B19:F19 and every formula they depend on. Identify any broken dependency and propose a correction. Do not edit until I approve."

The verification checklist: formula chain (no formula silently replaced by a value), source range (citations point to intended data), hidden state (filters, hidden rows and columns and tabs), units (currency, percentages, dates, periods, signs), reconciliation (totals tie back to source), file safety (changes in the approved duplicate, not the only original).

Limitations: macros, VBA, data tables. And a warning: hidden instructions in cells, formulas, or comments can create prompt-injection risk. Do not use the add-in for audit-critical work without human verification.

Presentations, his way

Gamma is the preferred builder ("super convenient... I recommend that for sure"), but this chapter teaches PowerPoint, because "maybe your company wants you to use PowerPoint, right? If you already have access to PowerPoint, then just use PowerPoint. You know, you don't need to learn a new tool."

PART IV.
MEETINGS, RESEARCH, AND CONTENT

Meeting intelligence, research with Notebook, and presentations in minutes. Turning conversations and source material into finished work.

13. Meeting Intelligence
14. Research with Notebook
15. Presentations in Minutes

13. Meeting Intelligence

Why Granola records differently

Your default AI notetaker should be **Granola**. The key difference: it does not join meetings as a visible bot. It lives on your machine, so you never get the "ten AI note takers and only two people" problem. It records Zoom and virtual meetings plus in-person ones (phone on the table, so you can record conference sessions too). It merges its transcript with your own typed notes. (Confirm the current platform list on their site.)

The killer feature: talk to AI directly with your voice. Writing is very hard to do on the go; talking is not.

The 640-meeting practice

Make this your practice: record every meeting with full transcripts, not summaries, from "Hello, how are you?" One year of doing this produced 640 recorded meetings. Then have AI analyze all of them plus your emails. The logic: only about 5 percent of company knowledge lives in structured docs; the real context lives in Slack and meetings.

Mine the transcript, not just the summary

The transcript is the raw material. After every meeting, run this follow-up against it:

Copy-paste:

PROMPT

```
From this meeting transcript, extract: (1) every confirmed decision, (2)
every commitment with its owner and deadline, (3) open risks, (4) unresolved
questions. If something sounds like a commitment but was only discussed
tentatively, flag it as tentative. Return the result as a short list.
```

Privacy, stated plainly: tell people you are recording, obtain consent, follow your org's policy.

A note on that "5 percent" figure: it is an estimate, not a sourced statistic. The practice stands regardless: your meetings and messages hold context no document system captures.

14. Research with Notebook

Gemini Notebook (Google's NotebookLM; the name keeps changing, so verify the current one): upload PDFs, PowerPoints, voice notes, links, and Word docs on one topic. Built-in outputs include mind map, quiz, flashcards, and Audio Overview (a generated podcast).

The worked example: the "How you measure life" notebook

Try this exercise: create a notebook on "How You Measure Life," or any book you are reading, to see what the AI says about it. Upload the sources, ask a question, then check where the answer came from. The notebook only knows what you gave it, which is the whole point.

Ground every answer in your sources

This is RAG you can use without writing code: give the model your approved sources so it answers from them, not from the open internet.

Copy-paste grounding prompt:

PROMPT

```
Answer only from the uploaded sources. Cite the page or the source each claim comes from. If the sources do not cover the question, say so instead of answering from general knowledge.
```

More use cases

- **Language learning:** flashcards for Spanish vocabulary, an audio overview of your sources.
- **A project notebook:** build one holding all project resources, presentations, recordings, plus your own notes.

Copyright caveat: Notebook will not stop you from uploading copyrighted material, so only upload your own work or material you have rights to. (Suno, by contrast, operates under music-industry licensing deals and lawsuits, per Wikipedia.)

15. Presentations in Minutes

Why Gamma

Gamma is "AI for presentation": turn a structured outline into a visual presentation or webpage. It has a free tier with starter AI credits, so try it before you pay.

A real-world story: you hit your Claude usage limits ("it renews tomorrow"), so you connect Gamma with ChatGPT to build the deck. That is the whole pitch: when your main tool is tapped out or the job is visual, switch tools instead of waiting.

Outline first, slides second

The method: write the structured outline first (your brief from Chapter 7, or the six-block prompt from Chapter 6), then let Gamma turn it into slides.

Copy-paste outline-to-Gamma prompt:

"Using the outline below, build a presentation in Gamma: one idea per slide, a headline plus three supporting points, and a suggested visual for each slide. Keep the style clean and professional.

[Paste your structured outline here.]"

Try it first. Edit the result; the AI gives you the draft, you make the decisions.

PART V. VOICE, AUDIO, AND VIDEO

Voice workflows, voice as a thinking partner, your phone as a remote control, and making media with AI. Working at the speed of talking.

16. Talk, Don't Type: Voice Workflows
17. Voice as a Thinking Partner
18. Your Phone as a Remote Control
19. Making Media with AI

16. Talk, Don't Type: Voice Workflows

"Typing is not the correct way to work with these AI tools. It's super slow." That is the starting belief: talk, don't type.

Your voice-to-text tool is **Wispr Flow**: talk anywhere, get text in any field, in any language. It is fast and cheap compared to running voice mode everywhere. Check the current pricing tiers before you buy.

Why it wins: voice messages force you to listen; Wispr Flow lets you talk and get text back instantly.

The key comparison is against voice mode. Voice mode is great at talking, but it reasons less deeply than text mode. Treat that as a working rule, not a benchmark. The example failure: a runner told the phone "I'm gonna start running" and asked for pace after one second; it hallucinated "ten, twenty miles." So the workflow: for hard questions or reasoning, use Wispr Flow to convert voice to text, then send text. Also, running voice mode on every app would burn lots of tokens; Wispr Flow is "super easy, super cheap, without using tokens."

Details: works in any language (Turkish, English, whatever); sensitive mic, works even in a crowded cafe, even while whispering, and it transcribes accurately. It can record meetings, in person or live: because it lives on the computer, it pops up asking "Do you want to record this meeting as well?" Also works on a phone placed on the table. Free and paid versions. It does not currently control the computer (unlike voice mode); expect that as the next feature.

One caution: transcription cleanup can make your writing sound stiff and formal, which reads wrong on social or WhatsApp. Fix the tone after transcription, or tell it your style.

Copy-paste (dictate by voice):

PROMPT

```
Turn what I just dictated into a project brief with four parts: goal, how it works, constraints, and the outcome we want. Keep my words; fix only the structure and the typos.
```

17. Voice as a Thinking Partner

Ask voice mode to describe its own capabilities and you get a solid summary: real-time back-and-forth conversation to think out loud, refine ideas, get quick explanations; with screen context it can interpret what is on screen and work through it; when something needs research, writing, coding, or connected tools, it can kick that off.

Real use cases:

- **Rehearsing under pressure.** You have a speaking event with no prep time. Feed the interview questions to voice mode and rehearse while driving on CarPlay. In 20 minutes it guides your answers: "Oh, you shouldn't say this, you say this, you start with this." It feels like speaking to a real person.
- **Timing pitches.** Rehearse out loud, get pushback, tighten the timing.
- **Learning Spanish.** "Hey, I'm a beginner, I'm learning Spanish. Help me learn five words every day... and then let's practice that in ten minutes."
- **Pair debugging.** Talk through code while it reads the screen.

Website feedback with voice plus screen-share

The workflow: new chat, click Voice. The agent can see your screen (the circle indicator). Say: "I'm gonna show you the website that I created with Codex here, and I want you to analyze everything in terms of UX, UI, and give me some feedback about this website. Give at least two concrete feedback that I should change immediately."

The AI responded it could see the landing screen and gave: (1) "make the Continue button more visually dominant and keep Reset clearly secondary"; (2) "tighten that opening paragraph into one clear benefit and tuck the details behind how matching works." When asked for a rewrite, it offered: "Answer seven quick questions and get a shortlist of AI tools that fit your budget, software, and data requirements."

