STEP-BY-STEP GUIDE

Build Your Own AI Token Meter

Find out what your AI habit is really worth.

A private app that reads your Claude Code history and shows what it would have cost at pay-as-you-go prices. No code. No API keys. One prompt does the building.

Beginner · 20 min · Claude Code · Free

@learnwithseb

One system at a time. Built, not theorised.

Contents

01 The Game Plan
02 What You Need
03 Get Your Tools Ready
04 Make a Folder and Open It
05 Build the Engine
06 Build the Screen
07 Make the Prices Update Themselves
08 Start It and Set Your Plan
09 Validation
10 Keep Building
11 Quick Reference

The Game Plan

If you pay for Claude Pro, you pay one flat fee a month however much you use. That's a good deal — but you have no idea how good, because nothing ever shows you a number.

The Token Meter shows you the number. It reads the session files Claude Code already writes on your own computer, prices every message at Anthropic's public pay-as-you-go rates, converts it to pounds, and puts a big total on screen.

It answers one question: if I were paying per use instead of a flat fee, what would this month have cost me?

Who this is for

- Anyone on Claude Pro or Max who wants to see what they're actually getting
- People who bill clients for AI work and need a cost per project
- Anyone curious where the money goes in an AI conversation

Why it works

Claude Code keeps a full transcript of every session in a folder on your machine, and every message in it records exactly how many tokens were used. The prices are published openly. That's everything the maths needs — which is why this app needs no account, no key, and no permission from anyone.

© 2026 Learnwithseb — Build an AI Token Meter

NOTE

This guide rebuilds a working app I use daily, as a set of prompts you can paste. The number it shows is the value of your usage at API prices, not a bill. Nothing charges you. On a flat-rate plan it is common to see well over £200 of usage on an £18 plan — that is the point of the exercise.

What You Need

Short list, and most of it is free. If you already use Claude Code, you have everything except the folder.

What	What it's for	Cost
Claude Pro	Includes Claude Code, which builds the app for you	$20/month
VS Code	Where you run Claude Code	Free
Node.js	Runs the meter on your machine	Free
API keys	None. Not one.	-

TIP

This is the rare build with nothing to sign up for. The meter reads local files and one public web page. There is no key to paste, which also means there is no key to leak.

READ THIS FIRST

The meter can only show what Claude Code has already done. If you've never used Claude Code, it will open completely empty — that's not a bug. Use Claude Code normally for a day or two, then come back. It picks up your whole history retroactively.

STEP 01

Get Your Tools Ready

Skip any of these you already have.

© 2026 Learnwithseb — Build an AI Token Meter

Install Node.js from nodejs.org — click the big green LTS button, run the installer, accept every default, then restart your computer. Node is the free software that runs the meter.

Install VS Code from code.visualstudio.com. It's called a code editor, but you won't write any code — it's just where Claude works.

Install Claude Code inside VS Code. Press Ctrl+Shift+X (Cmd+Shift+X on a Mac) to open Extensions, search for Claude Code, install the one published by Anthropic, then click its icon in the left sidebar and sign in with your Claude Pro account.

To check Node worked, open your terminal (Windows: press the Windows key, type cmd, press Enter — Mac: press Cmd+Space, type terminal, press Enter) and run:

```
node --version
```

A version number like v22.0.0 means you're ready. An error means Node isn't installed yet, or you skipped the restart.

NOTE

Claude Code is included in Claude Pro at no extra cost. You do not need an Anthropic API key to use it, and you will not need one for the meter either.

STEP 02

Make a Folder and Open It

Create a new empty folder called token-meter somewhere you'll find it again — your Desktop is fine.

In VS Code, click File, then Open Folder, and choose it. You should see token-meter in the panel on the left, empty. Click the Claude Code icon in the sidebar to open the chat panel. That's your build environment.

COMMON MISTAKE

Open the folder itself, not the folder above it. If you open your whole Desktop, Claude will scatter the app files across it.

STEP 03

Build the Engine

You'll build this in two prompts: the engine that does the maths first, then the screen that shows it. Paste this one into the Claude Code chat panel and press Enter. Watch the files appear on the left. Leave VS Code open.

PROMPT — COPY & PASTE INTO CLAUDE CODE

Build me a local AI Token Meter — a web app that runs on my machine and shows, in GBP, what my Claude Code usage would cost at API list prices. Start with the server and the maths; I'll ask for the interface next.

