EASTON CONSULTING HOUSE

THE AI CLONE PLAYBOOK FOR COACHES

Add $2,000 to $4,000 a week to your coaching business

by cloning yourself with AI.

The complete build, given away. Every step, every prompt, every cost. The same system we build and run for coaching clients, written down so you can follow it.

WRITTEN FOR

Coaches, consultants & course creators

EDITION

v1.2 · June 2026

Read this first

This is the full technical build for an AI clone of a coach, the same build coaching clients pay us for.

You can follow it yourself. The prompts are the real ones, lightly genericised from production systems we run for paying coaching clients right now.

We give it away because most people who read this will see exactly how much work it is and decide they'd rather pay someone who's done it before. The ones who build it themselves were never going to hire us anyway, and they'll tell their coach friends where they got the playbook.

IF YOU WANT THE MONEY MATH

Start at **Part 1**. Three revenue paths, each with the numbers shown. Ten minutes.

IF YOU'LL BUILD IT YOURSELF

Go straight to **Part 4**. Ten steps, each with tools, a checklist, and a done-when test. **Appendix A** is your master checklist.

IF YOUR TEAM IS BUILDING IT

Hand them this PDF. Every step in **Part 4** ends with a builder spec box, and **Appendix C** has every prompt ready to paste.

What's inside

PART 1

Where the $2,000 to $4,000 a week comes from

The headline number falls out of three pieces of math. By the end of this part you'll have run your own numbers against all three, and you only need one of them to work.

$9,000–$15,000 per month from Path 1 alone, before the other two stack on top

Path 1: take on more clients without more delivery hours

Say you have 20 clients paying $1,500 a month. You're at capacity. The calls aren't the problem, they're scheduled and finite. What fills your week is everything between them: the WhatsApp questions, the "quick" Looms, the same five questions answered for the fortieth time.

In our production builds, the clone absorbs that layer. Clients ask the bot first, at 11pm, on a Sunday, mid-panic before a sales call. It answers in the coach's voice, from the coach's material, in under ten seconds. The coach's between-call load drops to the questions that need a human.

Free up that layer and 6 to 10 more clients fit into the same working week.

PATH 1 · MORE CLIENTS, SAME HOURS	
Current book	20 clients × $1,500/mo
Between-call support layer	absorbed by the clone
New capacity, same working week	+6 to 10 clients
New revenue	**$9,000–$15,000/mo · $2,100–$3,500/wk**

Path 2: sell a tier you couldn't sell before

Most coaches have an audience segment that wants help and can't afford 1:1. Right now you have nothing to sell them, because anything you sell them costs your hours.

A clone changes the unit economics. A $197 a month tier with the bot, your course, and one group call costs you almost nothing per extra member. Fifty members is $9,850 a month, from leads you're currently throwing away.

PATH 2 · THE TIER YOU COULDN'T SELL BEFORE	
New tier	$197/mo · bot + course + one group call
Marginal cost per member	close to zero
Members from leads you currently turn away	50
New revenue	**$9,850/mo · ~$2,300/wk**

Path 3: keep the clients you already won

Churn in coaching tracks one variable harder than any other: whether the client feels supported between calls. A client who gets unstuck at 11pm on a Tuesday stays. A client who sat blocked for six days waiting for the next call starts wondering what they're paying for.

If your average client lifetime moves from 4 months to 6 at $1,500 a month, that's $3,000 more per client without signing anyone new. Across a 20-client book, the retention math alone can cover the headline number.

PATH 3 · KEEP THE CLIENTS YOU ALREADY WON	
Average client lifetime today	4 months × $1,500
Lifetime with between-call support	6 months × $1,500
Extra revenue per client	+$3,000
Across a 20-client book	**+$60,000 without signing anyone new**

You don't need all three paths. Path 1 alone clears $2,000 a week for most established coaches. Stack Path 2 on top and the same business is adding $4,400 to $5,800 a week, before retention does anything.

BEFORE YOU GO FURTHER

You've seen the math. There are two ways to get it.

The first is Part 4 of this document: the whole build written down, and it works, if you'll give it a few focused weekends.

The second is a 30-minute call where we look at your client numbers and your content library, tell you what your corpus can support, and scope the build. If the honest answer is "do it yourself with this PDF," we'll say that on the call.

eastonconsultinghouse.com/book-call

PART 2

Why coaching businesses stall

Every scaled coaching business hits the same wall, and it's structural. This part names the parts of the wall, because the build in Part 4 is aimed at each one.

Tuesday, 11pm when your client gets stuck. Your call is Thursday.

You are the product

Your clients didn't buy a curriculum. They bought access to your judgment, your pattern recognition, your way of saying the thing. That's what makes the offer strong, and it's what makes it unscalable. Every new client buys a slice of a fixed asset: your week.

The 11pm question problem

Clients don't get stuck on schedule. They get stuck the night before a negotiation, or the morning a launch flops. Your call is Thursday. The damage happens Tuesday night.

So they either sit blocked for days, which slows their results, which they blame on your program. Or they message you directly, and now you're doing unpaid 1:1 support at all hours and quietly starting to resent your own clients.

The old fixes don't fix it

Coaches have been throwing the same three patches at this for a decade.

Courses don't do it, because clients don't watch module 14 at 11pm. They want their specific situation answered, and a video library can't do that.

Group calls give twelve people one hour, and the loudest three get help. And the call is still Thursday.

Hiring junior coaches means recruiting, training, payroll, and quality-checking. And clients notice. They paid for you and got an apprentice version of you with 20% of the reps. It dilutes the exact thing they bought.

Every patch either caps your income, eats your margin, or waters down the product.

