David Soden

MARKET ASSESSMENT | AUGUST 2026

Your Low-Code Bill Never Went Down

Power Apps and Retool held seat rates flat for three years and added an AI meter on top. What it costs to build the desktop client instead.

David Soden

Independent analyst and practitioner, enterprise automation and applied AI

davidsoden.com/market-assessment

REPORT DS-2026-1AF0 · 2026 · VERSION 1.0 · PUBLIC

About this report

Every substantive claim in this report carries an evidence classification and, where applicable, a numbered source. Facts are separated from vendor claims, third-party estimates, the author's assessments, and untested hypotheses. Forward-looking sections use scenarios with observable tripwires rather than point forecasts. Stated assumptions are listed before the analysis begins.

Market Assessment reports are published and commissioned at **davidsoden.com/market-assessment**. This document is analysis, not investment, legal, or procurement advice.

Scope and evidentiary standard

This report covers the installed desktop client as a delivery route for internal business applications, and asks whether building one with an AI coding assistant now displaces the per-seat low-code platform that hosts the same applications today. It examines two frameworks in detail, Electron and Tauri, prices what it actually costs to ship and maintain a signed and auto-updating client, and tests three years of published low-code rate cards against archived copies of the same pages.

The category is unusually polluted. Framework comparison articles, "best desktop framework 2026" pages and hosting review blogs carry affiliate revenue and are formatted to read as research. The figure that circulates hardest, that Tauri produces applications 96 percent smaller than Electron, appears in search-optimised comparison pages with no measurement behind it and no stated method. Nothing in this report rests on that class of source. Every number here traces to a project repository, a vendor pricing page, an operating system vendor's own documentation, a standards body, an independent security audit, or a survey with a published sample and fielding window.

Two evidentiary weaknesses run through the whole document and are stated here rather than buried. First, no non-commercial body counts desktop framework adoption. Neither the Stack Overflow Developer Survey nor the JetBrains State of the Developer Ecosystem carries a desktop application framework category, and Tauri has removed its own showcase page, so the only measured adoption signals available are package registry download counts, which count continuous integration runs as readily as developers. Second, the single number that decides the build-versus-rent comparison is annual maintenance labour, and it is not published by anyone. It is estimated here from the author's own reasoning, carries the assessment label rather than the third-party estimate label because no research firm produced it, and is shown as a range rather than a point. No third-party estimate label appears anywhere in this document, because no research firm publishes a number this report needed.

This report reads documentation, pricing pages and published research. It does not benchmark the frameworks, measure deployment outcomes, or test the tools. A hands-on evaluation would be different work and would complement this rather than repeat it.

Label	Definition	How to treat it
FACT	Verifiable in the cited primary source: a filing, press release, official documentation, or standards body announcement.	Reliable as stated. Check the source if the exact wording matters.
VENDOR CLAIM	Reported by a vendor about its own business or product. Self-measured and not audited by any third party.	The vendor said it. Directionally useful, not a measurement. Definitions of the metric are usually undisclosed.
THIRD-PARTY ESTIMATE	A research firm forecast or survey result. Methodology is proprietary and generally not reproducible.	Use for direction and order of magnitude only. Do not build a model on it.
ASSESSMENT	The author's reasoned judgment from the cited evidence. Falsifiable, and the reasoning is shown so it can be argued with.	This is analysis, not reporting. Disagree with it where your evidence differs.
HYPOTHESIS	A proposition offered for testing. Not currently supported by sufficient evidence to assert.	Treat as a research question, not a conclusion. Each one names what would confirm or refute it.
SCENARIO	A structured future state with a stated subjective probability. The probability is the author's judgment, not a calculated figure.	Use the leading indicators attached to each, not the probability. The indicators are observable; the probability is not.

Assumptions this analysis rests on

Four premises are assumed rather than shown. Each is named with the conclusions that fail if it is wrong.

An AI coding assistant can produce a working desktop client and its backend at a small fraction of the historical cost of writing one. This is assumed from current practice rather than demonstrated here. If it is wrong, the first-year build cost rises to conventional software project levels and the entire comparison collapses in favour of the platform, because seat rent was always cheaper than a bespoke build when the build cost six figures.

