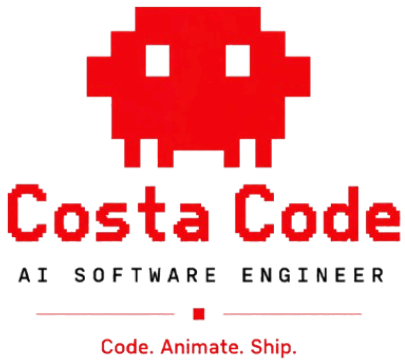


Costa Code

AI SOFTWARE ENGINEER — INSIDE COSTA STUDIO

A complete, local-first AI build platform: a browser app and a terminal tool that share one brain, one database, and one brand.

DOCUMENT
Project Documentation

VERSION
v3 — Web + CLI

DATE
5 August 2026

AUTHOR
Mazen Costa

INDEX

Contents

01 — OVERVIEW

What Costa Code is

Costa Code is a private AI build platform that runs entirely on your own machine. It turns a single prompt into finished work — websites, app screens, animations, documents, schemas, deploys — while staying inside the Costa Studio brand instead of producing generic output you then have to fix.

It is two tools

A **browser app** for working visually, and a **terminal tool** for working fast. They are not two copies of the same thing — they are two front doors into one system.

One shared brain

Notes, tasks, chats, prompts, your profile and your standing instructions all live in one local database. Write a note in the terminal, it is in the browser a second later.

What makes it different from just using an AI chat

A NORMAL AI CHAT	COSTA CODE
Forgets your brand every session	Every reply is pre-loaded with your colours, fonts, code rules and tone
You retype the same briefs	Prompt Engineer stores them; one click to reuse
Cannot touch your files	Runs the real Claude Code CLI locally, with real file access
Your data sits on someone's server	One SQLite file on your own disk
Locked visual design	Site Editor — change any logo, icon, image or word without touching code

The original concept board the platform was built from

02 — ARCHITECTURE

The two tools, side by side

BROWSER

costa-code-web

Runs at `localhost:4560`. Dashboard, chat, prompt library, settings, and the Site Editor. This is where you work when you want to see things.

Start it: `cd costa-code-web && npm start`

TERMINAL

costa-code-cli

The `costa` and `costa-build` commands. Staged project builds, chat, stats, notes, tasks, skills, prompts. This is where you work when you want speed.

Install it: `cd costa-code-cli && npm link`

The flow, end to end

1 You write a prompt browser or terminal

→

2 Brand rules + your instructions are attached automatically

→

3 A model runs it local CLI or an API

→

4 Result streams back and is saved to the shared database

Which models it can drive

PROVIDER	NEEDS	WHAT IT CAN DO
Claude Code (local CLI)	Nothing — uses your existing login	Reads and writes real files on your machine
Anthropic API	`ANTHROPIC_API_KEY`	Sonnet 5, Opus 5, Haiku 4.5 — text only
Google Gemini	`GEMINI_API_KEY`	Gemini 2.5 Pro / Flash
DeepSeek	`DEEPSEEK_API_KEY`	Chat / Reasoner

> **Keys never reach the browser.** They are read server-side from `.env`; the app only ever learns whether a provider is ready, not what the key is.

Brand identity

Every screen, both themes, and the terminal all pull from the same small set of decisions.

Colour

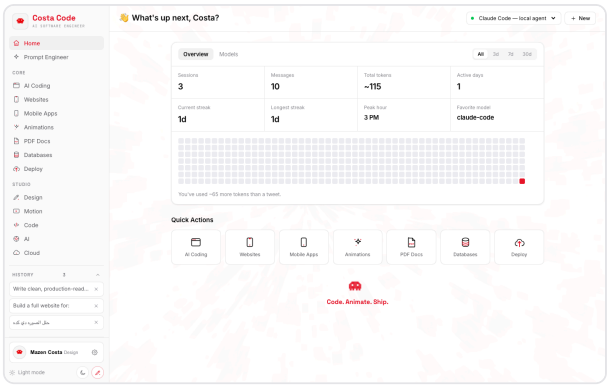
#E8162A	#FF2A2A	#0D0D0D	#0A0A0A	#FFFFFF	#F7F7F8
Accent — light	Accent — dark	Text — light mode	Background — dark	Background — light	Panel

> The rule that keeps it from looking generic: **one red element per component**. Red marks the single most important thing on screen — never decoration.

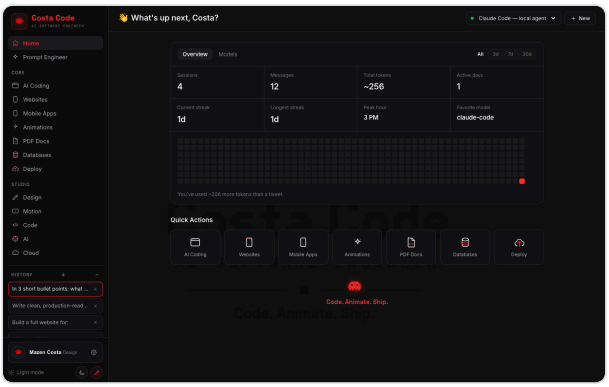
Type

ROLE	FACE	WHERE
Display	Space Grotesk	Brand name, hero headline, section titles
Body	Inter	All interface text
Mono	JetBrains Mono	Code, labels, keys, stats, terminal

Both themes are first-class



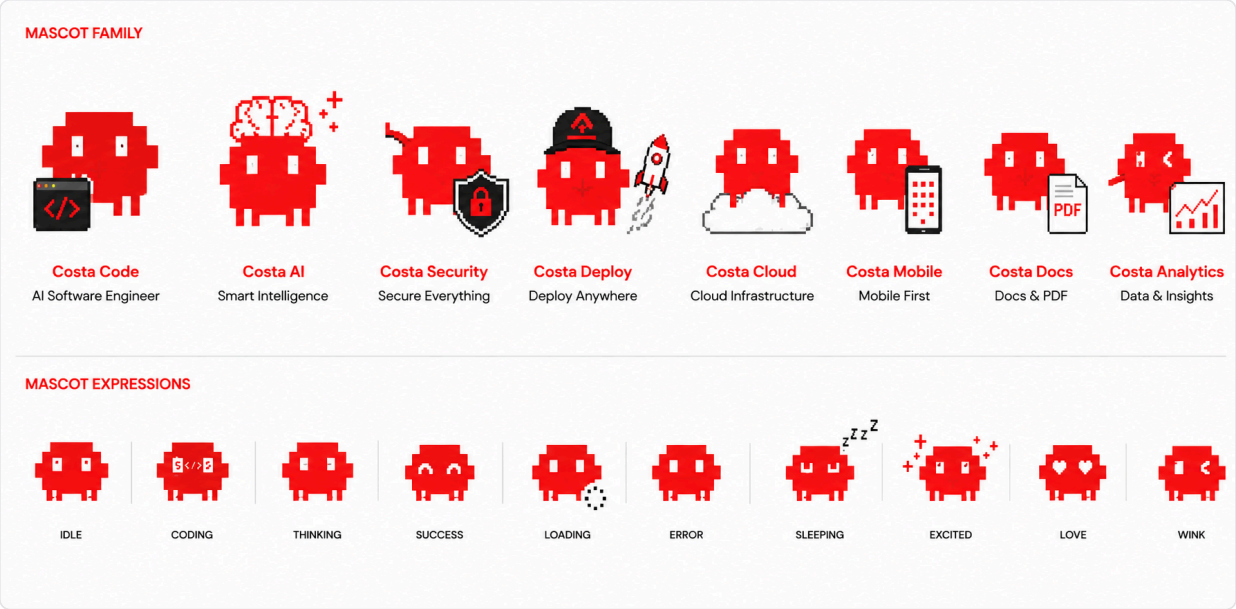
Light mode



Dark mode

The mascot

The pixel mascot is not a static image. It is drawn from a bitmap grid in code, so every expression shares the exact same silhouette — only the eyes and a small accessory layer change. That is why it never looks "off model".