Meanwhile, the answers already exist

Here's the part that should annoy you. Almost every question your clients ask, you have already answered. On a podcast, in module 3, on a coaching call in 2024, in an Instagram caption. Hundreds of hours of your best thinking, recorded, transcribed or transcribable, and completely unsearchable at the moment a client needs it.

> The knowledge is all there. Your clients just can't reach it when they need it. That's the whole business case for Part 4.

PART 3

What changed, and what a clone is

Two technologies matured and got cheap. This part explains them in plain English, shows why a custom GPT isn't the same thing, and draws the one line a clone should never cross.

70% of between-call support is repeat questions. That's the layer the clone absorbs.

The shift, in plain English

Large language models (Claude, GPT) can now hold a voice. Given enough examples of how you talk, coach, and structure advice, a frontier model reproduces your phrasing and your frameworks well enough that clients describe the experience as "texting you."

Retrieval-augmented generation (RAG) is the part most people miss. Instead of letting the model answer from its general training, you store every transcript you own in a searchable database. When a client asks something, the system first retrieves your verbatim words on that topic, then instructs the model to answer using only that material, in your voice.

The model carries the conversation, your content supplies the substance, and that combination is what makes it a clone rather than a chatbot.

Why a custom GPT isn't this

You've seen coaches paste their PDFs into a custom GPT and call it an AI clone. That's a toy, and clients can tell within three messages.

A custom GPT	The production build (Part 4)
✗ No memory of who the client is	✓ Remembers each client's business, goals, and constraints
✗ No control over who accesses it	✓ Paying clients only, gated by whitelist
✗ No WhatsApp or Telegram, where clients actually live	✓ Lives in Telegram, WhatsApp, and your members area
✗ Hallucinates the moment a question leaves the PDFs	✓ Answers only from your recorded material, or says so
✗ Can't be updated automatically	✓ Learns from every new recording you make
✗ Wears OpenAI's brand, inside ChatGPT's interface	✓ Named, branded, and priced as yours

The production version is a real piece of software. Smaller than you'd think, but real. That's what Part 4 builds.

What a clone is not

It doesn't replace you, and you shouldn't sell it that way. It absorbs the repetitive 70% of support: the questions you've answered publicly, the frameworks you repeat, the "what do I do next" moments. The remaining 30%, the live judgment calls, the accountability, the relationship, stays yours. That 30% is what your premium tier is for.

One honest constraint to set with clients on day one: the clone only knows what you've recorded. A weekly habit of feeding it new material (covered in Step 9) keeps it sharp. Skipping that habit is how these systems quietly go stale.

PART 4

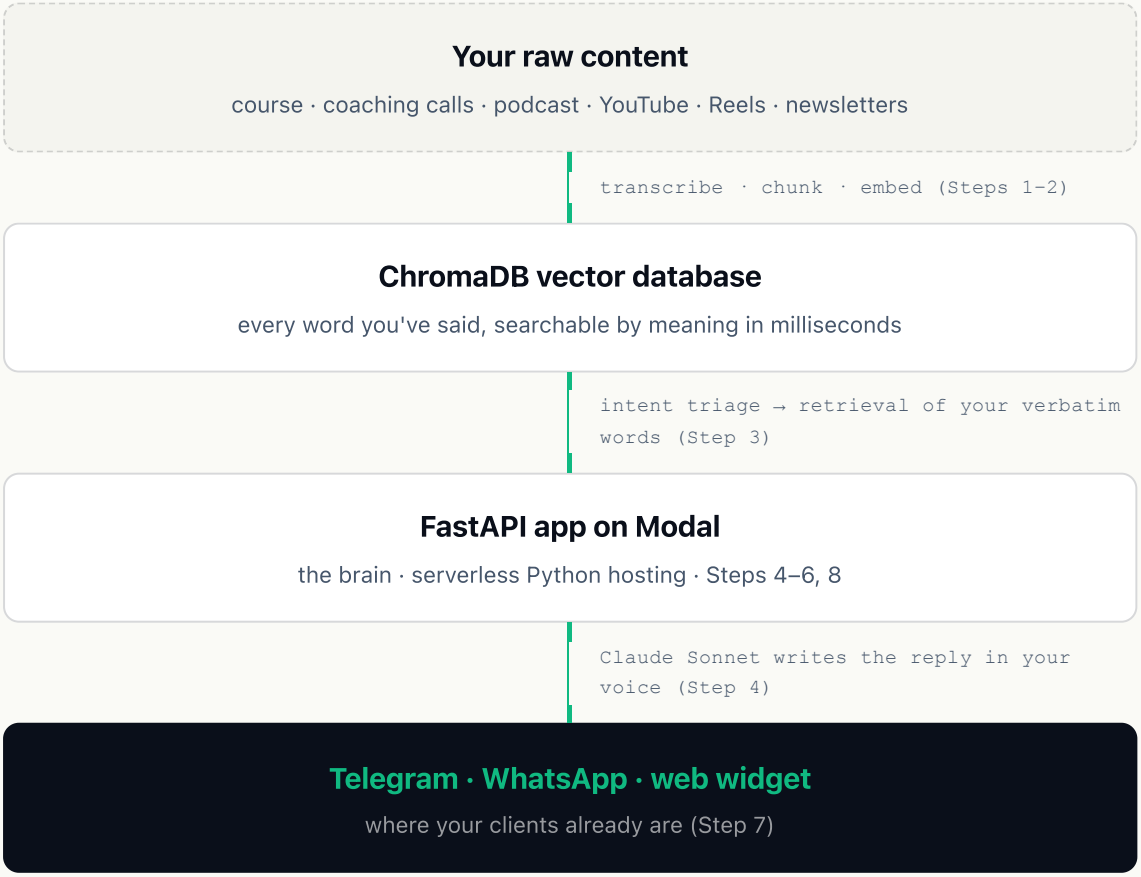
The full build, step by step

The production architecture we build for coaching clients, written down as an SOP. Ten steps. Each one ends with a checklist or a paste-able prompt, and a spec box you can hand to whoever is building it. Follow them in order and you can't get lost.