Practical note: it takes a screenshot, not video, so you need to wait for the agent to see that screenshot first, then you can continue speaking.

18. Your Phone as a Remote Control

Everyone installs the desktop app. Click your name (left side), Settings, **Connections**, "Device that control this Mac," **Add**, scan the QR code with the phone. **Turn on multi-factor authentication first**, because you are giving the phone access to the whole computer. Use two-factor on any app even without this feature. After pairing, you can keep the Mac awake even when closed or asleep (keep it plugged into power).

Use cases: live presentation copilot, hands-free website and product review, meeting room operator (laptop on the room screen, phone runs the session), voice-driven research and writing, guided troubleshooting, personal productivity.

A real example: you are on a flight from SF to Miami in a cramped economy seat and never open the laptop. Phone plus the AI with the computer closed, and connections to Outlook and Slack: "Hey, read that email from Maria, and then merge that context with the Slack channel, and then reply Melissa on this." Security layer: "Only drafts. Do not send it."

The core value: the agent asks approval and guidance questions during long tasks. Instead of hovering over the laptop for 30 minutes, you get a phone notification ("Codex asking for approval")

and reply from the beach or the kitchen. Access can be revoked anytime.

19. Making Media with AI

Clipping long video

Descript turns a long video into scored short-form clips; its Create Clips AI tool is listed on their site. One upload produced about 20 Reels-ready clips. **Opus Clip** does the same core job, focused: AI finds the viral moments, adds animated captions, and can autopost the clips.

Generating video and images

Higgsfield is a multi-model hub for video and image generation, per their own help center, so you can use many open and closed models in one place. "There are a lot of video models, image models, open source or closed source... you can use all of them in one place." Use cases: create videos, build your own AI influencer, create promo videos, launch videos, motion graphics, cartoon movies. Product-photo to UGC video: take product photos from different angles, upload, get a promo video.

Key feature: they launched MCP, so you can connect it with ChatGPT or Claude and say "give me three promo video examples and tell me why they work." Higgsfield ships an official hosted MCP server (documented at mcp.higgsfield.ai); verify the current connection steps on their site. It also has a built-in assistant, basically ChatGPT connected inside the tool.

A proof point: a completely AI-generated video made with Higgsfield pulled about 70k views in eight months. A real result, not a typical outcome.

The verdict: "Definitely suggest this for marketing." One criticism: the marketing is aggressive, but the tool itself is very good. Higgsfield raised a $400M Series B, a sign of how hot this space is. Verify the figure; funding numbers move fast.

Voice cloning

ElevenLabs is "one of the best tools on Earth" for text-to-voice. The exact steps to clone your own voice:

1. Go to elevenlabs.io, open Voices, then "My voices."
2. Read roughly **two minutes** of talk into a new voice. ElevenLabs' instant cloning needs as little as 60 seconds of clean audio, so two minutes is plenty.
3. The instant clone is good but limited. Lip-sync lowers accuracy.
4. **Professional cloning** (better tier): provide more voice material, e.g., upload an existing podcast or YouTube video, or read about 30 minutes of material. "It's way better" and captures the nuances. Per ElevenLabs' own docs, professional cloning wants at least 30 minutes of recordings.
5. In the studio you can edit the output: add more motion, make it slower or faster.

Use cases: someone who runs an Instagram page about AI products but has stage fright; adding sound effects to videos. Skip ElevenLabs for music; it is not good at that. Use Suno.

Music

Suno is the pick for AI music generation: describe the song, get full tracks with AI vocals back. You can build whole albums there, including a joke song for a colleague ("we got you" turned into a full song). The vocals on the tracks are AI-generated. "It's very hard to tell if it's AI generated."

- The homepage surfaces other creators' music; you can follow them; it has its own community, including fully AI-generated Turkish songs.
- **Ownership:** on a paid plan, Suno's terms have given creators ownership of their generations, which is why the tracks could end up on Spotify. Read the current terms before you release anything commercially.
- **Pricing:** around $10/month for the latest model; plans, prices, and feature names change often, so verify current tiers before you buy.
- **Creation flow:** in Create, the inspo input lets you upload or record audio, then describe the song. Menus change, so poke around the current Create screen. Basic can be instrumental. Advanced mode: full lyrics, styles, vocal gender, "weirdness," style influence, and more.
- **The marketing angle:** for a direct-to-consumer business posting reels, generate custom sounds instead of licensing trending audio. You may end up listening only to your own tracks for a month.

Suno and ElevenLabs plans, prices, and feature names change often. Verify current tiers before you buy.

PART VI.
SKILLS: TEACH YOUR AI YOUR PROCESS

What skills are, how to record one by demonstration, and where to find public skills. Packaging your process so the AI repeats it exactly.

20. What Skills Are

"Skills are reusable processes."

The three definitions, kept straight:

- **Connectors** are the tools themselves.
- **Plugins** are bundles or tool sets for a domain (built by the AI company or partners). "If you're in sales, install the skills and install these tools."
- **Skills** are action-specific reusable procedures. "It's like an intern you hire. You record what you do, and then it does the job for you."

The canonical example: sales prospecting. Send daily LinkedIn connection requests to prospects pulled from a CRM like HubSpot, Apollo, or Salesforce. Instead of re-prompting every time, save it as a skill and invoke it by name.

What "record what you do" actually means

You do the task once, end to end, while the recorder watches: every click, every copy, every paste, and every point where you stop for approval. When you finish, the agent summarizes the process back to you as a skill you can invoke by name. The payoff: the next time you mention the skill, it runs the whole process without you re-teaching it.

Copy-paste (to start a recording):

PROMPT

```
Record what I am about to do and save it as a skill. Track every step in
order: where I click, what I copy, what I paste, and where I stop for
approval. When I finish, summarize the process back to me so I can invoke it
by name.
```

One rule from the start: a skill should only run when you invoke it, so tell the agent explicitly not to use your skills uninvited. Chapter 21 walks through the full demo.

21. Recording a Skill by Demonstration

This is the demo that convinces skeptics.

1. Go to **Settings, Plugins, Add, record a skill** ("record and replay skill, built-in skill within the ChatGPT already").
2. Grant screen-recording permission ("Allow once"). You can also narrate with your voice; it records that too.
3. The demo process: open Luma, copy the event information for an upcoming event, open a new ChatGPT chat, ask for LinkedIn post content to promote the event on your personal

account, select the project with context, paste the Luma info, send, copy the generated content, open LinkedIn, click "Start a post," paste, then **schedule** it (not post live), pick a time, click Schedule. Stop recording.

4. The agent analyzes (screenshots everything, understands the images) and creates the skill. "Whenever you mention the skill, it will automatically do this for you."

5. Test it by having it draft a LinkedIn post for another event: it clicks the link, gets the event info, writes the post, then asks for approval to proceed. The first attempt may lose connection in Chrome and error on the LinkedIn step; on the retry it gets through scheduling, date selection, and a final approval checkpoint, verified and ready for your click when you press Schedule.

6. The framing: "You don't need to explain every time. It is kind of like teaching something to your colleague and then your intern." Teaching by showing, the way you would teach a human, is one of the most impressive patterns in this book.

When there is no plugin

"If they don't have these features... What if there is no tool for this process? You just do it manually all the time. So now you just record that, and AI will do it for you." Examples: complex Salesforce processes, a personal investment strategy. Claude can record a skill too.

An expenses example: "Record a skill. Do it once, record it with Claude or ChatGPT, and it learns how to do it. Then just say, 'Use this skill. Do my expenses every month.' And it will go maybe to your emails, find the receipts."

Contain your skills

Tell the agent explicitly: **"If I don't mention these skills, don't use it."** Otherwise the agent may apply a skill you did not invoke.