The mascot family and the ten expressions

The ten states, and when each one appears

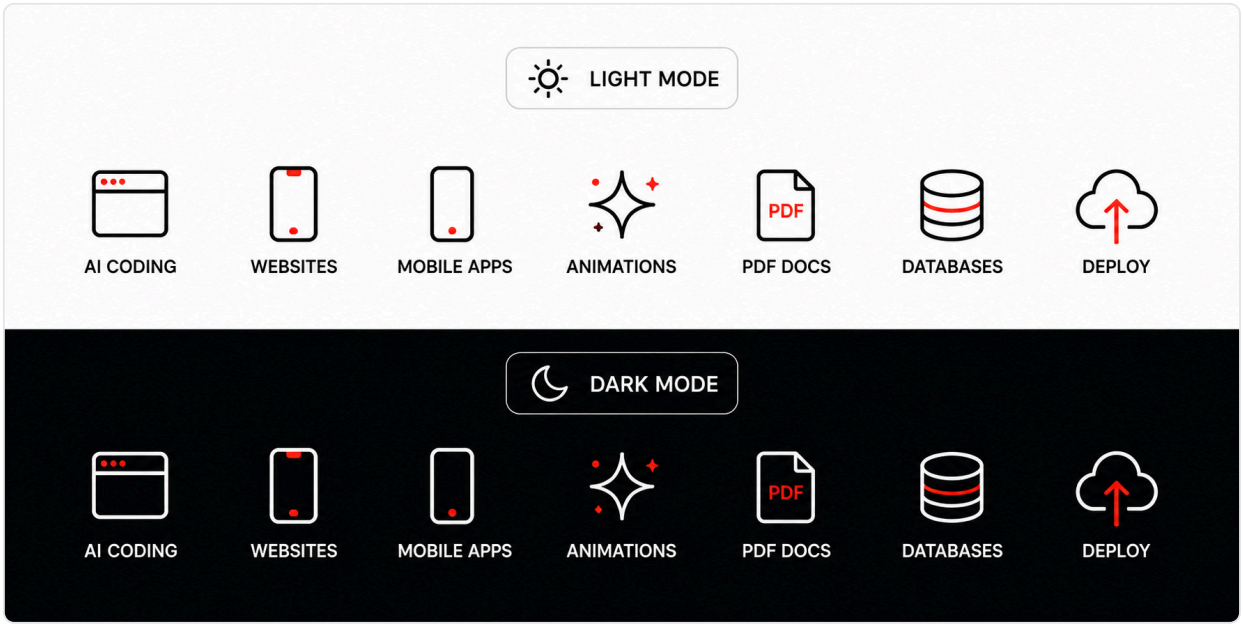
STATE	TRIGGER
Idle	Resting — a slow breathing scale
Thinking	From pressing Send until the first character arrives
Coding	While the reply is being typed out
Success	A reply finished cleanly — brief pop, then back to idle
Error	A request failed — brief shake
Sleeping	Two minutes with no interaction
Loading · Excited · Love · Wink	Available in the set for future states

> The same mascot rides along at the end of the text while a reply types itself out — a small red cursor that walks with the words.

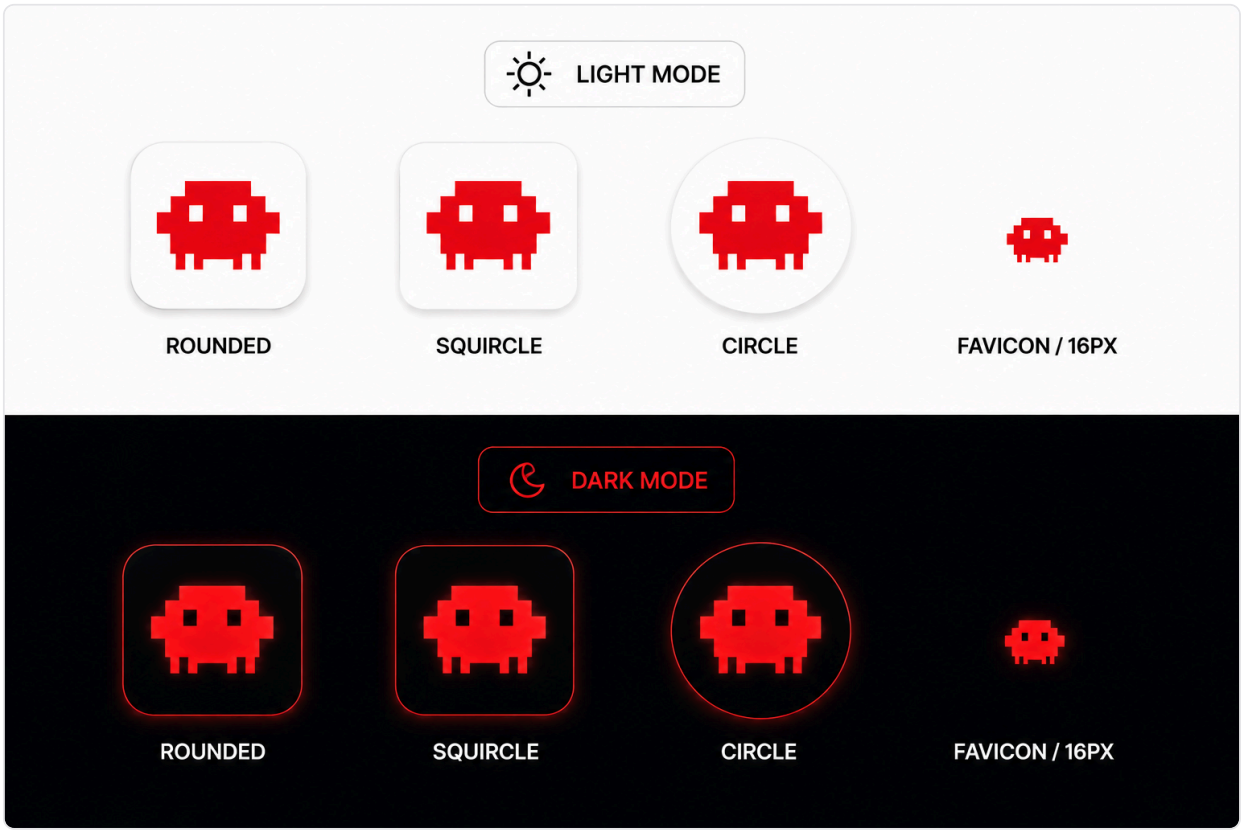
05 — IDENTITY

Icons & logo variants

Every icon is hand-drawn line art rendered as inline SVG — 1.5px strokes, rounded caps, monochrome outline with a single red detail. They inherit the theme colour automatically, so one definition serves light and dark.