Use pure Node.js with no dependencies and no npm install — only the built-in http, https, fs, path and os modules. Serve it on port 3200, bound to 127.0.0.1, and reject any /api/ request that doesn't come from localhost.

Where the data comes from: read every .jsonl session transcript under ~/.claude/projects. Keep only the lines where type is "assistant" and message.usage exists, and skip any model called <synthetic>. From each one, record the model, input tokens, output tokens, cache read tokens, cache write tokens (5-minute and 1-hour buckets kept separately) and the timestamp. Dedupe every call globally by message id plus request id — the same call can appear in more than one file and must never be counted twice. Cache the parse per file, keyed on that file's modified time and size, so refreshing stays fast.

Pricing: keep a pricing.json of USD prices per million tokens, matched to a model by the first entry whose "match" substring appears in the model id. Include rows for Claude Opus, Sonnet and Haiku, plus GPT-5, GPT-4o, Grok and Gemini. Convert USD to GBP using a rate fetched once a day from api.frankfurter.app, cached in config.json, falling back to 0.79. Also work out what prompt caching saved me: cache-read tokens times the gap between the full input price and the cache-read price.

Total my usage for today, the last 7 days, this month and all time, plus a daily figure for the last 30 days, and break it down by provider, by model, by project (the session's working directory) and by session. Keep settings in config.json, including my plan cost of £18 a month.

Also create a start.bat that opens the browser and starts the server, and a short README.

TIP

Claude may ask permission to create files or to read your home folder. Say yes — reading ~/.claude/projects is the entire job. Nothing leaves your machine.

© 2026 Learnwithseb — Build an AI Token Meter

STEP 04

Build the Screen

The engine can now answer the question. This prompt gives it a face. Paste it into the same chat, straight after the first.

PROMPT — COPY & PASTE INTO CLAUDE CODE

Now build the interface: one dark page in a monospace font — near-black background, orange #ff8748 accent — with four tabs.

LIVE: a giant pound figure for today's cost, a pulsing dot when a session is currently running, the model in use, cost per hour, time elapsed, call count, and four token boxes (output, input, cache read, cache write). Under that, a bar showing this month's usage against my £18/month plan, cards for today / last 7 days / this month / all time, a 30-day spend chart drawn on a canvas, and breakdowns by provider and by model.

PROJECTS: cost per project, where a project is the session's working directory by default. Let me create a named group that combines several directories and sessions into one project, so a job spanning folders shows as one cost.

SESSIONS: every session with its tokens, calls and cost, filterable by project.

SETTINGS: my plan cost and label, the exchange rate with an auto-update toggle, an editable pricing table, and a form to log usage from other AI tools by hand.

Refresh every 5 seconds so the meter ticks live.

Important framing: I'm on a flat-rate Claude Pro plan, so this figure is the API-equivalent value of my usage, not a bill. Say that on the meter card so nobody panics at the number. Once my monthly usage passes what I pay, show the multiple, like "3.2x value". Also show me what prompt caching saved me this month.

NOTE

Building in two passes isn't a compromise, it's the easier path. A shorter prompt gives Claude less to drop, and you get a working thing to react to before the details are locked in.

© 2026 Learnwithseb — Build an AI Token Meter

STEP 05

Make the Prices Update Themselves

Your meter works now, but its prices are frozen in time. Anthropic changes prices and ships new models, and hand-typed numbers rot — a wrong price makes every total wrong and you'd never know.

This second prompt fixes that for good. Paste it into the same chat, straight after the first.

PROMPT — COPY & PASTE INTO CLAUDE CODE

Now make the Claude prices update themselves, so I never have to type one in.

On startup, and then at most once every 24 hours, fetch Anthropic's public pricing page at https://platform.claude.com/docs/en/about-claude/pricing over HTTPS, with a normal browser user-agent and a 12-second timeout. No API key — it's a public page.

Parse it carefully. Split the HTML on <table and read each table's header cells. Only accept the table that has all of these columns: model, base input, 5m cache writes, 1h cache writes, cache hits, and output. That way you never pick up the Batch API table by mistake. In that table, keep only rows whose model name starts with "Claude", skip anything marked retired, deprecated or legacy, and read the family word (opus, sonnet, haiku, fable, mythos) and the version number. Keep only the highest version per family, so new Claude models get picked up on their own.

Sanity-check every row before trusting it: input above 0, output at least as much as input, input below 1000 and output below 2000. Reject any row that fails.

Cache the result in pricing-live.json with a fetched-at timestamp, and overlay it on top of pricing.json when loading prices, matched by family word. If the fetch or the parse fails, keep the last good prices, record the failure, and retry later. Never overwrite good prices with bad ones, and never quietly show me stale numbers.