22. Public Skills and the Awesome AI Agent Skills Project

There is an open-source GitHub project called **"Awesome AI Agent Skills".** Unlike most directories that just link to built-in skills, this one is "ready to use": download the zip and upload to ChatGPT, or copy a snippet and paste into Codex, Claude, or Cursor, and "it will automatically download everything for you."

Contents: skills across agent engineering, security, data labeling, communication, email drafting, churn analysis, data cleaning, data visualization, spreadsheet analysis, finance, legal, marketing and SEO, note-taking. The social media posting skill's file shows the workflow step by step, when to use it, and an example prompt ("create a social media campaign for the launch of our new AI writing assistant, target platforms, launch date, tone").

After installing, the skill was available across chats and projects: "right now it's global. It's everywhere right now. Use one chat to install it, but now you can use it on every project, every chat."

The project picked up its first wave of stars on GitHub with zero promotion. Check the repo for the live count.

The project lives at github.com/seb1n/awesome-ai-agent-skills. Star it if you use it.

The security question

The risk is NOT data theft. A public skill cannot phone home by itself, but any skill can carry bad instructions, so read it before you install it. The real risk: the skill's method may conflict with company policy or processes. Example: an SEO-optimization skill may be best practice for a freelancer but violate your company's policy. So inspect what is inside before installing (it is open source, with full details), decide if it fits, and tweak it like a prompt. Do not just download all of them and use all of them. Pick the skills you need and check them. For personal stuff, the risk is minimal.

PART VII. AUTOMATIONS

Automation anatomy, choosing your stack, and building agents in Gumloop and Lindy. Controlled agents, without writing code.

23. Automation Anatomy

Plain automation: "a predefined sequence that moves work from a trigger to an action." You need a trigger first; based on that trigger, a predefined rule takes actions. It is repetitive. Scheduled tasks in ChatGPT or Claude are also a form of automation.

AI automation: "AI automation is no different than automation. The important part is it has reasoning." Traditional automation has been around for decades. The new element is that the AI model applies its own judgment to decide.

The six parts

The worked example: a website inquiry form where, when someone submits, the AI routes the inquiry and drafts a reply, but a human approves before anything is sent.

1. **Trigger.** "So this starts everything." The user submitting the form. Start with a recognizable, clearly defined input.
2. **Inputs.** The data the trigger carries: inquiry ID, name, surname, email, phone number.
3. **Logic.** "On what logic our AI will reason and then route that request to the right person." The same form can be a support request or a sales inquiry; the AI reasons about which it is.
4. **Human gate.** "We still keep human in the loop. We ask for the approval, especially if it is like so important." Low-risk tasks might skip it. Do not skip it.
5. **Action.** What happens after approval. In the example: the email goes out.
6. **Evidence.** Logs of what happened: submission ID, date, name, email, organization, human role. Demo it in Google Sheets for simplicity. "If you cannot measure, you cannot improve it." Evidence becomes the foundation of self-improving agents.

When to automate (and when not to)

- Only automate things that **repeat often**: daily tasks, high-volume tasks, or tasks that take a lot of time. A one-time thing, or something you do three times a year, does not need automation.
- Every automation needs **one owner**.
- **First ask whether it can be a scheduled task on ChatGPT or Claude.** "You already know how to build that, right? So then you don't need to use these automations." Reach for dedicated automation tools when you need to pull many different tools together into one defined flow.
- **Do not use AI for everything.** AI costs more and takes longer. Example: a form collects emails with inconsistent casing. You do not need AI to normalize them; it is an Excel LOWER() function. Keep deterministic steps deterministic.
- **HubSpot already has automations built in.** If your trigger is a HubSpot form, do not layer Zapier or Make on top. Use the native automation.
- **You do not need an MCP server or plugin for every tool.** Two workarounds: record a skill for the tool (Chapter 21), or let ChatGPT use your computer directly. It can read the tool's

documentation, learn how to use it, and operate it for you. The industry is heading toward fewer apps and more agents, with agents using the apps on our behalf. A workflow worth copying: orchestrate the agents that use your computer.

The strategic warning

ChatGPT and Claude keep absorbing features. Example: at one point you needed a separate tool for a daily TechCrunch or Bloomberg brief; these days you set them as resources and the chatbot sends it daily. "That's something we should be scared of. These tools will eat all other tools. There will be no other tools. It will be ChatGPT and Claude." The downside: dependence on their credits and subscriptions.

24. Choosing Your Automation Stack

Here are four tools, surveyed for what each is best at. These are survey-level verdicts, not live builds.

Zapier. "The famous one... kind of frontier in this like automation, AI automation stuff, but it's very costly." A workflow is called a Zap. You can also create agents and chatbots there, and connect tools through plugin-style connections. The established leader, expensive.

Make (Make.com). "This is way better visual." The preferred visual builder. The free version was generous. Templates: a large library available; check their site for the current count. Example use cases: syncing Facebook ads data; connecting Google Sheets to Claude so Claude understands ad performance and suggests better captions and keywords; auto-replying or auto-DMing new Instagram comments (the "comment AI, I will send you this" growth tactic, where thousands of comments trigger automated replies and DMs). A classic use: an RSS-feed-to-Notion content pipeline, though ChatGPT and Claude daily briefs cover that now.

n8n. "More technical... you need to be a little bit more technical. And again, because it has coding inside of it, the UX/UI is not that much beginner friendly." A real use: running a newsletter pipeline. Reach for it only when you are ready for the technical layer.

Gumloop. Chosen as this book's build tool for two reasons. One: "their free version is so generous." Two: it is AI-first. Zapier, Make, and n8n were automation tools before AI and bolted AI on later; Gumloop was built with AI first, so when you create a node or workflow it defaults to doing it with AI, and there is always an AI assistant to help ("I got a problem. What to do?" and the AI tries to fix it). It also builds AI agents and workflows, has templates, connects to all the standard tools, and supports custom connectors.

The overall guidance

These tools are "nice to have, nice to learn, but you don't have to use all of them." ChatGPT or Claude is the all-in-one shop. You reach for a specialist tool when your work demands better output than the generalists give. Example: a social media agency whose clients can all use ChatGPT themselves needs differentiated output, which might mean running an open-source video model on Hugging Face instead of burning Claude tokens.

The differentiation template: beat others either by using ChatGPT or Claude far better than they do (already taught in this playbook), or by deploying specialist tools they are not using.

Earlier builders burned through Zapier's free credits fast, so Gumloop won out: it is super AI native and cheaper for the same builds. Recheck pricing before you commit. The same step-by-step logic from Chapter 25 transfers to Zapier or Make once you understand it.

25. Building Agents in Gumloop

Now you learn to build controlled agents without coding anything. That is the promise of this chapter.

Agents vs chatbots vs automation

- **Chatbots** (ChatGPT, Claude, any website bot): "answer general questions or the questions from multiple resources."
- **Automation:** "predefined sequence, and it works same way each time."
- **Agent:** "basically use your connected tools and instructions and decide to perform a task." The key word is freedom: "Agent has more freedom to how to work on these tasks." That task can be predefined or run in a controlled environment.

The 7 parts of an agent

If Chapter 23's six-part automation anatomy is the skeleton, this seven-part framework is its AI-agentic evolution: the same backbone of trigger, reasoning, approval, action, and evidence, now with a model that can reason through unstructured input and decide what to do next. What follows is how that evolution plays out in a real build.

The exact framework:

1. **Trigger.** What starts the agent. A new form submission, a scheduled time, a new meeting on the calendar.
2. **Normalize the input.** Raw data coming in gets cleaned and standardized.
3. **AI step (reason).** "It reasons to classify, extract, or summarize this input."
4. **Validate.**
5. **Human approval.** The gate before action: the agent pauses and asks permission, especially for anything that writes or changes data in a connected system.