The capability icon set in both themes

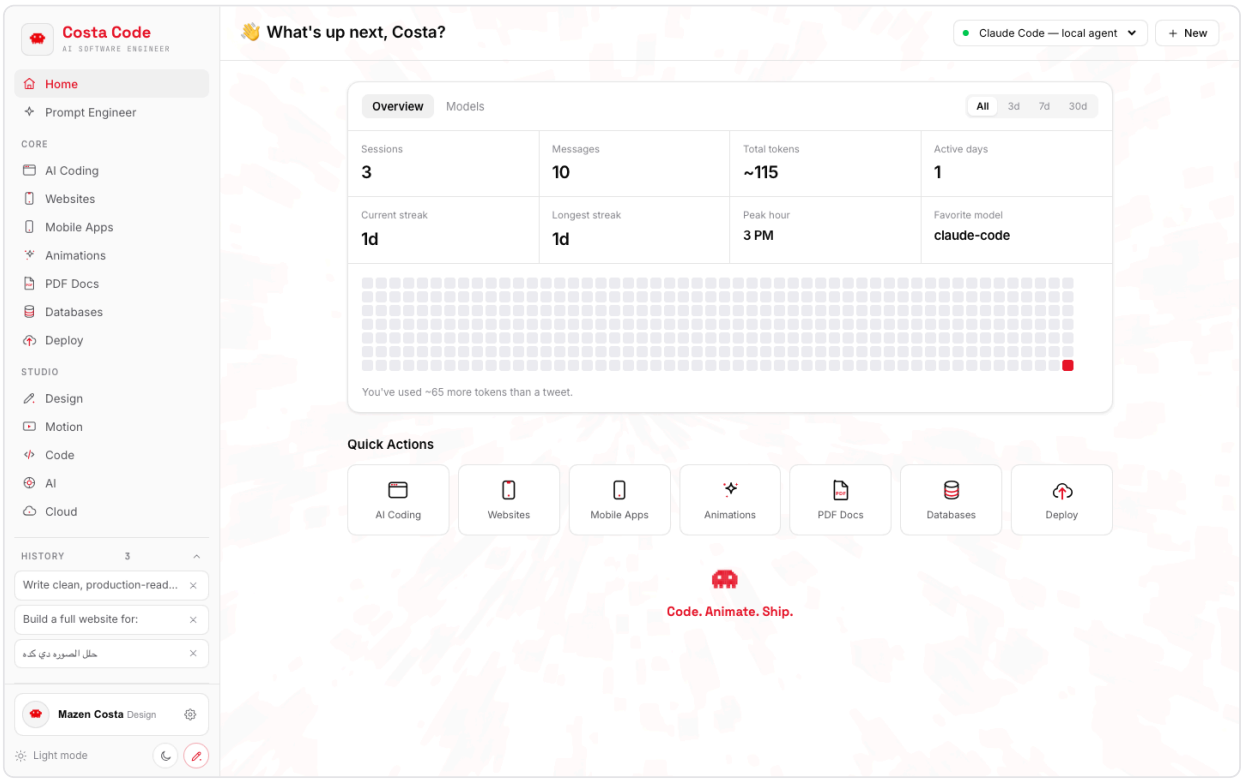


Logo framings — rounded, squircle, circle, favicon — with the dark-mode glow

06 — WEBSITE

The Website — Home

Home is a usage dashboard, not a marketing page. Every number is computed live from your own database — there are no placeholder figures anywhere in this product.



The Home dashboard — stats, activity heatmap, and Quick Actions

What the eight tiles mean

Sessions — separate conversations you have started

Messages — every message, both sides

Total tokens — real usage where the provider reports it

Active days — days you actually used it

Current / Longest streak — consecutive active days

Peak hour — when you work most

Favourite model — most-used model for replies

Honest numbers

The local Claude Code CLI does not report token counts. Rather than invent one, Costa Code estimates at ~4 characters per token and marks the total with a **~**.

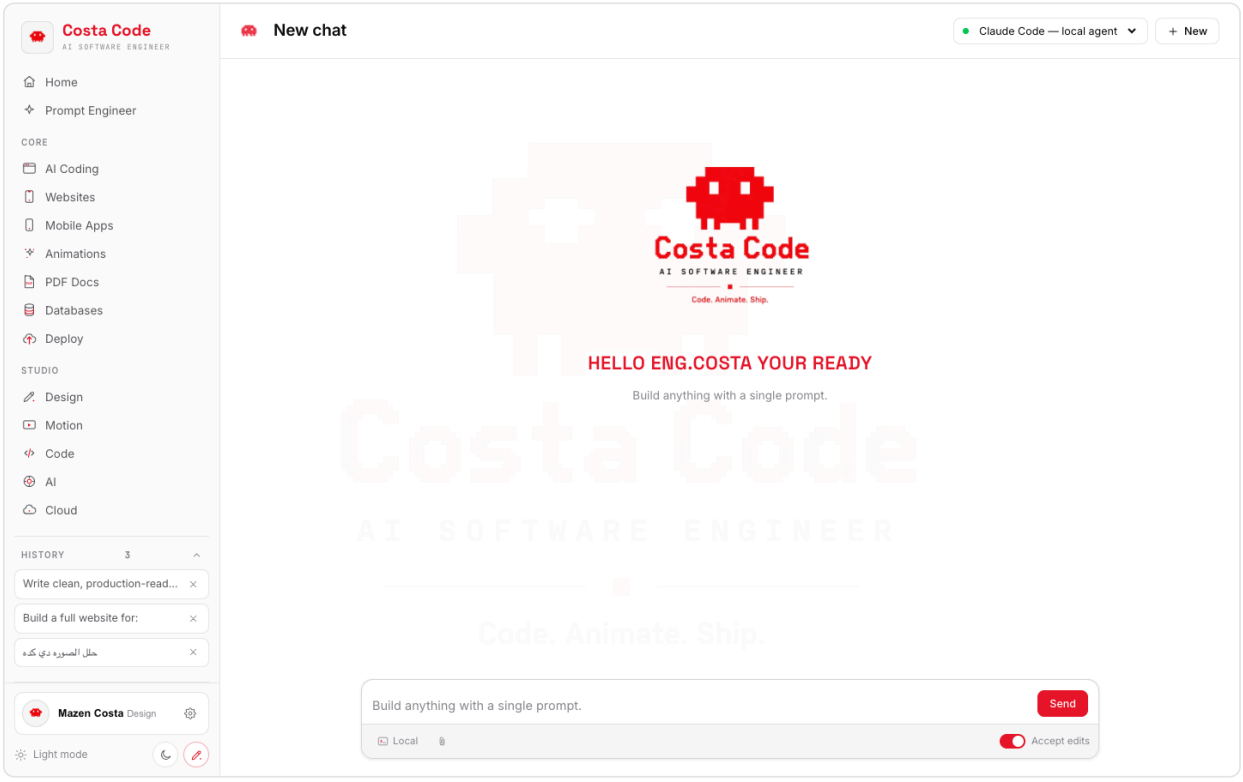
Range buttons (All / 3d / 7d / 30d) re-query the database; the heatmap below shows one square per day, five intensity steps.

Quick Actions — one click each

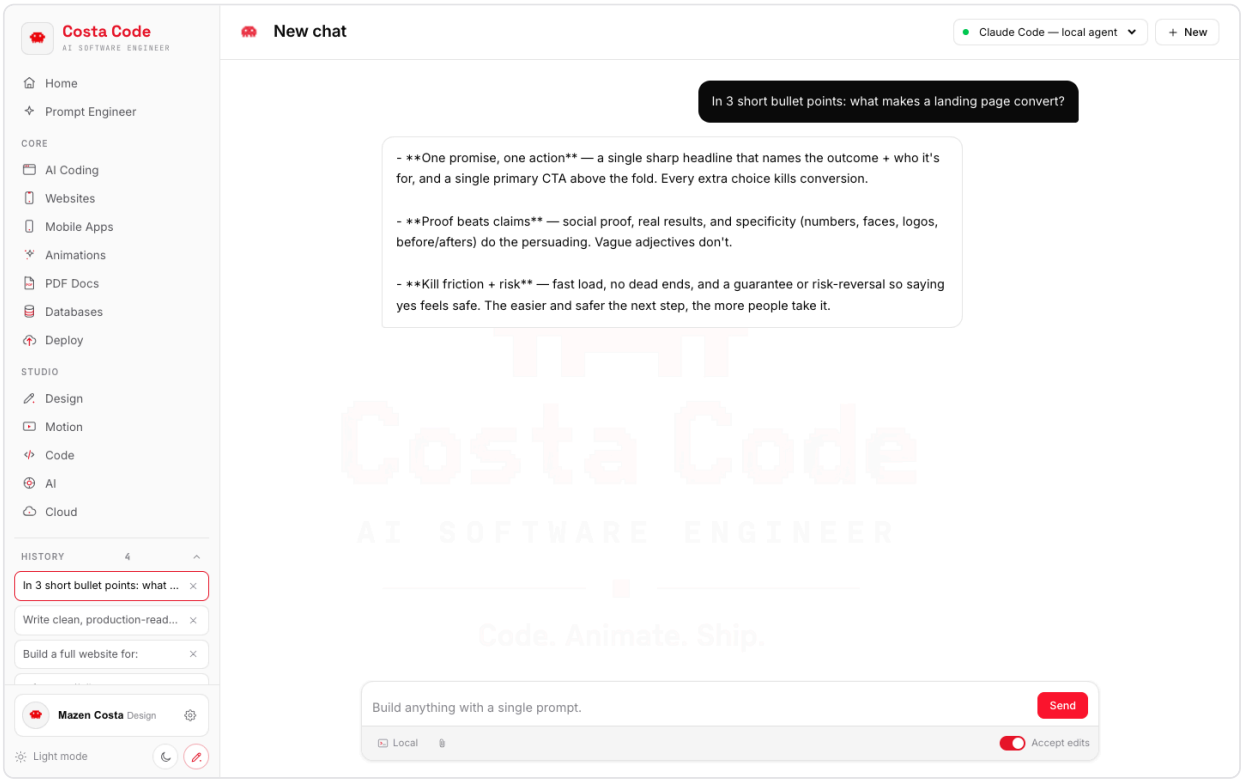
The seven cards drop a matching starter prompt into the composer and jump you to chat. Press Websites and the box already reads `Build a full website for:` with the cursor waiting.

