The Malleable Software Playbook

Your Guide to Auditing Your Stack and Building AI Business Systems You Own

01

What Is Malleable Software?

Malleable software is software you can shape to fit your exact workflow — not the other way around. Traditional SaaS tools are built for the average user, which means they are rarely great for anyone specific. With AI coding tools like Claude Code, you can now build and modify your own business systems without a dev team or a coding background. You stop renting tools and start owning systems.

> 💡 **QUICK TIPS**
>
> - Malleable = customizable, extensible, and yours
> - If you cannot change it when your business changes, it is not malleable

---

02

5 Signs Your Current Stack Is Holding You Back

Run through this quick check. If you answer YES to 3 or more, you are overdue for a malleable software upgrade.

> 💡 **QUICK TIPS**
>
> - You spend time manually copying data between tools every week
> - You are paying for a plan tier just to get one feature you actually need
> - Your tools break whenever one platform updates its API
> - You cannot see all your business data in one place
> - You have bent your workflow to fit the tool instead of the other way around

---

03 The 5-Question Stack Audit

For every tool in your stack, ask these five questions. Two or more NOs means that tool is a candidate to replace with a custom build.

> 💡 **QUICK TIPS**
>
> - 1. Can I export all my data in a format I control?
> - 2. Does it integrate directly with my other tools — no Zapier needed?
> - 3. Can I add a feature if my business needs change?
> - 4. Am I using at least 70% of what I pay for?
> - 5. Would I choose this exact tool if I were starting fresh today?

---

04 What to Build First (Full Playbook Below)

The complete Creative Studio Playbook is included on the following pages. It covers the exact systems to build, the Claude Code prompts to get started, and the step-by-step workflow for each one. Flip to the next page to get started.

> 💡 **QUICK TIPS**
>
> - Content Hub → Client Dashboard → Project Tracker → Social Automation → Analytics
> - Each system takes 1–2 Claude Code sessions to build from scratch

AISOCIALHUB · CREATIVE STUDIO BUILD PLAYBOOK

Creative Studio

Brand URL in. Scheduled posts out.

A step-by-step guide to building Creative Studio as a standalone app — from blank file to a client-ready brand carousel tool deployed on Railway.

FRAMEWORK
CRAFT (5 stages)

IMAGE LAYER
Kei.ai (GPT Image 2 or NanoBanana 2)

THINKING LAYER
Claude Sonnet via OpenRouter

SCHEDULER
Blotato or Zernio

HOSTING
Railway

DATABASE
PostgreSQL on Railway

WHAT YOU'RE BUILDING

A standalone brand-to-carousel app

Creative Studio is a single-page web app. A user pastes any brand's website URL. The app reads the brand's colors, fonts, and product images. The user types a brief and picks how many ideas they want. Claude writes the post concepts and captions. Kei.ai renders the branded carousel slides. Blotato or Zernio schedules them straight to Instagram.

This is the **standalone version** — a fully self-contained app with its own Railway deployment, its own PostgreSQL database, and its own API keys. It is not connected to any larger platform.

5	3	1	~$5
BUILD STAGES	API SERVICES	PAGE APP	TEST BUDGET
CRAFT framework, in order	OpenRouter, Kei.ai, Blotato/Zernio	Single HTML file to start, server added at Rig	Max spend during the Audit stage

WHAT YOU NEED BEFORE YOU START

1 **Claude Code (or any AI coding agent)**
Antigravity IDE or your editor's agent. This is what builds the app — you send prompts, it writes and runs the code.

2 **OpenRouter account + API key**
Provides Claude Sonnet as the Thinker. Free to sign up. Pay-as-you-go. Get key at openrouter.ai → Settings → API Keys.

3 **Kei.ai account + API key**
Provides GPT Image 2 and NanoBanana 2 for slide rendering. Pay-as-you-go credits. Get key at kei.ai → API Keys.

4 **Blotato or Zernio account**
Schedules finished posts to Instagram. Connect your Instagram account. Get API key from Settings → API in either platform.

5 **Railway account**
Hosts the app and the PostgreSQL database. Used at Fire Up stage. Get your API token at railway.com → Account Settings → Tokens.

6 **A test brand website**
Any brand's public website to test with. The original guide uses drinkpoppi.com — any brand with a clear color palette works.

HOW IT WORKS

Three roles, three services

The entire backend is three services, each with one job. Wire them in at the Rig stage. The frontend just talks to these three — nothing else.

THE THINKER

Claude Sonnet

via OpenRouter

Reads the brand URL and the user's brief. Writes post concepts, Instagram captions, and image generation prompts. Picks the right product photo for each slide.