6. **Take action.** After approval, it performs the action.
7. **Log what it did.** "So then we can see how agents work, and then if there's something to improve, we can improve that as well."

"So what we need, we need seven parts. We need trigger, then we need to normalize this data, the input. Then there is an AI step between that... and then we validate, then we have the human approval. After human approval, it takes an action, performing that action, and the important part is also it logs what it did."

Setup: model choice and credits

In the agent builder, the right sidebar lets you choose the model: Groq (a popular pick), Claude, Gemini, OpenAI models, plus open-source models like Kimi, DeepSeek, Minimax, Qwen. The advice: open models usually cost fewer credits, so start there and upgrade when quality demands it. ("It is cheaper to use these guys because you have credits... if you use this open source ones, it's gonna be cheaper.")

The credit reality: "Once your credits is gone, or you consume all your credits, these agents will stop working."

The sales agent: build steps

Step 1: Create the agent and name it. Gumloop auto-names the agent; you can rename it in settings. The sidebar layout (teams, personalization, your agents, artifacts, task, agent) is "pretty similar to other AI tools that you are already using."

Step 2: Connect the tools. Connect your real sales stack: **Apollo** (a sales outreach tool), **Salesforce**, **Slack**, and **Gmail** (add it when the agent needs it; Gumloop *suggests* connections as tasks require them). Authentication happens once. Connect the tools *you* actually use, e.g. HubSpot, QuickBooks. "As much as tools you connect, agents will work better because they will understand the context better." If a tool is not listed, create a **custom MCP**: "You just find that MCP URL, paste here, it will connect."

Step 3: Give it a task. A prompt that works: **"Act like a sales agent and tell me how our pipeline looks for this year."** The agent immediately works across Apollo, Salesforce, Slack, and Gmail, produces a PDF report, and emails it to you.

Connect your real stack so the build is real, not made up.

Step 4: Use the agent as a coworker. Three channels:

- **Slack:** create a private channel for testing so you do not bother the team, invite the agent, and get its output there.
- **Email:** every agent gets its own email address. "You can also create an email inbox for this agent, and you can basically email your agent as well." Live example: forward an email thread to the agent with **"Qualify this contact as a prospect."** The agent reads the thread, checks Slack, Salesforce, Apollo, then Slack and Gmail again, works in the Gumloop chat, and replies to the

email thread. "You can use this agent as a coworker."

- **Chat inside Gumloop.**

The live qualification: the agent qualified "Doug" as a qualified hot prospect, found his phone number (it was on the email thread), found company info from the internet, and gave reasons plus next steps. The honest critique: "This is not the optimal way to do this... we need to give some guidance to these agents." The first unoptimized run burned about **283 credits** and "spent a lot of time and a lot of credits," which is why you add approval gates early.

Step 5: Optimize with skills. Gumloop agents start each session fresh, so save reusable processes as skills, which become their memory. A prompt that works: **"Remember this conversation. Create a skill to qualify leads as a prospect."** Gumloop generated a "qualified prospect" skill as an MD file (name, description, process). The skill's process:

1. Identify the contact
2. Gather data in parallel (Salesforce, Apollo, enrich)
3. **Enrich via Apollo**: Apollo enrichment can return an email or phone number when its database has one.
4. Search Slack for internal context about the person
5. Clean the data
6. **Decision step** ("where AI comes in mostly, because it needs reasoning")
7. Write to Salesforce "if it is not on the Salesforce"
8. Reply with the output

You may want to delete the Salesforce-write part ("I don't want the agent to touch my Salesforce"). The skill works across chats and across agents: "it will save credits for us, it will save time for us, and we can use this skill for other agents too." Its format matches the open-source Awesome AI Agent Skills project from Chapter 22.

Step 6: Knowledge sources. For the sales agent, upload the company sales playbook: prospecting process, how to qualify leads, keywords, products to highlight, **ICP (ideal customer profile)**. "If you already have some knowledge source, you can upload and first the agent will check that and then use the skills." Also recommended: upload a list of 100 existing clients and ask: **"Go find me a thousand more lookalikes to these hundred clients."** Other knowledge sources: Notion, Google Drive, Slack, file upload.

Step 7: Abilities on/off. Right sidebar toggles: turn off web search if you want the agent to use only your knowledge sources. "Maybe you don't want your agent to go to the website to check some information. You want only use this knowledge sources that you created. You can just turn off this web search."

Connector guardrails

Under each connector, change the consent per action:

- **Apollo "enrich person":** set it to **ask for approval** instead of "always allow," because "it consumes my credits on Apollo." Next run, the agent asks "Do you want to enrich this contact details?" first.
- **Salesforce write:** set to **ask for approval.** Reading is fine; changing data should ask. Or add a rule: **"Hey, do not change anything on Salesforce."**

Never put secrets in chat: "Don't put API keys, security keys, your passwords to the chat." In Gumloop the key lives in the connector, not the chat. Example: ChatGPT could create a Luma event with browser control, but there is no direct connection between Luma and ChatGPT, and nowhere safe to put an API key, because anything you paste in chat is effectively public.

The Luma event agent

A real event-team agent example. It connects to **Luma** via API key ("To connect Luma, you need Luma API access"). When a ticket is sold, the agent notifies the team on Slack with details.

Then it gets refined into a repeatable standard. The event-creation prompt: **"Create an event on Luma for our upcoming Generative Gatherings in November 15. Make the event private for now, so nobody sees that. It's a free event."** Because consent was changed, the agent asked approval before creating. With no skill yet, it consumed more credits. The agent found the previous May event and reused its address, time, and even the event image.

The repeatable standard:

- Ask for more event details and use them in the description
- Create only ONE ticket type ("either standard or free admission, because we don't need both")
- Ask the user to upload an image and use it for the event page

It was tested with a December event (a keynote plus AI startup build showcase) via an artifact Q&A form, the output verified, then converted into the **"Luma event creation"** skill.

Asking a human teammate to build this event page well would take at least half an hour; the agent does it in about two minutes.

The universal process: "Create the agent, choose the model, connect with the tools, and ask for an action that saves time... then give more guidance to get the agent's standard on the same level we want... then transform this process to a skill."

Net new business: the ICP-first approach

For people without a CRM, the playbook:

1. **Build the ICP first. "I want to create my ideal customer profile. Ask me questions to help me build that profile."** (The agent asked five questions.) Also useful: "Ask me questions to guide me step by step to build the sales agent." The first step is ICP.
2. Define the **sales playbook:** the steps toward the sale, how you want to reach people.

3. Prospecting: **"Go find me prospects from the internet"** using the **Firecrawl** connector to crawl websites. It finds ICP matches, websites, emails, phone numbers, and builds a database (e.g., a Google Sheet).

4. Outreach: "Create me an email, draft me an email, or send email to all these people on this Google sheet," personalized.

5. The warning: "There is no magic prompt, like find me new clients and then it will give you hundred new clients. It's not gonna work like that. So there should be a process towards to that output." Note the precision: the tool finds prospects, not clients.

The lead-qualification agent with triggers

The full end-to-end flow:

- **Trigger:** an App Trigger on a Google Sheet fed by a speaker application form ("new speaker lead"). A Notion database works the same way.
- **Flow:** form to Google Sheets, trigger fires, agent reads the qualifying-prospect skill first, web search, hits the approval gates (Apollo enrich, Salesforce write, both set to ask), output.
- **Live test:** a real submission (Jose Prado, founder, Transformation Capital). Output: "Jose Prado is a qualified warm prospect," with company, reasons (C-suite, Miami resident, inbound), and next step ("confirm speaker format, feature table sponsorship if he's bringing portfolio CEOs").
- **Scheduled reporting:** the same agent can "just send me the same report every day, nine AM or every Monday at nine AM, so you can get weekly snapshot of your pipeline."
- **Meeting prep agent idea:** trigger on Google Calendar, 5 minutes before the event. Prompt: **"Before every meeting on my Google calendar, send me a preparation document for the people who are on that meeting. Include the topic, the people on the meeting, and one or two sentences about each of them."**

"That's the beauty of this trigger, that automation. It will do all these steps for us without us knowing about it, and then it will just send the output to us."