07 — WEBSITE

Chat



A fresh chat — the animated logo lockup and your greeting



A real reply, mid-session

Why replies type themselves out

The composer bar

The local CLI returns its whole answer in one burst. Shown raw, a long reply would simply appear all at once. Costa Code buffers it and reveals it at a steady pace, speeding up when there is a backlog — so it always reads like it is being written.

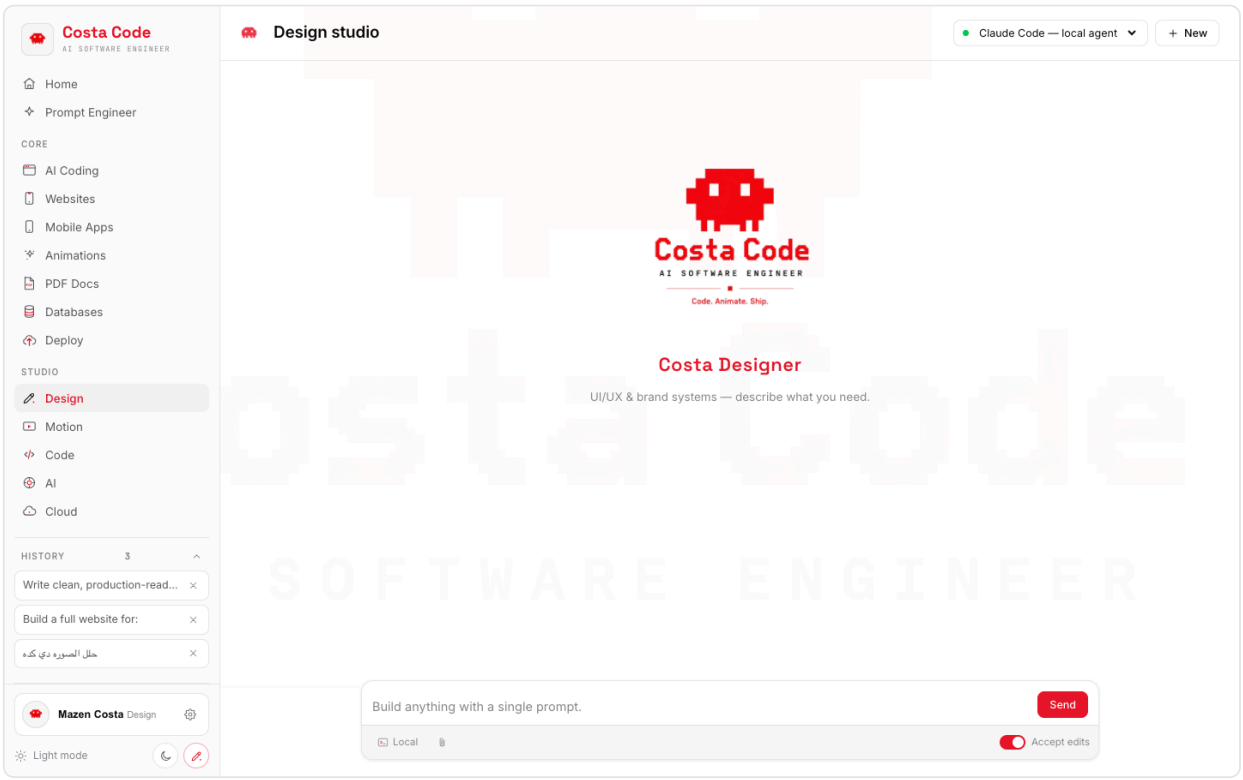
Local — this run happens on your machine

Attach — files, drag-and-drop, or paste a screenshot

Accept edits — let Claude Code edit files without asking each time

Studio agents

Five specialists. Each opens a chat pre-loaded with a different persona, so you are talking to the right kind of expert instead of a general assistant.



Opening the Design studio

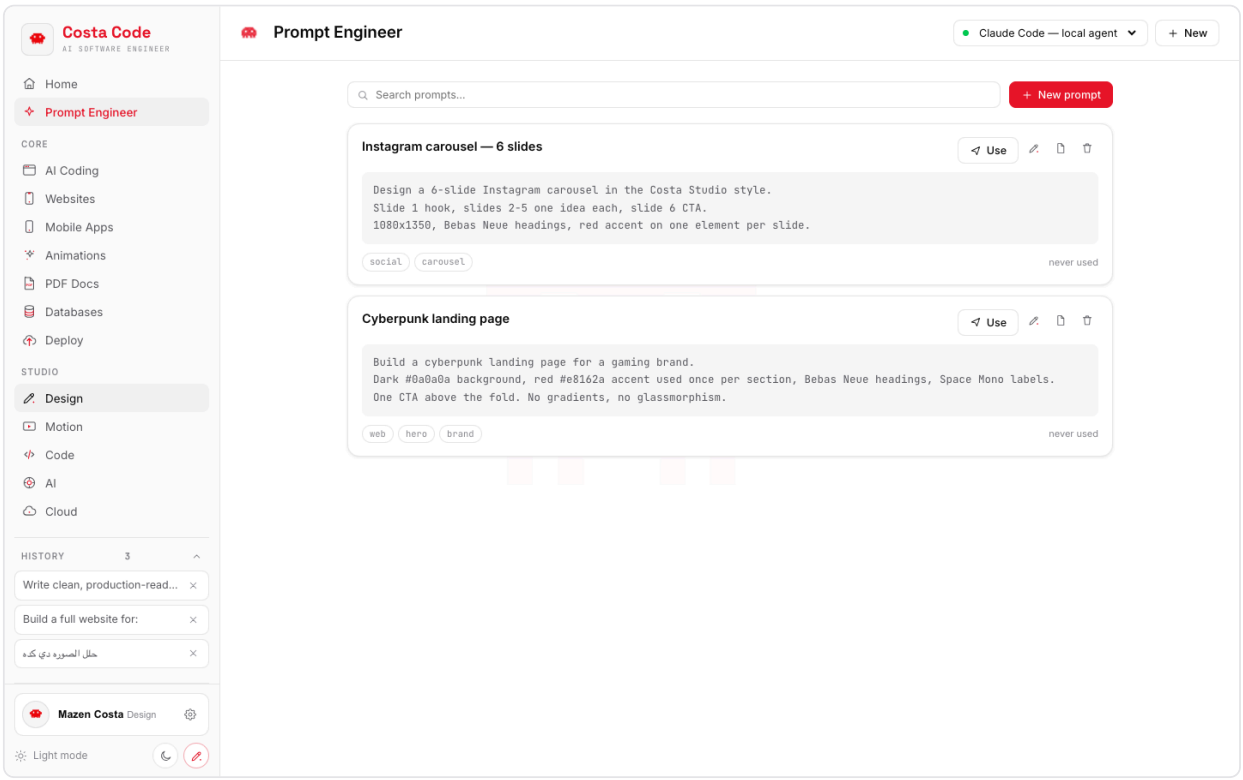
AGENT	PERSONA	GOOD FOR
Design	Costa Designer	Layout, hierarchy, spacing, colour, states
Motion	Costa Motion	Animation, transitions, micro-interactions
Code	Costa Frontend + Backend	The real, complete implementation
AI	Costa AI	Model choice, prompts, agents, cost and latency
Cloud	Costa Cloud	Hosting, builds, secrets, databases, deploys

> These are the same personas the terminal's staged build workflow uses — so an agent behaves identically whether you reach it from the browser or from `costa build`.