A representative internal application is one an ordinary employee opens to read, enter and approve records against a backend the organisation controls. It is not a real-time collaborative editor, not a media application, not a hardware integration. If the workload is any of those, the webview divergence section changes from a manageable constraint to a disqualifying one, and the framework choice stops being secondary.

Fully loaded internal engineering cost is around 800 US dollars per day. Every labour figure in this report scales linearly with that number. Organisations in lower-cost regions should scale the crossover point down proportionally, and the direction of the conclusion moves with it.

Published list rates are what mid-sized organisations actually pay. Enterprise agreements discount off list, and the discount is invisible from outside. If a reader's negotiated rate is half of list, every crossover seat count in this report doubles. That is the largest single source of error for any specific buyer, and it is one the reader can correct with information they already hold and this author does not.

The call

ASSESSMENT Build the desktop client. The seat rent for internal low-code has not moved in three years while a second meter, for AI generation, was added on top of it, so an organisation now pays twice by design rather than by accident. Past a small team the client costs less to run each year than the licences it displaces, and the framework choice matters far less than whether IT will sign the binary and push it. For a genuinely small team, keep renting. What would move me is maintenance labour measured from real deployments running hotter than estimated here, because that one unmeasured number decides the whole comparison.

My judgment on the evidence assembled here, nothing more. There is data I can't see and data I may have missed. New evidence can move me, including to the opposite position, and I reserve the right to move.

Executive summary

The argument this report set out to test was that AI-assisted coding collapsed the cost of the last mile while low-code pricing stood still, so organisations pay twice. The archives support the pricing half of that claim more cleanly than expected. They also show something the original framing missed, which is that the second payment is no longer implicit. It is now a separate meter with its own rate card.

FINDING 1 Seat rates did not move. Three snapshots, three years, identical numbers.

> **FACT** Retool's Business plan billed 50 US dollars per builder per month and 15 per internal user per month on the June 2023 archived pricing page, the same on the May 2024 page, and the same today. Power Apps has charged 20 US dollars per user per month across the June 2023, June 2024 and current pages. Dataverse database capacity has been 40 US dollars per gigabyte per month since at least June 2024. The rate card did not respond to anything that happened in AI-assisted development. [22] [23] [24] [25] [26] [27]

FINDING 2 What changed instead was that AI became a meter bolted on top of the unchanged seat.

> **FACT** Retool's current plans allocate AI credits by tier, 250 on Free through 3,000 on Business, sell additional credit packs, and bill agent time by the hour at rates that vary by

model. Microsoft ends seeded AI Builder credits on 1 November 2026 and requires customers to purchase Copilot Credits instead, at one US cent per credit on the pay-as-you-go meter. Neither vendor reduced the seat price to compensate. Paying rent twice is now the published design of the product, not an inference about it. [22] [28] [29]

FINDING 3 Microsoft moved a unit of measurement without moving a rate, and said so in writing.

FACT Copilot Studio's documentation records that on 1 September 2025 the common currency for agents changed from messages to Copilot Credits, and states plainly that there is no change in the quantity per prepaid pack or to the pay-as-you-go rate. This is the clearest available answer to the question of whether the pricing dimension moved: the dimension was renamed and re-denominated while the money stayed where it was. [29]

FINDING 4 The cheap option was withdrawn, not repriced.

ASSESSMENT The Power Apps per-app plan at 5 US dollars per user per app per month, present on the June 2023 pricing page, is absent from the current one. Licensing practitioners report it reached end of sale in January 2026 through a line in a licensing guide changelog rather than an announcement. The direction of travel in this category is consolidation upward into a single premium seat, which is the opposite of repricing away from seats toward usage. [25] [26] [48]

FINDING 5 The crossover is a seat count, and it sits somewhere between twenty and one hundred.

ASSESSMENT Published costs to run a signed, auto-updating Windows desktop client come to roughly 492 US dollars a year: code signing, crash reporting and a paid Workers plan. Everything else is engineering labour, which nobody publishes. At an estimated fifteen engineer-days a year the annual total is about 12,500 US dollars, and the client becomes cheaper than Power Apps Premium at 52 seats and cheaper than Retool Business at 62. At six days the crossover falls to about 19. At thirty days it rises past 100. [17] [32] [34]

FINDING 6 The framework choice is decided by two questions, and neither of them is bundle size.