In Settings, show a green dot with when prices last synced, or an amber dot saying the bundled fallback prices are in use. Mark the auto-synced rows in the pricing table.

WHY THIS MATTERS

The awkward truth about scraping a web page: it breaks when the page changes. That's why the prompt insists on sanity checks and on keeping the last good prices. A meter that admits it's stale is useful. A meter that confidently shows wrong numbers is worse than none.

© 2026 Learnwithseb — Build an AI Token Meter

STEP 06

Start It and Set Your Plan

On Windows, double-click start.bat in your token-meter folder. A black window opens — that's the server, leave it running — and your browser opens the meter.

On a Mac or Linux, open Terminal and run:

```
cd ~/Desktop/token-meter
node server.js
```

Then open http://localhost:3200 in your browser. Want a one-click launcher on a Mac? Ask Claude Code for one: "Add a start.sh launcher for Mac".

Now click SETTINGS and set your plan cost — what you actually pay each month. Claude Pro is roughly £18 in the UK. On Max, put in your real figure. Leave the exchange rate alone; it updates itself daily. Click Save settings.

That one number is what makes the plan bar mean something. It fills as your usage reaches what you paid. Once it's full, you've broken even, and everything after that is free value.

TIP

Closing the black server window stops the meter. That's all it takes to switch it off — nothing runs in the background, nothing starts with your computer.

Validation

Open http://localhost:3200 and check the following. If today shows £0.00 but All time doesn't, that's fine — it just means you haven't used Claude Code yet today.

- ✓ The All time card shows a figure above zero
- ✓ The SESSIONS tab lists your past Claude Code sessions
- ✓ SETTINGS shows a green dot and a recent price sync
- ✓ The meter card says the figure is API-rate value, not a bill
- ✓ Cache read is the biggest of the four token boxes
- × Everything reads £0.00 and no sessions are listed
- × The page won't load at all

© 2026 Learnwithseb — Build an AI Token Meter

Cache read dominating is correct, not a fault. Claude Code re-sends your whole conversation on every turn, so most of your spend is re-reading what's already been said. Caching makes those re-reads roughly ten times cheaper, which is why the meter shows what caching saved you — it's usually the biggest number on the page.

IF IT'S ALL ZEROS

The meter can't find your history. Either you haven't used Claude Code yet, or your transcripts live somewhere unusual. Ask in the chat panel: "The token meter isn't finding my sessions in ~/.claude/projects — can you check the path?"

IF THE PAGE WON'T LOAD

The server isn't running. Double-click start.bat, or run node server.js in the folder. If it says port 3200 is already in use, it's already running — just open http://localhost:3200.

Keep Building

It's your app now. Open the folder in VS Code, click the Claude Code icon, and ask for changes in plain English. Your settings and history survive every edit.

- Group folders into one project — on the PROJECTS tab, click New project group. Useful when one client job spans several folders.
- Count your other AI tools — ChatGPT and Grok leave no local logs, so paste their token counts into Settings under Add manual usage.
- "Show me my most expensive day this month, and which project caused it"
- "Warn me on the LIVE tab when today goes over £5"
- "Add a light mode toggle"

One genuinely useful thing to notice once it's running: long conversations cost more per message than short ones, because the conversation itself is the input. If a session's cost per hour is climbing, starting a fresh one is cheaper. The meter is how you spot that.

TIP

Check the meter after a big build session, not during. Watching a number tick up while you work is a good way to make yourself timid — and timid prompting is what actually wastes money.

© 2026 Learnwithseb — Build an AI Token Meter

Quick Reference

What	Where
Open the meter	http://localhost:3200
Start it (Windows)	Double-click start.bat
Start it (Mac/Linux)	node server.js in the folder
Stop it	Close the server window, or Ctrl+C
Check Node is installed	node --version
Your Claude Code history	~/.claude/projects
Node.js	nodejs.org
VS Code	code.visualstudio.com
Claude Pro	claude.ai
Prices come from	platform.claude.com pricing page
Exchange rate comes from	api.frankfurter.app
API keys needed	None

Three prompts build the whole thing: the engine in Step 03, the screen in Step 04, and the automatic price sync in Step 05. Everything after that is a conversation.

BuildWithSeb — Build an AI Token Meter

Built by @learnwithseb · One system at a time. Built, not theorised.

© 2026 Learnwithseb. All rights reserved.