09 — WEBSITE

Prompt Engineer

A library for the briefs you keep retyping. Save it once with a title, tags and reference images — then send it into any chat with one click.



The prompt library

Four ways to add an image

- Upload from your machine
- Paste a URL — the server fetches it and keeps a local copy
- Drag and drop onto the editor
- Paste a screenshot straight from the clipboard

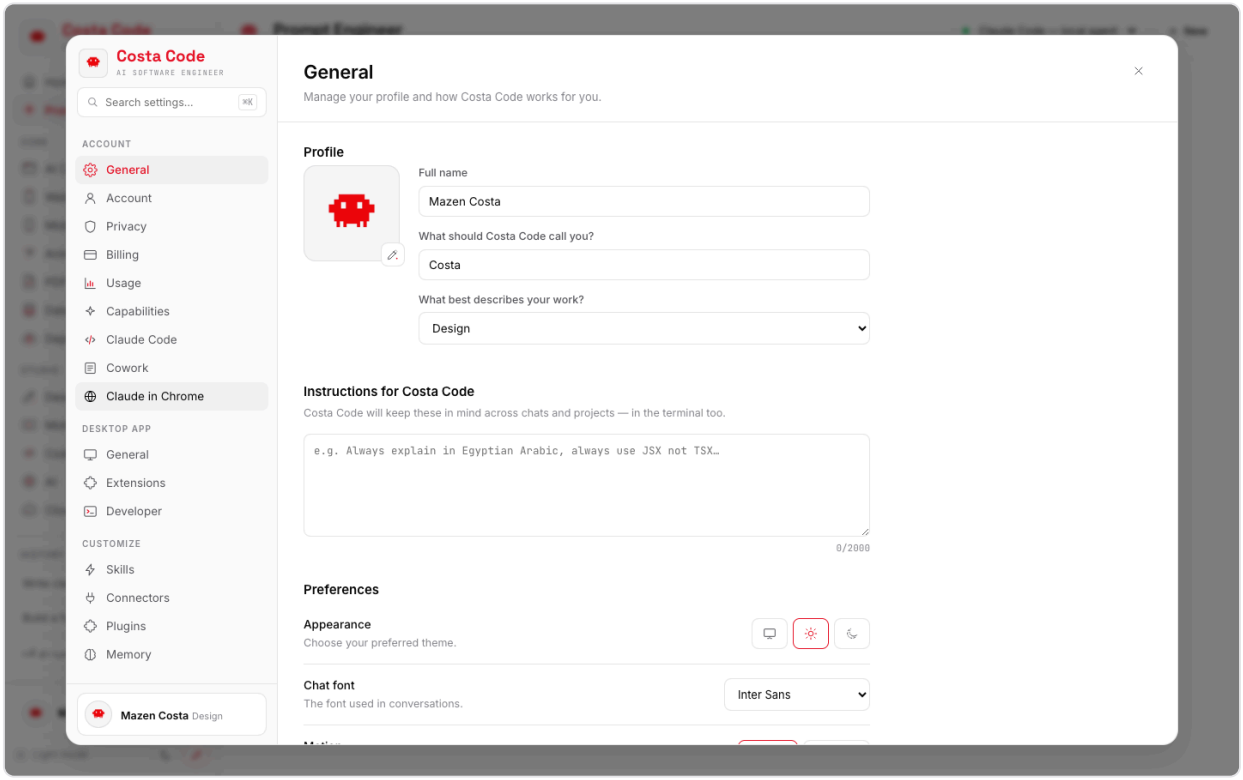
Imported images are stored locally, so a prompt keeps working even if the original link dies.

Every button

- **Use** — opens a new chat with the prompt and its images staged
- **Edit** — change title, tags, body, images
- **Copy** — the prompt text to your clipboard
- **Delete** — with confirmation

Each card tracks how many times it has actually been sent.

Settings



Settings → General — profile, standing instructions, appearance

Instructions that follow you everywhere

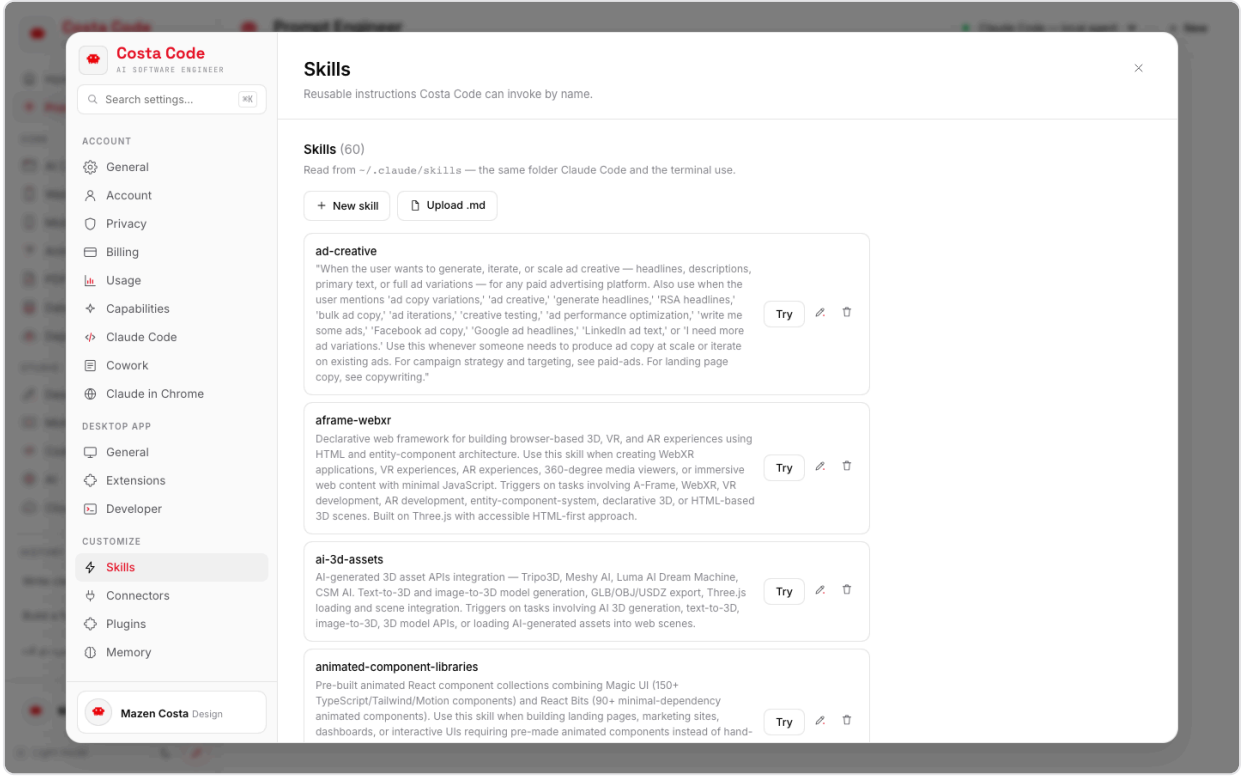
Whatever you write in the Instructions box is attached to every request — in the browser **and** in the terminal, immediately, without restarting anything.

Write "always explain in Egyptian Arabic" once and the terminal starts doing it too.

Sections that actually do something

General · Usage · Capabilities · Claude Code · Skills · Memory · Privacy · Developer.

Sections that only exist in the hosted Claude product — Billing, Cowork, Chrome, Connectors, Plugins — are present in the layout but clearly marked as unavailable locally, rather than shown as buttons that quietly do nothing.