THE DESIGNER

Kei.ai

GPT Image 2 or NanoBanana 2

Takes the approved concept and image prompt. Renders each carousel slide using the brand's palette and product images. Returns slide image URLs.

THE PUBLISHER

Blotato or Zernio

user picks at scheduling time

Takes the rendered slides and the approved caption. Schedules the post to Instagram at the chosen date and time. Confirms the scheduled slot.

Model choice at generation time: The user picks GPT Image 2 or NanoBanana 2 via a toggle on the brief form — before they hit Generate. GPT Image 2 is stronger for photorealistic brand imagery. NanoBanana 2 is faster and cheaper for graphic/illustrative styles.

Scheduler choice at publish time: The user picks Blotato or Zernio in the expanded post modal — after slides are rendered and approved. Both connect to Instagram. Use whichever account is already set up.

THE FULL FLOW

From brand URL to scheduled Instagram post — everything that happens between clicking Generate and seeing the confirmation.

Step by step

1. User pastes brand URL → app scrapes colors, fonts, product images
2. User types brief + selects idea count + picks image model (GPT Image 2 / NanoBanana 2)
3. Claude (via OpenRouter) generates N idea objects: concept, caption, image prompt
4. Ideas appear as cards — user reviews, edits captions if needed
5. User clicks "Generate Slides" on an idea → Kei.ai generates 3 carousel slides
6. User clicks "Expand" → sees all slides + caption in full modal
7. User picks scheduler (Blotato / Zernio), sets date/time, clicks "Schedule Post"
8. Confirmation appears — post is queued in the chosen scheduler's calendar

BUILD FRAMEWORK

Five stages, in order

CRAFT is a five-stage build framework. Each stage is one focused Claude Code prompt. Follow them in sequence — don't skip ahead. The Audit stage can't run until the Rig stage is wired, and so on.

C	R	A	F	T
CREATE	RIG	AUDIT	FIRE UP	TAILOR

STAGE 1 OF 5

C Create

Build the front end — app first, then brand it

Two prompts. The first builds a working app with mock content so every button is clickable immediately. The second reskins it to your design system. Build first, brand second.

PROMPT 1 — THE BUILD

```
build me a single web-page app called CREATIVE Studio. it takes a brand and
turns it into instagram carousel posts i can schedule.

- on the left: i paste a brand's website and it shows me that brand's look -
  their fonts, their colours, their vibe, and their product photos. i can
  change any of it or upload my own
- in the middle: i type what the posts should be about and how many ideas i
  want, hit a button, and i get a list. each idea has the post concept, a
  caption i can edit, a little preview, and a button to generate the slides
- i can open any post up big to see the whole carousel and a Schedule button
- image model toggle: GPT Image 2 or NanoBanana 2 (via Kei.ai)
- scheduler toggle in the modal: Blotato or Zernio

keep it all in one file so it just runs, and drop in some sample content so
i can see it working right away. make it look like a real finished app, not
a rough draft.
```

PROMPT 2 — THE DESIGN

```
now make it look like my brand instead of generic. use this design system:
- dark background #161616, burnt orange #D87657 as the only accent
- DM Serif Display for headings, Plus Jakarta Sans for body text
- cards with subtle borders, generous spacing, dark theme throughout
- a light mode / dark mode toggle i can flip between
make it feel premium — like proper software, not a side project.
```

> **Why two prompts:** Building with mock content first means you can click every button and review the UX before a single API key is connected. Much easier to catch layout issues early.

R

STAGE 2 OF 5

Rig

Connect the Thinker, Designer, and Publisher

Wire the three services to the buttons on screen. The app goes from looking the part to actually working. You'll plug in your API keys via a `.env` file — they never appear in the frontend code.

PROMPT — CONNECT THE SERVICES

```
now make it actually work, not just look the part. connect it to:

THINKER (Claude via OpenRouter):
- on Generate Ideas: call Claude Sonnet via OpenRouter
- pass the brand data (colors, fonts, product images) + user brief
- return N ideas as JSON: { concept, caption, suggestedImagePrompt, productImageIndex }

DESIGNER (Kei.ai):
- on Generate Slides: call Kei.ai using whichever model the user toggled
  (GPT Image 2 or NanoBanana 2)
- pass the brand palette + suggestedImagePrompt + confirmed product image
- generate 3 slides per carousel at Instagram aspect ratio (1:1 or 4:5)
- display generated slides in the card and expanded modal

PUBLISHER:
- in the expanded modal, user has already picked Blotato or Zernio
- on Schedule Post: send slides + caption to the chosen publisher
- show confirmation with queued date/time

keep all keys in a .env file on the server — never in the frontend code.
wire every button so the full flow runs end to end.
```

API keys to collect before this step:

`OPENROUTER_API_KEY` `KEI_API_KEY` `BLOTATO_API_KEY` `ZERNIO_API_KEY`

Store these in your `.env` file. Claude Code will tell you exactly where to paste them. Never commit this file to git.