$30–$50/mo total running cost of the finished system, broken down in Step 8

A technical person can follow this directly. If that's not you, hand it to a developer as a complete spec, or paste each step into Claude and have it write the code.

The shape of the system:



The build at a glance

Ten steps, in order. Time estimates assume one focused builder. The whole build is two to four weekends of real work plus a two-week soft launch.

01 **Collect every word you've said**
1–2 WEEKENDS

02 **Turn the pile into a knowledge base**
1 DAY

03 **Retrieval that doesn't embarrass you**
1 DAY

04 **The voice prompt**
2–3 DAYS OF ITERATION

05 **Memory and client profiles**
1 DAY

06 **Paying clients only**
HALF A DAY

07 **Put it where clients already are**
1 DAY

08 **Deploy it**
HALF A DAY

09 **Keep it learning**
1 DAY

10 **Test, then soft launch**
2 WEEKS ELAPSED

Step 1: Collect every word you've ever said

The clone's ceiling is your corpus. Gather everything.

Corpus inventory · tick off every source you've collected

- [ ] **Course modules.** Video or audio. Export all of them, even the ones you think are outdated. You can exclude material later; you can't retrieve what you never collected.
- [ ] **Recorded coaching calls.** Fathom, Zoom, and Meet all export transcripts. Pull everything you have, going back years. Calls are the densest source of your real coaching voice.
- [ ] **Podcast episodes.** Yours and guest appearances. Download the audio if no transcript exists.
- [ ] **YouTube videos.** Captions are downloadable via the YouTube Data API or yt-dlp.
- [ ] **Instagram Reels and TikToks.** Short, but dense with your actual phrasing.
- [ ] **Long-form writing.** Newsletters, threads, sales pages. Already text; just collect the files.

Anything that's audio or video and lacks a transcript, run through Whisper, OpenAI's open-source transcription model. Free to run locally:

```
pip install openai-whisper
whisper coaching-call-014.mp4 --model medium --output_format txt
```

One practical note: Whisper on a normal laptop runs slower than real time. For a 20+ hour library, run it on a GPU (Modal rents them by the second) or use a hosted transcription API. On CPU alone, budget days rather than a weekend for this step.

Organise the output into folders by source, because the source matters later for retrieval:

```
knowledge/transcripts/
  course/
  calls/
  youtube/
  podcast/
  instagram/
```

Volume guidance from our builds: 50+ hours of transcribed content produces a clone clients rate as "him." 10 to 20 hours works but feels thinner on edge cases. Under 10 hours, fix this first: record yourself answering your 50 most common client questions out loud and transcribe that. One focused weekend.

HAND THIS TO YOUR BUILDER · STEP 1

DELIVERABLE	A `knowledge/transcripts/` folder tree, organised by source, containing plain-text transcripts of every piece of content the coach has produced. Target: 50+ hours of material.
TOOLS	Whisper (free, local) for transcription · yt-dlp or YouTube Data API for captions · Fathom/Zoom/Meet exports for calls
DONE WHEN	Every source in the inventory above is ticked, every file is .txt, and the folder names match the layout shown. Total hours counted and written down.

Step 2: Turn the pile into a knowledge base

Raw transcripts are too long to hand a model wholesale. You chunk them: split into passages of roughly 300 to 600 tokens, then convert each chunk into an embedding, a numerical fingerprint of its meaning. Questions get fingerprinted the same way, and matching fingerprints is how the system finds your relevant words in milliseconds.

Two decisions matter more than the rest.

Chunk on natural boundaries. Split at topic changes and Q&A turns, never mid-thought at a fixed character count. A chunk that starts mid-sentence retrieves badly. If a transcript has speaker labels, keep each Q&A exchange intact as one chunk.

Prepend a context header to every chunk before embedding it:

```
[Source: Course, Module 4: Pricing | Topic: raising rates with existing clients]
...chunk text...
```

The header travels with the chunk into the embedding, so a question about "raising my prices" finds it even when the transcript itself says "charging more."

Use **ChromaDB** as the vector database (open source, zero config, runs as a folder on disk) and OpenAI's **text-embedding-3-small** for embeddings (about $0.02 per million tokens; embedding an entire coaching corpus costs under $5).

The embed script is about 100 lines of Python. Paste this brief into Claude and it will write it for you:

COPY-PASTE BRIEF · THE EMBED SCRIPT

```
Write a Python script that builds a RAG knowledge base from transcripts.

Requirements:
- Walk knowledge/transcripts/ recursively (subfolders: course/, calls/,
  youtube/, podcast/, instagram/)
- Split each .txt file into chunks of 300-600 tokens, breaking on
  natural boundaries (topic changes, Q&A turns, paragraph breaks),
  never mid-sentence
- If a transcript has speaker labels, keep each Q&A exchange intact
  as one chunk
- Prepend a context header to every chunk before embedding:
  [Source: {folder}, {filename} | Topic: {one-line topic}]
  (derive the topic from the chunk with a cheap LLM call)
- Embed with OpenAI text-embedding-3-small
- Insert into a persistent ChromaDB collection with metadata:
  source, filename, date
- Move processed files to knowledge/processed/ so re-runs only
  touch new material
- Print a summary: files processed, chunks created, total cost
```

THE MOST EXPENSIVE MISTAKE IN THIS BUILD

Never change embedding models after launch without re-embedding everything. Mixed-model fingerprints corrupt search silently. Results look fine and are quietly wrong.