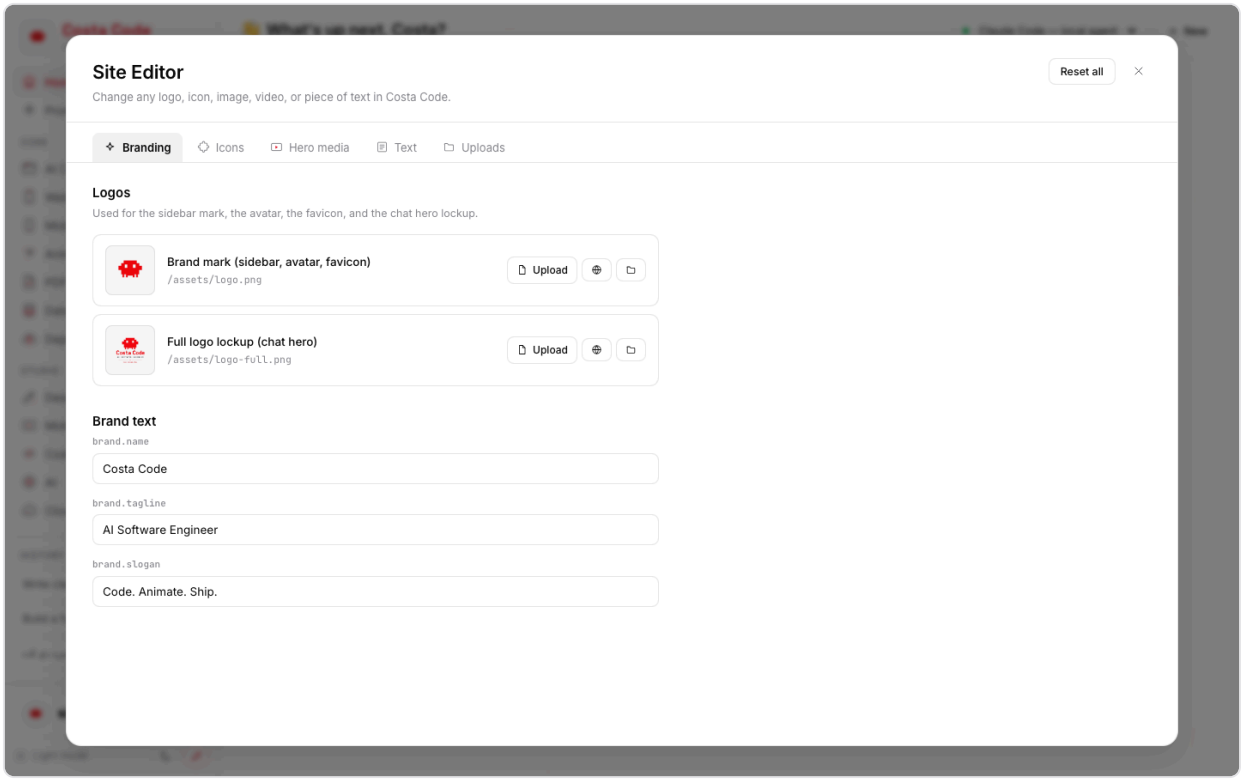


Settings → Skills — full create, upload, edit and delete over ~/.claude/skills

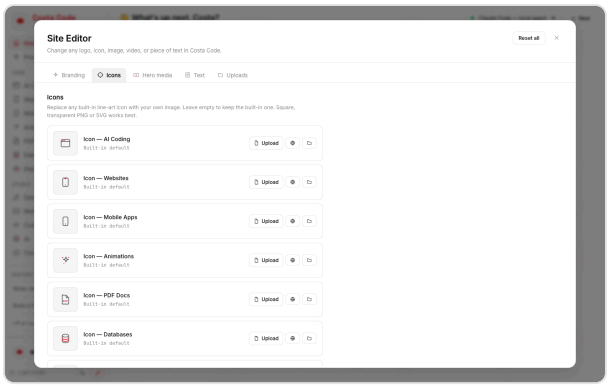
11 — WEBSITE

Site Editor

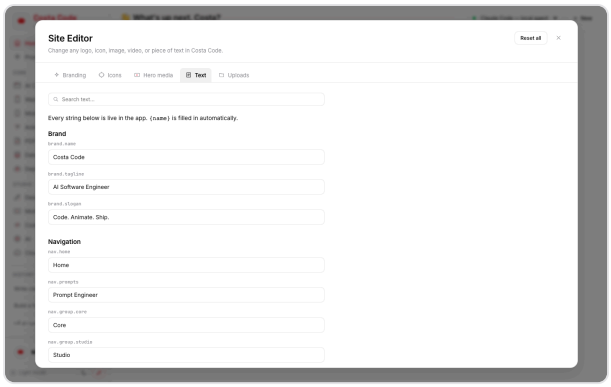
The round red button beside the light/dark toggle. It opens a full admin panel that lets you change **any** logo, icon, image, video or word in the entire app — with no code and no restart.



Branding — swap the brand mark and the hero lockup



Icons — replace any of the 14 built-in icons



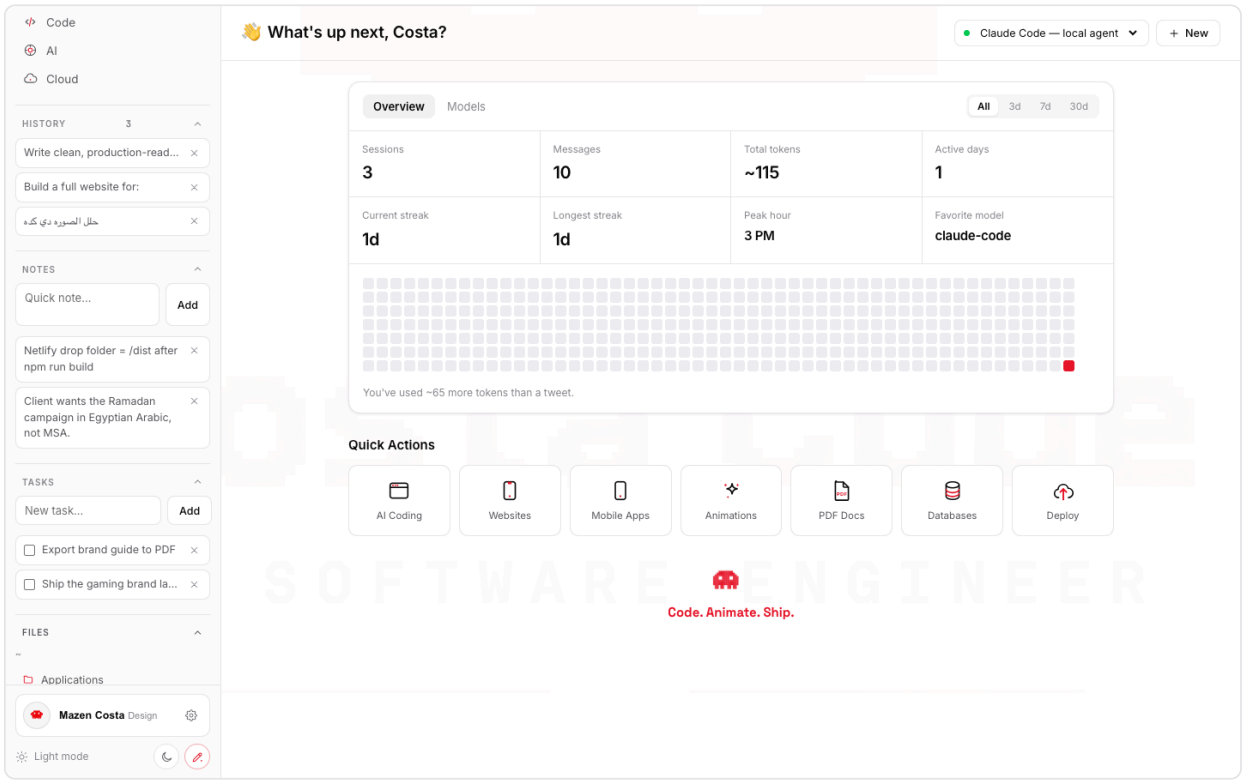
Text — every string in the app, searchable

TAB	WHAT YOU CAN CHANGE
Branding	Brand mark, full lockup, brand name, tagline, slogan
Icons	All 7 capability icons, 5 studio icons, Home and Prompt Engineer
Hero media	The background video or image
Text	Every label, heading, placeholder and starter prompt

Uploads	The library of everything you have uploaded

> Each changed item shows its original value and its own **reset** link, and one **Reset all** button restores the entire app to defaults.

Notes, Tasks & Files



The collapsible sidebar panels

Notes

A scratchpad that persists. Client details, decisions, anything you would otherwise lose in a chat.

Same list as `costa notes`.