A

STAGE 3 OF 5

Audit

Hand it a goal — let it test the whole flow itself

Don't click through the app yourself. Hand Claude Code a /goal prompt with hard limits and let it test end-to-end on three different brands. It uses a real browser, clicks through the UI, and schedules real posts.

PROMPT — THE /GOAL TEST

```
/goal

Ask:
test the whole app for real on 3 different brand URLs. spend $5 max on
images. actually schedule at least one post at the end.

Goal:
don't stop until all 3 brands have gone through the full flow — paste the
brand URL, pick a direction, generate ideas, generate slides, expand, schedule.
each brand must look unmistakably like its own brand, not a template.
- click through the UI like a real user. if something breaks, say exactly
  what broke. never fake a result or skip steps.
- verify the model toggle (GPT Image 2 / NanoBanana 2) actually switches
  which Kei.ai model is called.
- verify both Blotato and Zernio scheduling paths work.

Negations:
don't exceed $5. don't post to the wrong account. don't bypass the UI.

Tools:
use a real browser, take screenshots, give one report at the end: each
brand's posts, what it cost, proof they were scheduled, any bugs found.
```

> **Why a goal, not a manual click-through:** Claude Code with a hard budget ($5 cap, no faking, real browser) finds edge cases you'd miss by hand — and proves the full flow works before any client touches it.

F

STAGE 4 OF 5

Fire Up

Deploy on Railway, locked behind a login

Put the app online so a client can use it. Railway handles hosting, auth, and secret key storage. Your API keys are stored as server-side environment variables — never in the code.

PROMPT — DEPLOY BEHIND A LOGIN

```
put this online so my client can use it, locked behind a login.
- host it on Railway — create a new Railway project for this app
- add PostgreSQL on Railway for session storage
- add a simple username and password so only the client can get in
- store all API keys as server-side environment variables in Railway:
  OPENROUTER_API_KEY, KEI_API_KEY, BLOTATO_API_KEY, ZERNIO_API_KEY
  — never in the frontend code
give me the Railway deploy steps, then double-check it works end-to-end
in production: log in → paste a brand → generate ideas → generate slides
→ schedule a post.
```

After deploying: Railway gives you a public URL (e.g. creative-studio.up.railway.app). Share this with your client along with the username/password you set. They access a fully working app — no code, no config needed on their end.

Railway setup order

1. Create new Railway project
2. Add PostgreSQL plugin to the project
3. Deploy the app service from your local folder or GitHub repo
4. Add all API keys as environment variables in the Railway dashboard
5. Set AUTH_USERNAME and AUTH_PASSWORD as env vars too
6. Get your Railway API token (Account → Tokens) and give it to Claude Code
7. Claude Code handles the deploy and confirms the live URL

T

STAGE 5 OF 5

Tailor

Rebrand the template for any client

Once it works, it's a template. Reskin the whole app for any client in one prompt. The engine stays exactly the same — only the look and the output type change.

PROMPT — REBRAND FOR A CLIENT

```
this is my template now. rebrand the whole thing for a specific client.
- pull the client's website (use drinkpoppi.com as the example) for their
  colors, fonts, and logo
- restyle the whole app to match their brand — the top bar, the buttons,
  the cards, the colors, the fonts — but keep how it works exactly the same
- put their name on it (e.g. "Poppi Studio")
- add a Reset Brand button so i can switch to a different client URL later
- if the client wants something other than Instagram carousels — say YouTube
  thumbnails or merch designs — just change what i ask Kei.ai to generate
  and leave everything else alone
don't touch how it works underneath. only change the look and output type.
```

> **This is the payoff.** One build becomes unlimited clients. The CRAFT engine — thinking, rendering, scheduling — is reused every time. You swap the brand URL and the image output type. That's it.

DATABASE

PostgreSQL session table

One table stores everything needed to resume a session, audit what was generated, and track scheduling status. Claude Code creates this automatically during the Fire Up stage.