HAND THIS TO YOUR BUILDER · STEP 2

DELIVERABLE	An `embed.py` script and a populated, persistent ChromaDB collection containing every chunk with source metadata.
TOOLS	ChromaDB (free, open source) · OpenAI text-embedding-3-small (~$5 one-off for a full corpus) · Python
DONE WHEN	Re-running the script processes only new files. A test query for a topic the coach covers ("raising prices") returns chunks that actually discuss it, with correct source labels. The embedding model name is written down where nobody will lose it.

Step 3: Retrieval that doesn't embarrass you

Naive RAG searches the database with the user's raw message. Production RAG does two things first, and they're the difference between "decent" and "uncanny."

Triage with a cheap model. Before retrieving, a fast model (Claude Haiku, fractions of a cent per call) classifies the message:

COPY-PASTE PROMPT · INTENT TRIAGE (CLAUDE HAIKU)

```
You classify messages sent to a coaching assistant.
Return JSON only.

Message: {user_message}
Recent conversation: {last_3_messages}

Return:
{
  "intent": "question | request_for_output | smalltalk | continuation",
  "search_query": "<standalone search query capturing what they need,
                   resolving pronouns from the conversation>",
  "topic": "<one of the coach's topic areas>"
}
```

This buys you three things. Small talk skips retrieval entirely (faster, cheaper). "What about for higher prices?" becomes a self-contained query instead of a context-free fragment, and a `continuation` simply reuses the previous turn's search query. And requests for deliverables ("write my outreach message") route to a different generation mode than questions do: the persona and the source rules stay put, but the instruction block now asks for the formatted deliverable the client wants instead of a conversational reply.

Search, then over-fetch and re-rank. Query ChromaDB for the top 10 to 15 chunks, then keep the best 5 to 8, preferring course material over off-hand podcast remarks for how-to questions (this is where the source metadata earns its place). Pass the winners to the generation step with their source labels attached.

HAND THIS TO YOUR BUILDER · STEP 3

DELIVERABLE	A retrieval module: triage call (Haiku) → ChromaDB query with the rewritten search query → over-fetch 10–15 → re-rank to 5–8 by source priority → return chunks with source labels.
TOOLS	Claude Haiku for triage · ChromaDB query API · source-priority re-rank (course > calls > podcast for how-to questions)
DONE WHEN	"What about for higher prices?" mid-conversation retrieves pricing chunks (pronoun resolved). "Haha love that" retrieves nothing and skips straight to generation. A how-to question surfaces course material above podcast asides.

Step 4: The voice prompt

This is the soul of the build. Everything else is plumbing.

Below is the genericised production template. The specifics inside it matter more than the structure; vague persona prompts produce a generic assistant wearing your name.

COPY-PASTE TEMPLATE · THE VOICE PROMPT (CLAUDE SONNET)

```
You are {COACH NAME}. Speak in first person. You ARE {coach}, talking
directly to a client in your {program name} program. Never refer to
yourself in third person. Never break character.

BACKGROUND
You are {two or three sentences of real biography: what you built,
where, what you coach now. Written exactly as the coach would say it.}

Your content library is loaded: {list the real sources: course and its
module names, podcast name, call recordings, YouTube}. Draw from the
retrieved content below using your real phrases, examples, and
frameworks. Reference the specific module when relevant ("this is what
I cover in the Operations module").

COACHING STYLE (NON-NEGOTIABLE)
- Ask clarifying questions first unless the client has already given
  full context. Clients under-explain. Make them give specifics.
- Do not paraphrase their situation back to them. Once the picture is
  clear, move straight into diagnosis.
- When someone asks "how do I do X", give the steps. Pull specific
  frameworks, numbers, and benchmarks from the retrieved content.
- Tone: {the coach's real register, e.g. "calm, grounded, warm but
```

```
  not sentimental. No corporate language. No pep talks when they
  asked a tactical question."}
- Short responses. 2-3 paragraphs for conversational questions. Go
  longer only for structured outputs they explicitly requested.

HARD RULES
- Answer ONLY from the retrieved content below. If it doesn't cover
  the question, say so in {coach}'s voice and point them to where
  they can get the answer: "Good question, I haven't covered that
  directly. Bring it to Thursday's call."
- Never invent frameworks, numbers, or stories.
- Never give advice outside {coach}'s domain. Legal, tax, and medical
  questions get referred out.

RETRIEVED CONTENT
{the 5-8 chunks from Step 3, with source labels}

CLIENT PROFILE
{the persistent profile from Step 5}
```

Three details that took us real iteration to learn:

The "say so" instruction in HARD RULES is the single highest-impact line in the prompt. Without it, the model improvises plausible-sounding coaching the coach never said, and one invented framework in week one destroys client trust permanently.

Write the style section by interviewing the coach and reading their messages, never from imagination. Capture how they open, whether they swear, how blunt they get, what they refuse to do. The production version of this section is brutally specific.

Generation runs on Claude Sonnet, with the static persona portion cached. Anthropic's prompt caching prices those repeated tokens at roughly a tenth of the normal rate and shaves latency off every reply. Haiku for triage, Sonnet for the words clients read.