Tasks

A simple checklist with open/done state, sorted so unfinished work stays at the top.

Same list as `costa tasks`.

Files

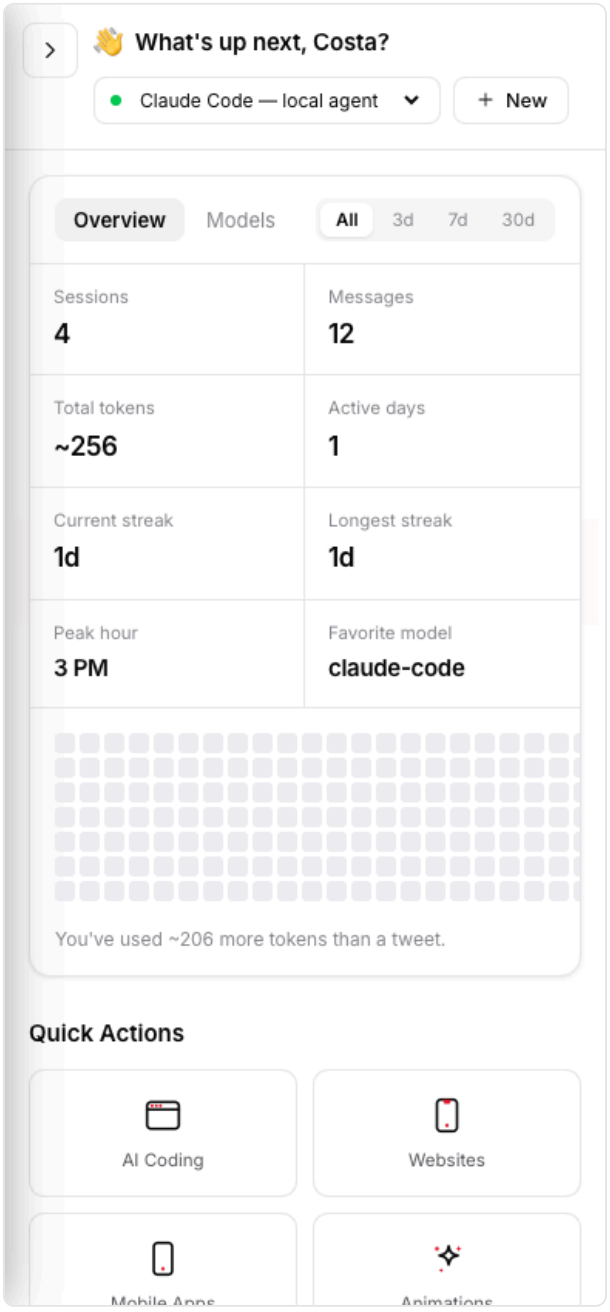
Browse your home directory, preview any text file, and insert a reference into the prompt so the model knows exactly which file you mean.

Read-only by design.

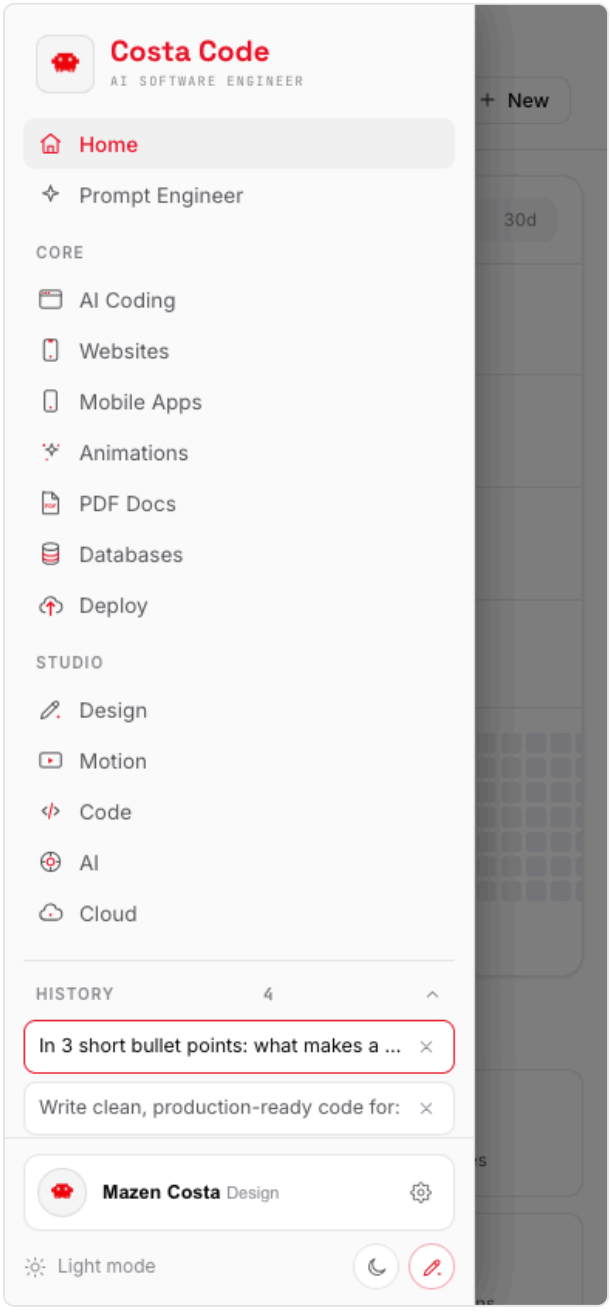
13 — WEBSITE

Light, Dark & Mobile

Three appearance settings: **System** follows your OS live, or force **Light** or **Dark**. The whole interface is built from design tokens, so both themes are complete — not one theme with the colours inverted.



Mobile — the dashboard reflows to two columns



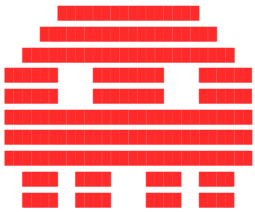
Mobile — the sidebar becomes a slide-in drawer

> It works on your phone over the local network, on the same Wi-Fi — but see the security note at the end of this document before you expose it to anything wider.

The Terminal

Everything the browser does, minus the pictures, plus staged project builds. Install once with `npm link` and the `costa` command is available everywhere.

costa — zsh



Tips for getting started
› Run costa init to start a project from an idea
› Run costa chat to talk to Costa Code right here

What's new
› Notes, tasks & skills — shared with Costa Code Web
› costa chat resumes your last conversation
› Real logo in iTerm2 / VS Code terminals
› Brand system prompt on every claude call

COSTA CODE
AI SOFTWARE ENGINEER — INSIDE COSTA STUDIO

Code. Animate. Ship.

v0.1.0 | Claude Code | costa-code-cli

◆ Build a project:

init <name>	Start a new project from an idea
build [name] [--from <phase>]	Run the build workflow
status [name]	Show build progress
deploy [name] [--live]	Ship it (manual by default)
prompts <list|show|add|rm>	Your saved prompt library

◉ Chat, notes, tasks, skills:

chat [--new]	Shared with Costa Code Web
stats [all|3d|7d|30d]	Your usage dashboard
notes <add|list|rm>	Saved to the shared database
tasks <add|list|done|rm>	Saved to the shared database
skills <list|add|rm>	Manage ~/.claude/skills

★ Shortcuts:

/init	/status	/chat	/build	/tasks	/skills	/deploy
Start a project	Show progress	Start a chat	Run build	Task list	Manage skills	Deploy live

› costa <command> — e.g. costa chat, costa init my-project
costa help to see this again

The real output of `costa help`

The Terminal — daily commands

Your usage dashboard, in the terminal

costa — zsh

```
Costa Code — All time

Sessions          4       Current streak   1d
Messages          12      Longest streak   1d
Total tokens      ~256    Peak hour        3 PM
Active days       1       Favorite model   claude-code

Activity
····························································

You've used ~206 more tokens than a tweet.
```