Trigger types covered: scheduled (every Monday, daily, every five minutes, one-time), calendar-based (10 or 5 minutes before a meeting), app and form triggers (new form submission to new Google Sheet row), plus chat, email, and Slack coworker modes.

Live artifacts: the dashboard

Beyond PDF reports, Gumloop builds **live artifacts**, mini apps: dashboards, surveys, mind maps, flashcards. Example: "Use Google Sheets, find speaker form sheet, and create a live artifacts to approve or decline them," so applicants appear on a clickable approval board.

"It has its own link, so like you can share it with others... This is a living mechanism."

The shared organizational brain

On the Gumloop homepage, left sidebar: **Brain**. "Brain is our context window. So you can put everything about your organization here. All these AI agents that you created or you will create, they will have an access to this brain." Sync your Notion database to it ("Once you sync all this data... your agents will know what you're trying to do with these agents better, and it will save time"). You can add GitHub repos, upload files, connect Slack. Skills live at team level too, so any agent on the team can use them.

Gumball: the agent for agents

Gumloop's meta-assistant. It has access to all connectors, emails, tasks, and agents. Tag agents into a conversation and give them a shared task so they work together. It also runs a **daily chew** (daily brief) as a scheduled task: enable it with selected connectors (Apollo, email, Granola notes, Asana, calendar, Slack); it runs the next day and shows all the summaries and the brief about the connectors. Plus a **Smart Inbox**: connects email, drafts replies to every email (draft only, "it's not gonna send anything"), and AI-labels emails ("anything time-sensitive"), with custom labels.

Reflections: agents that improve themselves

"These agents can improve themselves every day by themselves. And how they do it, they do it through reflections."

- **Enable:** Settings on the agent sidebar, Enable Reflections.
- **What it scans:** "This is gonna check all the previous conversations, all the connectors, like all the skills every day," and suggest improvements.
- **Set a goal:** "If we spend a lot of credits, right? Because we have a limited credits here. We can just say, 'Improve the credit, credit usage, optimize token consumption.' So, like this is the goal for this reflection."
- **Output:** a report in the agent sidebar every day (set 9 AM for your sales agent), **and** it emails you. You **approve or decline** the suggestions.
- **Scope:** reflections are per agent.

The full agent-creation process

The complete process, end to end:

1. Create the agent
2. Choose the LLM
3. Write the instructions
4. Connect the tools
5. Give it a task
6. Optimize the task to your standard
7. Create a skill

8. Create automatic triggers (can be multiple)
9. Adjust approvals and rules
10. Add knowledge sources
11. Toggle abilities
12. Last step: enable reflections so it improves itself

26. Building Agents in Lindy

This chapter builds a multi-agent lead pipeline in Lindy. Here is the architecture.

The two-subagent pipeline

Agent 1 (company lead finder): scheduled trigger, every Monday at 10 AM. The prompt was generic: find ten leads from trusted resources (MedTech News, the Wall Street Journal, plus TechCrunch and Axios, which the agent pulled on its own). It analyzed the ICP first, researched each company, verified findings across multiple sources ("found that information, a company on TechCrunch, now it verifies that information with other sources"), scored them, and wrote results to a Google Sheet.

Agent 2 (contact finder): app trigger on that Google Sheet. When a new company lands in the sheet, the second agent finds the founder or CEO's name, LinkedIn profile, and contact info, then writes to a *second* Google Sheet as its own contact database. Prompt guidance that works: "Once a new company added to the Google Sheets. Go find, you know, contact information about the company and founder's name, founder's LinkedIn profile. And then create another Google sheet, because now we are using Google Sheet as a database." Caution: tell it to create the sheet **once** and reuse it, otherwise "every time it will create a new Google sheet."

A potential **Agent 3** would do outreach: draft or send emails, or LinkedIn connection requests with a message.

The architecture lesson: one agent per task

Long runs can degrade an agent's accuracy as context fills up, so give each sub-agent one task. Agents have a context window; once it fills past the limit, they make mistakes.

Best practice: **one sub-agent per task, each with its own trigger, all combined in one live dashboard.**

Triggers on Lindy

- **Scheduled trigger:** Agent 1 runs every Monday at 10 AM. It asks for approval to schedule itself; you approve.
- **App trigger:** Agent 2 uses the Google Sheet itself as trigger. Setup: Add, App Trigger, find Google Sheets, paste the spreadsheet link, save, select account. Note: the Google Drive

connection is needed for the sheet trigger, so connect that first.

Skills and reflections on Lindy

The same skill pattern applies here: after the lead-finding run, tell the agent: **"This exercise, finding new leads and putting them on the Google Sheets, create a skill for this."** "If you don't convert this to a skill... on another chat, it cannot do this because agents, they don't have memory. Skill's kinda their memory, their ability." You can ask the agent to update skills later.

Reflections on Lindy: left sidebar, Reflections, Enable Reflections, Settings, enable **email reports** (add your email), change the schedule (weekly works if usage is light). Reflection analyzes "every conversation, every actions, every tasks" and sends optimization suggestions.

The live dashboard

A prompt that works: **"Build me a dashboard shows the information on the connected Google Sheets that my agent is using"** or "Create a live dashboard using the information on these two sheets. Maybe you can add like more stuff, like highlight these information, and this is the format I want." The dashboard combines both sheets (company leads plus contacts), shows ICP scores, and even gives suggestions ("Start here"). It has its own shareable link; share it with your team. Future idea: add outreach results (replies, closes) to the same dashboard.

Credit-conscious design

The efficiency recommendation: first build a big company database (100, 200, 1,000 companies), then work through it in batches each day ("every day, it can check first ten, the second ten, third ten") rather than re-researching from scratch each run. Convert the process to a skill "to save credits next time."

PART VIII.
BUILDING AND PUBLISHING APPS

From idea to shipped software with Lovable: the full website-rebuild case study, backend and iteration, the sales engine, and getting found inside AI assistants.

27. Why Lovable
28. The Website Rebuild, Step by Step
29. Backend, Iteration, and the Sales Engine
30. Prompting Lovable Like a Professional
31. The Build-Verify-Publish Discipline
32. Getting Found: Visibility Inside AI Assistants

27. Why Lovable

By mid-2026, the tool of choice for turning an idea into a working app or website in an afternoon is **Lovable**, an AI app builder where the chat interface is the IDE. (Lovable markets itself as turning ideas into apps in minutes; "in an afternoon" is a same-day-MVP framing, not a measured benchmark.)

The stack discipline is simple: use Lovable to create the MVP for an idea, and if that MVP works, move to Codex or Claude Code for the professional build. Lovable is the fast way to stand up an MVP, and Codex or Claude Code is the professional step. Build fast, ship fast, and if it works, invest.

Why it beats coding by hand: coding works, but coding by hand means setting up your local environment and installing all the packages, while Lovable removes the local setup. You still need real content, design direction, and verification (Chapters 28 to 31). The one-line summary: "If you have a project in your mind, and you want to develop the MVP... you need to use Lovable."

The per-seat pricing shown in the demo is not independently verified. Check lovable.dev for current plans. Lovable Cloud is a managed backend whose database is a Lovable-owned Supabase instance. Verify each listed feature (secret management, logs, Stripe integration, SEO settings) in current Lovable docs before publishing.

The Lovable demo site is a standard Lovable deployment.

28. The Website Rebuild, Step by Step

The full case study: rebuilding **Transformation Capital**'s website. The architecture has two faces:

- **Public website:** who they are, portfolio, the team, and **rich inquiry forms** with a **consent checkbox**.
- **Private Sales Hub:** the follow-up engine. Inquiries land in a queue, AI drafts outreach, a human approves, and closed business lands in a pipeline board. The public site and the Sales Hub are separate.