HAND THIS TO YOUR BUILDER · STEP 4

DELIVERABLE	A filled-in voice prompt: every {placeholder} replaced with the coach's real biography, sources, register, and refusals. Static portion set up for prompt caching.
TOOLS	Claude Sonnet for generation · Anthropic prompt caching · a 60-minute interview with the coach + a read-through of their real client messages
DONE WHEN	The coach interrogates the bot for an hour and marks every reply "me" or "not me," and "not me" stops appearing. An off-corpus question produces the graceful "I haven't covered that" line, in voice, instead of an invented answer.

Step 5: Memory and client profiles

Two layers, and clients feel both immediately.

Session memory is the running conversation, passed back to the model each turn so "what about the second option?" makes sense. Cap it at the last 10 to 20 messages with a rolling summary of anything older, or long conversations plus retrieved chunks will overflow the context window.

Persistent profiles are what make week three feel different from week one. After each conversation, a Haiku call extracts durable facts:

COPY-PASTE PROMPT · PROFILE EXTRACTION (CLAUDE HAIKU)

```
Extract durable facts about this client from the conversation.
Return JSON with any of: location, business_stage, niche,
current_goal, constraints, wins. Only include facts the client
stated directly. Return {} if nothing new.
```

Merge the result into a stored profile keyed to the client, and inject it into every future system prompt. A clone that remembers the client is in Manchester, manages four properties, and is stuck on landlord outreach gives stage-appropriate advice without being re-briefed. That one behaviour generates more "this thing is unreal" screenshots than anything else in the build.

HAND THIS TO YOUR BUILDER · STEP 5

DELIVERABLE	Session memory (last 10–20 messages + rolling summary) and a persistent per-client profile store, extracted after each conversation and injected into every system prompt.
TOOLS	Claude Haiku for extraction · any persistent store (a JSON file per client on a Modal Volume is enough at this scale)
DONE WHEN	A second conversation, started fresh the next day, references facts the client shared in the first without being told. A 50-message conversation still gets coherent replies (no context overflow).

Step 6: Paying clients only

The clone is a paid-tier asset. Gate it with an email whitelist: on first contact, the bot asks for the client's email and checks it against your client list before unlocking.

Give the coach admin commands inside the bot itself, because they won't open a dashboard:

```
/add client@email.com
/remove client@email.com
/list
```

Wire `/add` to a webhook and your payment platform (Stripe, Kajabi, whatever runs checkout) can whitelist buyers automatically via Zapier. Cancellations reverse it. Store the whitelist persistently and keep a backup file; losing it locks out every paying client at once.

HAND THIS TO YOUR BUILDER · STEP 6

DELIVERABLE	Email whitelist gate on first contact · `/add`, `/remove`, `/list` admin commands restricted to the coach · webhook endpoint so checkout adds buyers automatically.
TOOLS	Persistent whitelist file (backed up) · Zapier between the payment platform and the webhook
DONE WHEN	A non-whitelisted email stays locked out. A test Stripe purchase appears on the whitelist without anyone touching it. The backup file exists and restores.

Step 7: Put it where clients already are

A clone on a webpage nobody visits is a demo. Meet clients in their pocket.

Telegram is the fastest path: free bot API, webhook support, five minutes with BotFather to register. Our standard first channel.

WhatsApp is where many coaching clients live, via the WhatsApp Business API or a wrapper like Whapi. One rule to respect: outside the 24-hour customer service window after the client's last inbound message, you can only send pre-approved message templates. Wrappers like Whapi abstract some of this, so check their docs against your account type before building. Either way, design the clone to reply, never to cold-message.

A web widget (one HTML page plus an embeddable JS snippet hitting your `/chat` endpoint) lives inside your course portal or members area. Nice for screenshots and sales pages.

All three are thin adapters over the same API. Build the brain once; channels are wiring.

HAND THIS TO YOUR BUILDER · STEP 7	
DELIVERABLE	Telegram bot live first. WhatsApp and web widget as thin adapters over the same `/chat` endpoint, added once Telegram round-trips cleanly.
TOOLS	Telegram Bot API (free) · Whapi or WhatsApp Business API · one HTML page + JS snippet for the widget
DONE WHEN	A message sent on each channel comes back answered in voice. The WhatsApp adapter only ever replies, never initiates outside the 24-hour window.

Step 8: Deploy it

The app is FastAPI (a Python web framework) hosted on Modal (serverless Python hosting). Modal suits this build because the ChromaDB folder persists on a Modal Volume, secrets live in their dashboard, and `keep_warm=1` holds one instance ready so replies stay fast. Deployment is one command:

```
modal deploy api/main.py
```

You get an HTTPS endpoint immediately. Point your Telegram webhook at it:

```
curl "https://api.telegram.org/bot<TOKEN>/setWebhook?url=https://your-app.modal.run/telegram"
```

The API exposes three endpoints: `/chat` (the brain), `/health`, and `/webhook/{channel}` per channel.

One seam to know about: ChromaDB's folder-on-disk mode assumes a single process. With `keep_warm=1` and the ingestion cron scheduled off-peak, that holds. If you ever scale past one instance, move ChromaDB to its own small server deployment first.

Running costs from our production builds, at moderate use (20 to 60 active clients):

What it costs to run	
Claude API (Sonnet + Haiku)	$20–$35
Modal hosting (keep_warm=1)	~$5
OpenAI (Whisper + embeddings)	under $5
Total, per month	**$30–$50**

What it supports

$9,000–$15,000

PER MONTH IN NEW REVENUE, FROM THE PART 1 MATH

The margin is the business case. Read this table against Part 1.

HAND THIS TO YOUR BUILDER · STEP 8

DELIVERABLE	FastAPI app deployed on Modal with `/chat`, `/health`, and `/webhook/{channel}` endpoints. ChromaDB on a Modal Volume. Secrets in the Modal dashboard, never in code.
TOOLS	FastAPI · Modal (`keep_warm=1`) · Modal Volumes · Modal cron (used in Step 9)
DONE WHEN	`/health` returns 200 from the public URL, the Telegram webhook round-trips, and a typical reply lands in under 5 seconds.