CREATIVE_STUDIO_SESSIONS		
id	UUID	Primary key, auto-generated
brand_url	TEXT	Client brand website URL
brand_data	JSONB	Extracted colors, fonts, product image URLs
brief	TEXT	User's post direction input
image_model	TEXT	'gpt-image-2' or 'nanobana-2'
ideas	JSONB	Array of {concept, caption, suggestedImagePrompt, productImageIndex}
rendered_slides	JSONB	Array of Kei.ai image URLs per idea
scheduler	TEXT	'blotato' or 'zernio'
scheduled_at	TIMESTAMPTZ	When the post is queued to publish
status	TEXT	'draft' | 'rendered' | 'scheduled'
created_at	TIMESTAMPTZ	Auto-set on insert

THINKER PROMPT TEMPLATE

This is the structure Claude Code will use when calling OpenRouter. Include this in your CLAUDE.md so the agent builds the backend with the right prompt shape.

```
SYSTEM:
You are a creative director for social media content. Generate on-brand
Instagram carousel post ideas. Match the brand's visual identity, voice,
and positioning exactly. Return JSON only — no prose, no markdown fences.

USER:
Brand URL: {brand_url}
Brand palette: {extracted_colors}
Brand fonts: {extracted_fonts}
Product images: {product_image_urls}
Brief: {user_brief}
Number of ideas: {n}

Return a JSON array of {n} objects:
[
  {
```

```
    "concept": "One-line post concept",
    "caption": "Full Instagram caption with hashtags",
    "suggestedImagePrompt": "Detailed image gen prompt using brand palette",
    "productImageIndex": 0
  }
]
```

QA GATE

Done when

Check every item before calling it shipped. Each maps to the CRAFT stage that produces it.

☐ App builds and runs with sample/mock content visible in all three sections	C — Create
☐ Dark theme applied: #161616 bg, #D87657 accent, DM Serif Display + Plus Jakarta Sans	C — Create
☐ Light/dark mode toggle works	C — Create
☐ Brand URL extraction works — colors, fonts, and images pulled from any website	R — Rig
☐ Generate Ideas calls Claude Sonnet via OpenRouter and returns structured JSON	R — Rig
☐ Image model toggle works — GPT Image 2 and NanoBanana 2 both render via Kei.ai	R — Rig
☐ Slides render and display correctly in card view and expanded modal	R — Rig
☐ Captions are editable before scheduling	R — Rig
☐ Blotato scheduling path works — post appears in Blotato calendar	R — Rig
☐ Zernio scheduling path works — post appears in Zernio queue	R — Rig
☐ All API keys stored in .env / Railway env vars — none in frontend code	R — Rig
☐ Self-test passed: 3 brands, each brand-accurate, under $5, post scheduled	A — Audit
☐ Self-test report produced with screenshots and cost breakdown	A — Audit
☐ Sessions persist to PostgreSQL on Railway	F — Fire Up
☐ App live on Railway behind username/password login	F — Fire Up
☐ Production smoke test passed: login → extract → generate → slides → schedule	F — Fire Up
☐ Brand URL + extract flow works for any client website	T — Tailor
☐ Reset Brand button clears extracted brand for next client	T — Tailor
☐ Output type is changeable per client without touching core logic	T — Tailor

SUMMARY

The whole build on one page

	STAGE	WHAT YOU DO
C	**Create**	Build a single-page app with mock content — three sections: Brand Input (URL + extracted palette/fonts/images), Post Brief (direction + idea count + model toggle), Generated Ideas (cards with concepts + captions). Restyle to dark theme with Burnt Orange accent. Two prompts total.
R	**Rig**	Wire the Thinker (Claude Sonnet via OpenRouter), Designer (Kei.ai — GPT Image 2 or NanoBanana 2), and Publisher (Blotato or Zernio). Every key lives in .env on the server, never in frontend code. One prompt connects all three services.
A	**Audit**	Hand Claude Code a /goal prompt. It clicks through the app using a real browser on 3 different brand URLs, renders slides, schedules posts, verifies both model and scheduler toggles, and returns a cost + screenshot report. $5 budget cap.
F	**Fire Up**	Create a new Railway project. Add PostgreSQL. Deploy the app. Store all API keys as Railway environment variables. Add username/password auth. Get a public URL to share with clients. Run a post-deploy smoke test.
T	**Tailor**	Turn the working app into a client-specific tool. Pull the client's brand URL, restyle every surface to match, put their name on it, add a Reset Brand button. Change the image output type (carousels → thumbnails → merch) by changing only what Kei.ai is told to generate.

> **How to use this with Claude Code:** Copy each prompt block in order and paste it into your Claude Code session. Or paste this whole PDF into your agent and say: "Build Creative Studio by following the CRAFT stages in order." Every prompt is copy-paste ready.

Standalone vs. GrowthOS version

This playbook covers the **standalone build** — a self-contained app with its own Railway deployment and API keys. If you want to add Creative Studio as a module inside GrowthOS instead, use the GrowthOS-integrated CLAUDE.md and playbook. The core CRAFT logic is the same; the difference is that the GrowthOS version reuses existing API integrations, injects Brand Clarity context automatically, and deploys with the existing GrowthOS Railway service rather than creating a new one.