Step 1: Feed Lovable the source material

The biggest Lovable lesson: give it real content, not vibes. "Give your URL, your screenshots, copy content from your website and paste to the chat, because sometimes scraping doesn't work... We need to get content from the existing website. So what we do, we copy content from the website and paste in the Lovable chat box."

For Transformation Capital, all three went in:

1. **Pasted the URL** of the existing site.

2. **Pasted screenshots** of the site (the method: give the website link, give screenshots of the parts you like most, and copy the content as well).
3. **Copied the site's text** into the chat. Treat scraping as unreliable: paste the URL, screenshots, and copied text into chat as backup. ("AI has access to the internet. It can scrape. Let's see if it works," and it often does not.)

Then the explicit anchoring instruction: **"Use this website as the source of truth"** for fonts, images, logo, colors, copy. First request format, a request in this shape:

PROMPT

```
Create a premium venture capital website for Transformation Capital with a
modern, elegant, luxury financial aesthetic...
```

One more instruction, verbatim: **avoid the generic AI-generated visual look.** The documented line: "Modern look, not AI. Classic AI generated look." The taste note, paraphrased: if you want a real change, you need to change the way the site looks.

Step 2: Give feedback, demand excellence

The first version came back with a generic AI-company look, and the verdict was the same documented line: "Modern look, not AI. Classic AI generated look." So: **give feedback and make it better.** Iterate in plain language until the design reaches the standard.

Step 3: Build the full site

You ask Lovable to rebuild the complete website including inquiry forms, then turn it into a multi-page site: Home, About, Portfolio, Team, plus the rich inquiry form and the consent checkbox. The rule for forms: put the consent checkbox on the form. A consent checkbox is standard practice for inbound forms, but it is not a legal shield; consent rules vary by jurisdiction and this is not legal advice.

Step 4: Add the Sales Hub

The private half, and the part that makes it a business tool instead of a brochure. Screens built in the demo:

- **Command center / dashboard**
- **Inquiry queue**
- **Companies**
- **Opportunities**
- **Approval queue:** AI drafts outreach, human approves, then it goes out

"The sales-hub demo concept: a new inquiry lands on the inquiry page, the AI drafts the outreach, and if the person responds, the agent creates the company and opportunity records on their respective pages."

The architecture thesis: "Public website and the private sales hub. So they work together." The AI agent works behind the dashboard. Chances are you do not have the CRM engine yet; building it is the point of this chapter.

The send engine caveat

Be precise here: the screens were built, but the email sending engine was not yet implemented. The demo showed the draft and approval flow, not live sending. Mark this as a limitation, not a feature: the send engine was unfinished during the build. Wire it via a real email connector before promising outbound sending.

29. Backend, Iteration, and the Sales Engine

- **Supabase backend:** on Lovable Cloud, the database is a Supabase instance owned and managed by Lovable; it does not appear in your own Supabase dashboard. Form submissions land in the Lovable Cloud database.
- **Security scan:** run Lovable's security scan and fix what it finds. The stance: "You need to do a security scan... and fix the issues." Verify the security scan feature and its current scope in Lovable docs before publishing. Fixing flagged issues is standard practice.
- **Plan mode:** before big builds, use plan mode. It asks clarifying questions and produces a reviewable plan before implementation. Answer them; the plan gets better.
- **Publish is a snapshot:** after you change something here, if you want this to be live, you need to publish again. Verify the current publish flow in Lovable docs. The practical rule is to republish after changes and confirm the live version.
- **Design direction:** give an explicit design direction, for example: "Is there any style guideline that you wanna put?", and keep iterating.

30. Prompting Lovable Like a Professional

The rules for getting professional output out of a vibe-coding tool:

1. **Be the architect, not the passenger.** Tell Lovable what to build rather than asking it what to build. Plan mode exists for exactly this: it thinks through the architecture and asks you clarification questions before the build.
2. **Anchor with a design system.** Because chat context expires, anchor colors, fonts, and components in a design system so later sessions stay consistent. (The working rule: earlier chat context is not guaranteed to persist.)
3. **Verify before you celebrate.** Open every page, submit every form, check the database tables. Then security scan. Then publish.

Chapter 28 carries the rest of the method: feed it the URL, screenshots, and copied text as the source of truth; demand a real professional standard instead of the generic AI look; build the public site

before the private Sales Hub.

31. The Build-Verify-Publish Discipline

The companion deck to the Lovable chapters gives this discipline its formal shape. It is written for a local **Content Studio** (React/TypeScript) but the discipline ports to Lovable builds.

Confirm before you build

- **Confirm environment and folder first.** Before generating anything, verify the project folder and environment. "Do not generate into the wrong folder."
- **Write a product brief.** One page: what the app does, who it serves, what success looks like. Scaffold from the brief, then take a **Git checkpoint** so you can roll back.
- **Write AGENTS.md.** A short instruction file in the repo telling any AI agent how this project works: commands, conventions, what not to touch.

Build in small verifiable steps

- **First screen, then the prompt builder, then local saving.** Each step gets verified before the next begins.
- **Save locally first, cloud second.** Build and verify locally before any cloud sync, so the app works offline and you always have a fallback.
- **Add tests as you go.** Test generation, editing, saving, refresh survival, separate user identities, mobile layout, and keyboard navigation. Deliberately cause failures and confirm the app handles them.
- **Distinguish two things:** an AI-generated codebase versus an app that calls AI. They have different security and cost profiles. Know which one you are building.
- **Ask for evidence, not "all done."** When the agent says a feature works, make it show you: screenshots, test output, a working flow.

Keys, review, and release

- **Real server-side AI keys belong in environment secrets, never in browser code or chat.** This is the same rule as the Gumloop connector rule, applied to code.
- **Review deliberately, then README, then commit.** Read the diff, write the README, commit with a clean message.
- **Release candidate, restricted deployment first.** Deploy to a restricted audience, verify on the hosted URL (not just localhost), then decide on the wider audience.
- **"Deployment success alone does not mean the site is public."** Check the actual visibility settings.

- **"Restoring code does not reverse database migrations."** Rolling back code is not rolling back schema. Plan migrations accordingly.

32. Getting Found: Visibility Inside AI Assistants

AI-search visibility, how businesses get surfaced inside AI assistants and AI search, deserves its own treatment. The detailed framework behind this topic was not captured in full, so this chapter marks the topic and points to the durable principles rather than reconstructing a method.

The documented SEO teaching

From the website rebuild's SEO scan, the rule for AI visibility: **a sitemap helps.** A sitemap helps crawlers discover pages, especially on new or large sites, but it is a suggestion, not a requirement. Do not state that agents look at the sitemap first.

The working framing was that agents scrape your site by looking at the sitemap first. Treat that as the mental model from the build, not doctrine: per Google Search Central, crawlers discover URLs through previously crawled pages, and sitemaps are useful suggestions rather than absolute requirements.

What the material supports on this topic:

- **Your site is your source of truth.** The same discipline from Chapter 28 (real content, real copy, real structure) is what makes a site legible to both humans and AI systems.
- **Structured content wins.** Clear pages (Home, About, Portfolio, Team), descriptive copy, and real contact paths make a site legible to humans and assistants. Phrase this as best practice; no evidence was found that it directly changes AI citations.
- **The consent checkbox is not optional.** If AI-driven discovery brings you inbound leads, your forms need consent handling from day one.

The full visibility framework behind this topic was not captured in detail. Treat this chapter as a pointer to the durable principles rather than a complete method.

PART IX.
SECURITY AND GOOD JUDGMENT

Data boundaries, permissions done right, the draft-only rule, and autonomous agent safety. The guardrails that keep you out of trouble.