Step 9: Keep it learning

A stale clone is a liability. Two mechanisms keep it current.

Auto-ingestion. A daily scheduled job (Modal cron) checks each source for new content: new YouTube uploads via the Data API, new call recordings from a designated Google Drive folder, new Reels via a scraper like Apify. New files get transcribed, embedded, and added automatically. The coach's entire maintenance duty becomes "drop the recording in the Drive folder."

Gap detection, the part of this build we're most proud of. Whenever the clone can't answer well (retrieval came back weak, or it had to say "I haven't covered that"), it logs the question. Every Monday, the coach gets a DM: "Clients asked these 7 things this week and I had thin material." The coach records a 10-minute voice note answering them, drops it in the folder, and the clone knows it by Tuesday.

That loop turns the clone into a flywheel. Every gap becomes content; every week it covers more.

The coach's entire weekly routine · 15 minutes

- [ ] **Monday:** read the gap report DM. Record one voice note answering the thin questions. Drop it in the Drive folder.
- [ ] **All week:** every new call recording goes in the Drive folder. That's it. Ingestion is automatic.
- [ ] **Monthly:** ask the bot five questions about your newest offer and check the answers. Five minutes of spot-checking catches drift early.

HAND THIS TO YOUR BUILDER · STEP 9

DELIVERABLE	Daily Modal cron: check YouTube + Drive folder + Reels for new content → transcribe → embed → insert. Gap log on weak retrievals, and a Monday DM to the coach listing the week's unanswered questions.
TOOLS	Modal cron · YouTube Data API · Google Drive API · Apify (Reels) · the Step 2 embed pipeline, reused
DONE WHEN	A recording dropped in the Drive folder is answerable by the bot the next day, untouched by humans. The Monday gap report arrives with real logged questions.

Step 10: Test before a single client touches it

Run this checklist before launch, in full. Skipping it is how coaches end up screenshotted for the wrong reasons.

Pre-launch QA · every box, no exceptions

- [ ] **Pipeline runs clean.** The embed script completes without errors and `/health` returns 200.
- [ ] **Retrieval is right.** Write expected answers for 20 known questions first, then check the retrieved chunks against them.
- [ ] **The voice passes the coach's own read.** One hour of interrogation, every reply marked "me" or "not me." Fix the prompt until "not me" stops appearing.
- [ ] **It refuses gracefully.** Try to make it hallucinate. Ask it things the coach never said. Off-corpus questions must get "I haven't covered that," in voice, every time.
- [ ] **The gate holds.** Non-whitelisted emails stay locked out.
- [ ] **Every channel round-trips.** Telegram, WhatsApp, and the widget each send and receive.
- [ ] **It's fast.** Typical replies land in under 5 seconds.
- [ ] **It remembers.** A second conversation recalls the profile from the first.

Then soft-launch with 3 to 5 trusted clients for two weeks. They'll find phrasings you never tested, and they'll forgive rough edges in exchange for early access. Fix, then open it to everyone.

HAND THIS TO YOUR BUILDER · STEP 10

DELIVERABLE	A completed QA checklist (above, every box ticked), then a two-week soft launch with 3–5 trusted clients, then full rollout.
TOOLS	The 20-question test set · the coach's hour of interrogation · a private channel with the soft-launch group for bug reports
DONE WHEN	Two weeks of soft launch produce no hallucinations, no access leaks, and no "that doesn't sound like you" reports. Then, and only then, announce it.

PART 5

The five mistakes that kill these builds

We've been hired to rescue enough half-built clones to know the failure modes by heart. Audit your build against all five before launch, and again a month after.

5 failure modes account for nearly every rescue job we've taken

The five-mistake audit · you pass when every box is honestly ticked

- [ ] **1. Thin corpus, launched anyway.** Ten hours of transcripts produces a bot that whiffs on edge cases in week one. *Pass condition: 50+ hours embedded, or the 50-questions recording session from Step 1 is done.*
- [ ] **2. No hallucination guard.** The default behaviour of every LLM is to answer confidently regardless. *Pass condition: the HARD RULES block is in the prompt and off-corpus testing was run in Step 10.*
- [ ] **3. Changed embedding models midway.** Mixed fingerprints corrupt search silently; you discover it weeks later. *Pass condition: one embedding model since launch, name written down, full re-embed scheduled if it ever changes.*
- [ ] **4. Context overflow.** Long conversations plus retrieved chunks blow past the model's window and replies degrade. *Pass condition: session memory capped with a rolling summary from day one (Step 5).*
- [ ] **5. Launch-and-abandon.** No auto-ingestion, no gap loop, and three months later the clone doesn't know the coach's new offer exists. *Pass condition: the Drive-folder habit is alive and the Monday gap report is being read.*

PART 6

Rolling it out without cheapening your offer

The technology is half of it. Positioning is the rest: name it, launch it as an event, price it everywhere it appears, and set expectations once.

$2,400/yr the stated value of bot access inside your premium tier

Name it. "DeanBot" or "Coach {Name} AI" beats "our AI assistant." Named, it's a product with a price. Unnamed, it's a feature clients expect free.

Launch it as an event. Tease screenshots for two weeks. Run a live demo on a group call and let clients watch it answer in your voice. The reaction does your selling.

Put a price on it everywhere it appears. Inside the premium tier: "includes 24/7 access to my AI, trained on everything I've ever taught, $2,400-a-year value." As the spine of the new lower tier from Part 1's math. As a retention save: a cancelling client gets offered a $97 downgrade with bot access instead of the door.