`costa stats` — *the same numbers as the web Home, with an activity strip*

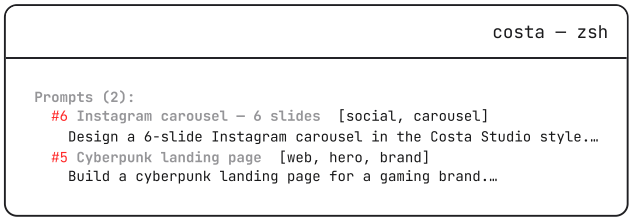
A conversation, without leaving the shell

costa — zsh

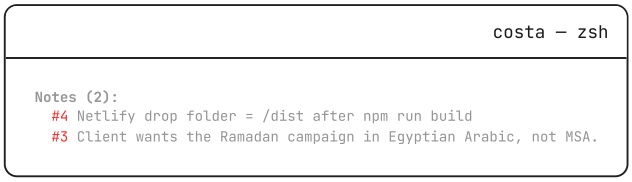
```
costa chat — /new for a fresh conversation, /exit to quit.

You: In one sentence: what is a good CTA?
Costa: A good CTA gives one clear, specific action tied to what the user actually wants — a strong verb plus
the payoff, low friction, no vague "Submit" or "Learn More" (e.g. "Start my free brand audit" beats "Get
started").


You:
```

`costa chat` — *resumes your last conversation; a spinner runs while it thinks, then the reply types out*

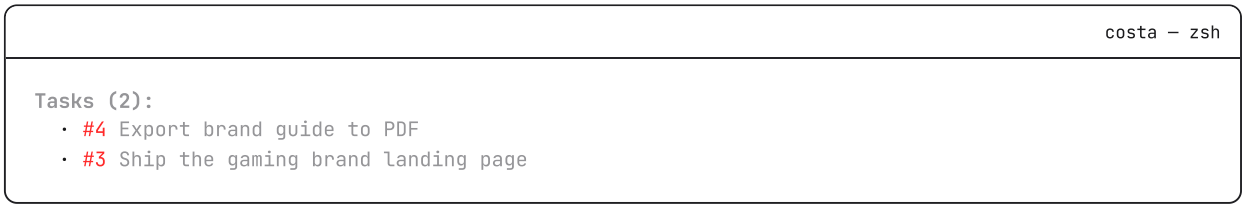
The Terminal — your shared data



costa prompts list



costa notes list



costa tasks list

Staged project builds

The other half of the CLI is `costa build` — an eight-phase workflow where each phase is a separate agent run, handing off through real files in `.costa/` so a failed phase can be inspected and resumed.

01 Analyze Idea	02 Plan Project	03 Design Interfaces	04 Build Code
05 Add Animations	06 Test	07 Documentation	08 Deploy

```
costa init acme-landing --prompt "landing page for a gaming brand"
costa build acme-landing
costa status acme-landing
costa deploy acme-landing
```

> Deploy is deliberately a separate command — publishing should be a decision, never a side effect.

How the two stay in sync

They are not synchronised — there is nothing to sync. Both tools read and write the same file.

Browser app
`localhost:4560`

Terminal
`costa`

One SQLite file
`costa-code-web/data/costa.db`

SHARED	WHERE IT LIVES
Chats and messages	`data/costa.db`
Notes	
Tasks	
Prompt library	
Profile, instructions, theme, Site Editor overrides	
Skills	`~/.claude/skills/` — the same folder Claude Code itself reads

> The terminal loads the web app's own modules directly rather than duplicating them, so the two can never drift apart in schema or behaviour. If the folders are not side by side, set `COSTA_WEB_DIR`.

Try it yourself

```
costa notes add "test from the terminal"
# open the browser → Notes panel → it is already there
```

Code & file structure

costa-code-web

```
server.js                routes: chat, providers, stats, settings, personas,
                         prompts, assets, fs, uploads, skills, notes, tasks
server/db.js             SQLite — notes, tasks, chats, messages, settings, prompts
server/stats.js          dashboard aggregates (streaks, peak hour, heatmap)
server/systemPrompt.js   brand rules + your profile/instructions, composed per request
server/personas.js       the five Studio agents
server/fs.js             home-rooted file browsing, traversal-guarded
server/skills.js         ~/.claude/skills read + create/upload/edit/delete
server/providers/*.js    one file per model provider

public/content.js        every string and asset key — what the Site Editor edits
public/store.js          state, API client, theme, mascot controller
public/mascot.js         the ten generated expressions
public/icons.js          the line-art icon set
public/app.js            bootstrap, router, global events
public/views/*.js        sidebar · home · chat · prompts · settings · editor · panels
```

costa-code-cli

```
bin/costa.js             the multi-command entry point
bin/costa-build.js       the one-shot build entry point
lib/ui.js                box drawing and layout for the terminal UI
lib/logo.js              real inline image where supported, ASCII everywhere else
lib/phases.js            the eight build phases and their agent personas
lib/webApp.js            loads the web app's modules — the shared-brain bridge
lib/commands/*.js        init · build · status · deploy · chat · stats
                         notes · tasks · skills · prompts · help
```

No build step

The browser client is plain scripts — no bundler, no framework, no compile. `npm start` is the entire toolchain.

Two dependencies

The web app runs on `express`, `better-sqlite3`, `multer` and `dotenv`. The CLI adds only `chalk`.

Problems it solves

”I explain my brand every single time”

Colours, fonts, code conventions and tone are attached to every request automatically — plus whatever standing instructions you write once in Settings.

”I keep retyping the same briefs”

Prompt Engineer stores them with tags and reference images. One click sends a brief into a fresh chat.

”The AI can’t see or touch my project”

The Claude Code provider runs the real CLI on your machine with real file access, and the Files panel lets you point at exactly the file you mean.

”My notes, tasks and chats are scattered”

One local database, reachable from the browser and the terminal, with no account and no cloud.

”Changing a label means editing code”

The Site Editor changes any logo, icon, image, video or word live — and can reset any of it.

”I don’t know how much I’m actually using”

A real dashboard: sessions, messages, tokens, streaks, peak hours, per-model breakdown — computed from your own data, never estimated silently.

20 — IMPORTANT

Security — why it stays local

Do not put this on a public server

This is deliberate, not an oversight. Three parts of Costa Code assume the server is your own laptop:

- The **Claude Code provider** spawns a real agent with file-editing permissions for whatever prompt it is given
- The **Files panel** can read anywhere under your home directory
- **Uploads and the Site Editor** write files to disk

On your own machine that is exactly what makes it useful. On a public host it would be remote code execution and arbitrary file read. If you ever want a hosted version, remove the `claude-code` provider and the `/api/fs`, `/api/upload` and asset-write routes first, keeping only the plain API providers.

What it does protect

RISK	HANDLING
Path traversal out of your home folder	Every path is resolved and pinned under the home directory before any read
API keys leaking to the browser	Keys stay server-side; the client only learns ready / not-ready
URL import used to probe your network	Local and private addresses are refused
Deleting a shared skill by accident	Symlinked skill folders are refused
Your data leaving the machine	Nothing is sent anywhere except the provider you pick for that message

Your data, in your hands

Settings → Privacy exports everything as JSON, and can wipe chats, notes or tasks individually.

21 — CHEAT SHEET

Quick reference

Starting up

```
# the browser app
cd costa-code-web && npm install && npm start     → http://localhost:4560

# the terminal tool (once)
cd costa-code-cli && npm link                       → costa, costa-build
```

Every command

COMMAND	WHAT IT DOES
`costa` / `costa help`	The full command board
`costa chat [--new]`	Chat — resumes your last conversation
`costa stats [all|3d|7d|30d]`	Usage dashboard
`costa init <name> --prompt "…"`	Start a staged project
`costa build <name> [--from <phase>]`	Run the eight-phase workflow
`costa status <name>`	Phase progress
`costa deploy <name> [--live]`	Ship it
`costa notes add|list|rm`	Notes
`costa tasks add|list|done|rm`	Tasks
`costa skills list|add|rm`	Manage `~/.claude/skills`
`costa prompts list|show|add|rm`	The prompt library
`costa-build "…"`	One-shot build, no phases

In the browser

WHERE	WHAT
Sidebar → Home	Dashboard and Quick Actions
Sidebar → Prompt Engineer	Your prompt library
Sidebar → Core	Seven starter prompts
Sidebar → Studio	Five specialist agents
Round red button (bottom left)	Site Editor
User card / `⌘K`	Settings

Code. Animate. Ship.