33. Data Boundaries

The rule that runs under everything: **know what data you are feeding the machine, and where it goes.**

- **Never put secrets in chat.** No API keys, passwords, or security keys in a chat window: ChatGPT may use your conversations for model training unless you opt out. "If I put on the chat, it would be almost public, right?" Keys live in connectors, vaults, and environment secrets (Chapters 25, 31).
- **Separate your worlds.** One useful practice: run agent experiments on a separate Gmail and separate cloud account, a personal cloud and an agent cloud (two subscriptions), so experiments never touch personal data.
- **Memory has borders too.** Keep family life and work life in separate systems. In enterprise settings: never share SSNs or payment info with an agent. Brand protection rules belong in your guardrails.
- **The screenshot rule:** anything pasted into ChatGPT, including screenshots, becomes model input and may be used for training unless you opt out in Data Controls. Paste accordingly.

34. Permissions Done Right

- **Least privilege, always.** The Copilot permissions story: a company rolled out Copilot without setting permissions, and employees could read financial data and payroll documents they should never have seen. Set permissions before rollout, not after. Treat this as an anecdote, not a verified case study.
- **Read is safe, write needs a gate.** Gumloop connector consent (Chapter 25): enriching a contact costs credits, writing to Salesforce changes your system of record. Treat writes to Salesforce as high-stakes and gate them for human approval. Both should ask first. The three human positions: human-before-the-loop for risky starts, human-in-the-loop mid-execution, human-after-the-loop for review.
- **Powerful tools get boundaries.** OpenClaw can touch your whole computer. Treat the deletion story as an anecdote rather than a verified incident; the guidance stands on its own: capability without scoping is a liability.
- **Review the grants regularly.** Connectors, OAuth grants, API keys: audit what your agents can still touch. Revoke what the task no longer needs.

35. The Draft-Only Rule

Your email triage agent (Chapter 10) does not send email. The Smart Inbox (Chapter 25) drafts replies and labels mail but "it's not gonna send anything." The Lovable Sales Hub (Chapter 28) drafts outreach for human approval, and the send engine was deliberately left unfinished during the build.

The pattern is the policy: **agents draft, humans send.** This is a recommendation, not a measured result. It is the cheapest safety upgrade in the whole playbook: any agent that touches outbound communication, money movement, or system-of-record writes gets a human gate first.

36. Autonomous Agent Safety

For agents that run on triggers while you sleep:

- **Start with read-only agents.** Let the agent observe, summarize, and draft for a week before it gets any write permission.
- **Approval gates on every write path.** New contacts, emails, database writes, purchases: each gets its own gate (Chapters 25, 35).
- **Budgets and kill switches.** Gumloop credits cost $0.005 each and the monthly allotment resets each billing period, so set spend alerts before they run out (Chapter 25). Know how to pause every scheduled agent in one place.
- **Watch the ledger.** Log what the agent did (the seventh agent part), and for high-stakes systems keep an immutable ledger of every write plus a watchdog agent that watches the ledger.
- **Reflections are a safety tool, not just an optimizer.** A weekly reflection report shows you what the agent actually did, which is the cheapest audit you have (Chapters 25, 26).
- **Never automate what you have not watched.** Run every workflow manually supervised first. The unsupervised version comes only after the supervised version is boring.

APPENDICES

Reference material to keep beside you: the copy-paste prompt library, the terminology map, the open-source starter guide, and answers to the questions that come up along the way.

Appendix A. Copy-Paste Prompt Library

The six-block production prompt (Chapter 6): **Outcome, Context, Inputs, Output contract, Boundaries, Quality check**.

The professional brief (Chapter 7): "You are a senior [field] consultant. A client asks for [deliverable]. Before proposing anything, ask me the questions a real professional would ask. Then deliver [format], covering [sections]. Flag anything I have not told you that a professional would need to know."

The daily brief seed (Chapter 10): give the agent your sources (calendar, email, tasks, news topics, health), your constraints (wake time, commute, what counts as urgent), and the output contract (what sections, what order, what time it arrives). The deck digest preserves the operating contract; the full verbatim prompt was shared on screen.

Lead qualification (Chapter 25): "Qualify this contact as a prospect." **Skill builder:** "Remember this conversation. Create a skill to qualify leads as a prospect." **Sales pipeline:** "Act like a sales agent and tell me how our pipeline looks for this year." **Lookalikes:** "Go find me a thousand more lookalikes to these hundred clients." **ICP builder:** "I want to create my ideal customer profile. Ask me questions to help me build that profile." **Meeting prep:** "Before every meeting on my Google calendar, send me a preparation document for the people who are on that meeting. Include the topic, the people on the meeting, and one or two sentences about each of them." **Event creation:** "Create an event on Luma for our upcoming Generative Gatherings in November. Make the event private for now, so nobody sees that. It's a free event." **Salesforce rule:** "Hey, do not change anything on Salesforce."

Lindy lead pipeline (Chapter 26): Agent 1, scheduled Mondays 10 AM: find and score ten companies matching the ICP, write to Google Sheets. Agent 2, app trigger on the sheet: "Once a new company added to the Google Sheets. Go find contact information about the company and founder's name, founder's LinkedIn profile. And then create another Google sheet" (tell it to create the sheet once and reuse it). Dashboard: "Build me a dashboard shows the information on the connected Google Sheets that my agent is using."

Lovable build seed (Chapter 28): "Create a premium website for [company] ([url]) with a [describe the aesthetic] style..." plus: paste the URL, screenshots, and copied site text; "Use this website as the source of truth"; demand a real professional standard, never the generic AI look.

The planning prompt: "Think with me before you come up with the answer." "Can we plan? Can we think about this in steps?" **Observation prompt:** "Have you completed the task? Is this to the right level of quality that I was expecting?"

Appendix B. AI Terminology Map, Quick Reference

The full map is Chapter 4. The working set:

- **LLM / SLM:** large / small language model.
- **Prompt / system prompt:** what you type / the hidden instructions shaping behavior.
- **Context window:** how much the model can consider at once. Context windows vary widely by model and change fast, so check the model card for the current figure.
- **Hallucination:** confident wrongness; verify everything (Chapter 8).
- **RAG:** retrieval augmented generation, pulling outside data into context.
- **Embeddings:** meaning as numbers, the basis of semantic search.
- **Agent:** software that perceives, reasons, and acts through tools.
- **MCP:** model context protocol, how models connect to tools (Anthropic's open standard; official docs at modelcontextprotocol.io).
- **Function / tool calling:** the model invoking an API mid-reasoning.
- **Guardrails:** what the agent must not do.
- **Human-in-the-loop / before / after:** where the human sits relative to execution.
- **Skill:** a reusable, demonstrable capability; an agent's memory.
- **Trigger:** what starts an agent (schedule, app event, chat, email).
- **Reflection:** the agent reviewing its own history and suggesting improvements.
- **Fine-tuning:** retraining a model on your data. Try prompting and RAG before fine-tuning; enterprise data shows RAG is the more common and cost-efficient first adaptation step.
- **Inference cost:** what you pay per answer; the third corner of the triangle (a decision framework, not an industry standard).

Appendix C. Open-Source Models, Starter Guide

From the Part 1 deck:

- **Why open source:** no per-token fees to a model vendor, runnable locally or on your own infrastructure, though self-hosting still costs compute. Some open models publish weights you can inspect, but auditability depends on each model's license. The trade: open models often trail frontier models, but the gap changes over time and no fixed lag can be stated.
- **Names worth knowing:** DeepSeek, Meta's Llama family, Mistral, Qwen, Kimi, Minimax. In Gumloop's model picker these appear as the cheaper options: Gumloop bills tokens at providers' list prices plus an orchestration fee, so cheaper open models consume fewer credits.
- **When to reach for them:** high-volume, low-stakes tasks (classification, extraction, first-draft summarization), cost-sensitive agent loops, and anywhere data must stay inside your own infrastructure.
- **When not to:** the reasoning-heavy steps, final customer-facing copy, anything where the quality gap costs more than the tokens save. Apply the triangle (performance first, then latency, then cost): a decision framework, not an industry standard.