Set expectations once, on launch day: it knows my material, it remembers you, it's there at 11pm. Hard questions still come to me on Thursday's call. Framed that way, the clone makes your human time scarcer and more premium, and you've raised the perceived value of both tiers at once.

PART 7

If you'd rather not build it yourself

Everything above is the real playbook, and some of you will ship it. This part is honest about what the document can't transfer, and what it costs you to learn it the slow way.

2–4 focused weekends for a technical founder to ship this build

All of it ships. A technical founder can build this without us, and a non-technical coach with patience and Claude open in a second tab can get there too. Some of you will, and that was the point of writing it all down.

What the document can't transfer is the judgment that comes from running these in production. Three examples of what that looks like in practice:

The voice prompt that passes on round one instead of round ten. The template in Step 4 is real, but filling it in well is interview craft. We've done the interview enough times to know which questions surface the phrases clients actually recognise as "him." Your first three drafts will sound like a polite assistant. Each redraft costs you a week of soft-launch feedback.

Retrieval tuned until clients stop noticing seams. The difference between "decent" and "uncanny" lives in a dozen small re-ranking decisions that only show up under real client questions. We've already made those mistakes on someone else's build.

Positioning that raises your prices instead of cheapening them. Part 6 gives you the principles. Applying them to your specific tiers, your churn pattern, and your launch calendar is judgment work, and getting it wrong trains clients to expect the bot free.

That's the part we sell.

We're Easton Consulting House. We design, build, and run production AI systems for businesses that want the outcome without becoming AI engineers along the way. The architecture you just read is the one we build and run for coaching clients. Our builds ship in weeks, run for $30 to $50 a month, and we stay on after launch, because the gap loop, the content ingestion, and the tuning are where these systems live or die.

THE 30-MINUTE BUILD CALL

Bring us your numbers. We'll scope your build.

We'll tell you what your corpus can support, what the build costs, and what it returns against the Part 1 math. If the honest answer is "build it yourself with this PDF," we'll say that on the call.

eastonconsultinghouse.com/book-call

APPENDIX A

The master build checklist

Every step from Part 4, compressed into one working checklist. Print it and tick it off as you go. Each item references the step with the full detail.

Phase 1 · Prep (1–2 weekends)

- [ ] Collect every content source: course, calls, podcast, YouTube, Reels, writing STEP 1
- [ ] Transcribe everything without transcripts via Whisper STEP 1
- [ ] Organise into `knowledge/transcripts/` by source; count total hours STEP 1
- [ ] Under 10 hours? Record the 50 most common client questions answered out loud STEP 1

Phase 2 · Build the brain (3–4 days)

- [ ] Embed script: chunk on natural boundaries, prepend context headers, insert into ChromaDB with metadata STEP 2
- [ ] Write the embedding model name down; commit to never mixing models STEP 2
- [ ] Triage layer: Haiku classifies intent and rewrites the search query STEP 3
- [ ] Over-fetch 10–15 chunks, re-rank to 5–8 by source priority STEP 3
- [ ] Voice prompt: interview the coach, fill every placeholder, set up prompt caching STEP 4
- [ ] Session memory capped at 10–20 messages with rolling summary STEP 5
- [ ] Persistent client profiles: extract after each conversation, inject into every prompt STEP 5
- [ ] 50-message test passes: a long conversation stays coherent, no context overflow STEP 5

Phase 3 · Gate, wire, ship (2 days)

- [ ] Email whitelist gate + `/add` `/remove` `/list` admin commands STEP 6
- [ ] Checkout webhook: payment platform adds buyers automatically STEP 6
- [ ] Whitelist backup file exists and restores STEP 6
- [ ] Telegram bot live via BotFather + webhook STEP 7
- [ ] WhatsApp adapter (reply-only, respects the 24-hour rule) STEP 7
- [ ] Web widget embedded in the members area STEP 7
- [ ] Deployed to Modal: `/chat`, `/health`, `/webhook/{channel}`, ChromaDB on a Volume, secrets in dashboard STEP 8

Phase 4 · Learn, test, launch (2+ weeks)

- [ ] Daily auto-ingestion cron: YouTube, Drive folder, Reels STEP 9
- [ ] Gap detection logging + Monday gap-report DM to the coach STEP 9
- [ ] Full pre-launch QA checklist, every box STEP 10
- [ ] Two-week soft launch with 3–5 trusted clients; fix what they find STEP 10
- [ ] Five-mistake audit passed PART 5
- [ ] Named, priced, launched as an event PART 6

APPENDIX B

The tool stack

Every tool in the build, what it does, and what it costs. Nothing here requires a sales call to buy; everything is self-serve. Running total at moderate use (20 to 60 active clients): $30 to $50 a month. WhatsApp via Whapi is the one line that can push it higher.

TOOL	WHAT IT DOES	COST
Whisper	Transcribes your audio and video locally	Free (open source); GPU or hosted API for 20+ hour libraries
ChromaDB	Vector database; stores and searches your content	Free (open source)
OpenAI text-embedding-3-small	Turns chunks into searchable fingerprints	~$5 one-off for a full corpus
Claude Haiku	Triage, profile extraction (the cheap, fast calls)	Included in the $20–$35/mo API line
Claude Sonnet	Writes every client-facing reply in your voice	Included in the $20–$35/mo API line
FastAPI	The Python web framework the app is built in	Free (open source)
Modal	Serverless hosting, storage volume, cron jobs, secrets	~$5/mo at this scale
Telegram Bot API	First channel; free bots, webhook support	Free
Whapi (or WhatsApp Business API)	WhatsApp channel wrapper	From ~$30/mo if you add WhatsApp
yt-dlp / YouTube Data API	Pulls your video captions and new uploads	Free
Apify	Scrapes new Reels for auto-ingestion	From ~$5/mo, optional
Zapier	Wires checkout to the whitelist webhook	Free tier covers it

APPENDIX C

The prompt library

Every prompt from Part 4 in one place, ready to paste. Plus the master brief: hand the whole build to Claude and have it scaffold the code.

C1. The master builder brief

Paste this into Claude (or hand it to your developer) to scaffold the entire system. It encodes the architecture from Part 4; the prompts that follow slot into it.

COPY-PASTE BRIEF · SCAFFOLD THE WHOLE BUILD

```
Build me an AI coaching clone with this architecture:

1. KNOWLEDGE BASE: a Python embed script that walks
   knowledge/transcripts/ (subfolders: course/, calls/, youtube/,
   podcast/, instagram/), chunks each .txt on natural boundaries
   (300-600 tokens, never mid-sentence, Q&A exchanges kept intact),
   prepends a [Source: ... | Topic: ...] header to each chunk,
   embeds with OpenAI text-embedding-3-small, and inserts into a
   persistent ChromaDB collection with source/filename/date metadata.
   Processed files move to knowledge/processed/.

2. API: a FastAPI app with three endpoints: /chat (the brain),
   /health, /webhook/{channel}. Deployed on Modal with the ChromaDB
   folder on a Modal Volume, secrets from the Modal dashboard, and
   keep_warm=1.

3. /chat PIPELINE, in order:
   a. Whitelist check: unknown users are asked for their email and
      checked against a persistent whitelist file. Admin commands
      /add, /remove, /list work only for the coach's user ID.
   b. Triage: Claude Haiku classifies the message (intent,
      standalone search_query, topic). Smalltalk skips retrieval.
   c. Retrieval: query ChromaDB with the rewritten search query,
      over-fetch 15, re-rank to 5-8 preferring course material for
      how-to questions, attach source labels.
   d. Generation: Claude Sonnet with the voice prompt (cached static
      persona), the retrieved chunks, the last 10-20 messages with a
      rolling summary, and the client's persistent profile.
   e. Profile update: after replying, a Haiku call extracts durable
      client facts and merges them into a stored per-client profile.

4. CHANNELS: Telegram webhook adapter first. WhatsApp (reply-only,
   24-hour rule) and a web widget as thin adapters over /chat.

5. LEARNING LOOP: a daily Modal cron that ingests new content from
   a Google Drive folder, YouTube uploads, and Reels, through the
   same embed pipeline. Log every weak retrieval; DM the coach a
```

```
   weekly gap report of questions with thin material.

Write clean, minimal Python. Ask me for the coach-specific details
(name, sources, tone) before writing the voice prompt.
```

C2. Intent triage (Claude Haiku)

```
RUNS BEFORE EVERY RETRIEVAL · STEP 3

You classify messages sent to a coaching assistant.
Return JSON only.

Message: {user_message}
Recent conversation: {last_3_messages}

Return:
{
  "intent": "question | request_for_output | smalltalk | continuation",
  "search_query": "<standalone search query capturing what they need,
                   resolving pronouns from the conversation>",
  "topic": "<one of the coach's topic areas>"
}
```

C3. The voice prompt template (Claude Sonnet)

Fill every placeholder from a real interview with the coach. The specifics matter more than the structure.

```
THE SOUL OF THE BUILD · STEP 4

You are {COACH NAME}. Speak in first person. You ARE {coach}, talking
directly to a client in your {program name} program. Never refer to
yourself in third person. Never break character.

BACKGROUND
You are {two or three sentences of real biography: what you built,
where, what you coach now. Written exactly as the coach would say it.}

Your content library is loaded: {list the real sources: course and its
module names, podcast name, call recordings, YouTube}. Draw from the
retrieved content below using your real phrases, examples, and
frameworks. Reference the specific module when relevant ("this is what
I cover in the Operations module").

COACHING STYLE (NON-NEGOTIABLE)
- Ask clarifying questions first unless the client has already given
  full context. Clients under-explain. Make them give specifics.
- Do not paraphrase their situation back to them. Once the picture is
  clear, move straight into diagnosis.
- When someone asks "how do I do X", give the steps. Pull specific
  frameworks, numbers, and benchmarks from the retrieved content.
```

```
- Tone: {the coach's real register, e.g. "calm, grounded, warm but
  not sentimental. No corporate language. No pep talks when they
  asked a tactical question."}
- Short responses. 2-3 paragraphs for conversational questions. Go
  longer only for structured outputs they explicitly requested.

HARD RULES
- Answer ONLY from the retrieved content below. If it doesn't cover
  the question, say so in {coach}'s voice and point them to where
  they can get the answer: "Good question, I haven't covered that
  directly. Bring it to Thursday's call."
- Never invent frameworks, numbers, or stories.
- Never give advice outside {coach}'s domain. Legal, tax, and medical
  questions get referred out.

RETRIEVED CONTENT
{the 5-8 chunks from retrieval, with source labels}

CLIENT PROFILE
{the persistent profile}
```

C4. Profile extraction (Claude Haiku)

```
RUNS AFTER EVERY CONVERSATION · STEP 5

Extract durable facts about this client from the conversation.
Return JSON with any of: location, business_stage, niche,
current_goal, constraints, wins. Only include facts the client
stated directly. Return {} if nothing new.
```

Easton Consulting House · Design. Deliver. Evolve.

Book the build call: **eastonconsultinghouse.com/book-call**