- **How to start:** pick one task you already run on a frontier model, run it on an open model, and compare outputs blind. Promote the open model only where the outputs are indistinguishable for your use case.

Specific model versions move fast. Verify current leaders before committing.

Appendix D. ChatGPT Plus vs Pro and Other Questions

Do I need ChatGPT Pro? The answer from Part 4: most professionals do not. "I think Plus is enough. You don't have to be Pro." That is a recommendation for most professionals, not a universal fact. The detail that Pro buys higher limits and priority on the newest models is the editor's summary, not a direct quote. ChatGPT Plus costs $20 per month and OpenAI offers a $100 Pro tier; new sign-ups for the $200 Pro tier have been temporarily paused since September 10, 2026. Verify current pricing before publishing. The "Plus is enough" framing still holds for most people: buy it when you can point to a specific workflow that Plus throttles.

Which model should I use? The one that is best this month, for your task, at your budget. The triangle (performance, latency, cost) is the decision tool: a working framework, not an industry standard. For agent loops that run unattended, cheaper and faster usually beats smarter. That is practical guidance for unattended loops, not a measured result.

Should I fine-tune a model? Almost certainly not as a first move. Prompting, RAG, skills, and better context win first. Fine-tuning is the expensive last resort.

Is my data training their model? Retention is product-specific: OpenAI may use ChatGPT conversations for training unless you opt out in Data Controls, while Anthropic states Claude is not trained on user prompts. This is why Chapters 33 through 36 exist: boundaries first, cleverness second.

How do I keep up? The actual method: the innovation reaction loop (Chapter 1). Run one focused experiment per week, share what you learned, repeat. This playbook is the curriculum; the loop is the habit.

Appendix E. Links and Resources

Every tool linked in this book, organized by chapter, plus the sources behind the numbers. Links were verified in September 2026. Tools change fast: if a link dies, search the tool's name plus "official site."

Chapters 1-5

- ChatGPT: https://openai.com/chatgpt/
- MCP (Model Context Protocol): https://www.anthropic.com/news/model-context-protocol (Anthropic's announcement, November 2024)

- Claude: https://claude.com
- Claude Code: https://claude.com/product/claude-code
- Codex: https://openai.com/codex/
- Microsoft Copilot: https://www.microsoft.com/en-us/microsoft-365-copilot
- Claude Microsoft 365 connector docs: https://claude.com/docs/connectors/microsoft/365
- Hugging Face: https://huggingface.co
- Google Colab: https://colab.research.google.com
- Meta Llama: https://github.com/meta-llama/llama-models
- Qwen: https://github.com/QwenLM
- Hugging Face leaderboards: https://huggingface.co/docs/leaderboards/en/leaderboards/finding_page
- MarkItDown (Microsoft's PDF-to-Markdown converter): https://github.com/microsoft/markitdown
- Apollo: https://www.apollo.io/product/prospect
- Asana: https://asana.com
- Canva: https://www.canva.com
- Dropbox: https://www.dropbox.com
- Gamma: https://gamma.app
- GitHub: https://github.com
- Gmail: https://workspace.google.com/gmail/

Chapters 6-8

- Hugging Face: https://huggingface.co

Chapters 9-12

- Zapier: https://zapier.com
- Granola: https://www.granola.so/
- Asana: https://asana.com
- Gamma: https://gamma.app

Chapters 13-15

- Granola: https://www.granola.so/
- Gamma: https://gamma.app

Chapters 16-19

- Descript: https://www.descript.com/
- Opus Clip: https://www.opus.pro
- Higgsfield: https://higgsfield.ai (hosted MCP documented at mcp.higgsfield.ai)

- ElevenLabs: https://elevenlabs.io
- Suno: https://suno.com/

Chapters 20-22

- Awesome AI Agent Skills: https://github.com/seb1n/awesome-ai-agent-skills

Chapters 23-26

- Zapier: https://zapier.com
- Make: https://www.make.com/en
- n8n: https://github.com/n8n-io
- Gumloop: https://www.gumloop.com
- Firecrawl: https://www.firecrawl.dev/crawl
- Lindy: https://lindy.ai
- Apollo: https://www.apollo.io/product/prospect

Chapters 27-32

- Lovable: https://lovable.dev
- Codex: https://openai.com/index/introducing-the-codex-app/
- Claude Code: http://claude.com/product/claude-code/enterprise
- Stripe: https://stripe.com/payments
- Supabase: https://github.com/supabase/supabase
- Lovable plan mode docs: https://docs.lovable.dev/features/plan-mode
- React: https://github.com/reactjs/react.dev/
- TypeScript: https://www.typescriptlang.org
- Git: https://git-scm.com

Chapters 33-36

- Microsoft Copilot: https://www.microsoft.com/en-us/microsoft-copilot/organizations
- OpenClaw: http://github.com/openclaw/openclaw
- Gumloop: https://www.gumloop.com
- Salesforce: https://www.salesforce.com

Appendices A-D

- Lindy: https://lindy.ai
- Google Sheets: https://workspace.google.com/products/sheets/
- MCP docs: https://modelcontextprotocol.io
- DeepSeek: https://github.com/deepseek-ai

- Meta Llama: https://github.com/meta-llama/llama-models
- Mistral: https://mistral.ai/
- Qwen: https://github.com/QwenLM
- Kimi: https://github.com/MoonshotAI
- MiniMax: https://www.minimax.io
- Hugging Face: https://huggingface.co

Sources for the numbers

- MIT NANDA "The GenAI Divide: State of AI in Business 2025" (the 95% pilot figure): covered August 2025 by Tech.co, Computing.co.uk, Campus Technology, and Virtualization Review.
- Zapier 9,000+ integrations: zapier.com.
- Microsoft Copilot $30/user/month: 2023 launch pricing (Microsoft blog); current list pricing $23.50 to $32 (microsoft.com).
- Claude Microsoft 365 connector: public reporting dates the launch to October 2025; docs at claude.com/docs/connectors/microsoft/365.
- OpenAI training opt-out ("Improve the model for everyone" in Data Controls): OpenAI help center.
- ChatGPT connectors respect existing user permissions: The Register, October 2025.
- Copilot oversharing via broad permissions: Computerworld.
- Replit CEO on fewer apps, more agents: Platformer interview.
- Suno licensing deals and lawsuits: Wikipedia (Suno AI).
- Sitemap guidance ("useful suggestions to Googlebot, not absolute requirements"): Google Search Central.
- ChatGPT Pro tiers ($100 Pro; $200 Pro new sign-ups paused September 10, 2026): OpenAI help center; TechCrunch on the $100 Pro tier.
- Data retention: OpenAI Help Center (Data Controls FAQ); Anthropic Enterprise (prompts not used for training).
- Lovable Cloud is a Lovable-managed Supabase instance: Supabase docs.
- Gumloop credits ($0.005 each; monthly allotment resets; 8% orchestration fee): gumloop.com/pricing.
- ElevenLabs instant cloning (60 seconds of audio) and professional cloning (30 minutes): ElevenLabs docs.
- Descript Create Clips: descript.com. Opus Clip virality scores: opus.pro.
- Higgsfield multi-model hub: higgsfield.ai help center.
- Granola: Business Wire, May 2024.

Keep Going

You now hold the complete playbook: the mental models, the prompts, the personal systems, the automations, the agent architecture, and the safety rules. Everything in these pages uses real tools and real examples.

The full paid playbook goes deeper, with advanced builds, the complete copy-paste prompt library, and new systems as they ship.

Get the weekly letter:

newsletter.burhansebin.com