ASSESSMENT The first is whether the application needs a browser capability that WebKit does not implement, because Tauri binds to the operating system webview and therefore ships WebKit on macOS, iOS and Linux. The second is whether the team ships Linux, where Tauri's own documentation records blank windows, flicker and silent software rasterisation on Nvidia hardware. If both answers are no, Tauri is the better choice. If either is yes, Electron's single bundled Chromium deletes the problem class, and the footprint is the price of that deletion. [8] [9] [10]

FINDING 7 The webview is a rendering surface. Anything touching the machine goes through Rust. This holds on desktop and does not yet hold on mobile.

FACT Tauri's plugin repository marks six official plugins as explicitly unsupported on both iOS and Android, including the updater, and a further eight as unknown, untested or planned, including filesystem, shell, process and websocket. The tick in that table means partially

supported, which the project states directly. The architectural rule is sound where the plugins are complete, and mobile is not where they are complete. [12] [13]

FINDING 8 Signing no longer buys what buyers think it buys.

FACT Microsoft's own documentation now states that Extended Validation certificates no longer bypass SmartScreen, that paying a premium for EV solely to avoid warnings is no longer justified, and that reputation builds only through download volume, taking several weeks and hundreds of clean installs from a wide audience. An internal application with sixty users will never reach that threshold on the consumer path. The same page names the enterprise escape hatches, which is where the answer actually lives. [17]

FINDING 9 The strongest counter-argument is organisational, not technical, and it is real.

ASSESSMENT Nothing in Windows prevents an internal desktop client from being deployed. Trusted intranet locations are not subject to SmartScreen review, IT administrators can submit files for accelerated trust, and application control policies treat a managed installer such as Intune as an allowlisting authority. Every one of those is an IT function. A business unit cannot ship a binary; IT can ship it trivially. The thesis therefore depends on an internal approval that the seat rent was purchasing implicitly. [17] [18]

FINDING 10 The end-user premise is better supported than most claims in this category, and the two available measurements disagree by a factor of four.

FACT Pew Research Center, using a probability-based panel, found in September 2025 that 21 percent of US workers say at least some of their work is done with AI, and in February 2025 that 9 percent use an AI chatbot at work weekly or more while 29 percent had not heard of them. Microsoft's Work Trend Index, a survey the vendor commissioned and published, reported 75 percent of knowledge workers using AI. Both are cited here with their sponsor and method named. [35] [36] [37]

The category has no noun, and that costs money

Six candidate terms were tested against this artefact: a program an employee installs, opens, uses and closes, which renders a user interface over services the organisation owns and which hides whatever model or orchestration sits behind them. None of the six fits, and the reason each fails is instructive.

FACT Only one of the six has a published definition to test against. Local-first software was defined by Kleppmann, Wiggins, van Hardenberg and McGranaghan at Onward! 2019 as software treating the copy of the data on the local device as the primary copy, with servers holding secondary copies, and set out seven ideals including that the network is optional and that the user retains ultimate ownership and control. Every other candidate is a term of art with no authority behind it. [38]

Term	Where it comes from	What it gets right	Why it fails here
Harness	Software testing, where a test harness is scaffolding around code under examination	Conveys something wrapped around a system rather than being the system	Already means something specific and different. A harness is temporary and exists to exercise code. This artefact is the product
Thick client, fat client	Client-server architecture, 1990s, in contrast with thin client	Names an installed application with local storage and offline capability	Describes where processing happens, and in this pattern the processing is remote. Most of these clients are thin in the original sense and reading as legacy in procurement
Rich client	Eclipse Rich Client Platform and the 2000s desktop revival	Correctly names a native shell over a remote service	Near-synonym of thick client, dated, and vague enough that two architects will draw different diagrams from it
Local-first application	Kleppmann, Wiggins, van Hardenberg and McGranaghan, Onward! 2019	Has an actual published definition, which is rare in this space	The definition treats the local copy as the primary copy and requires the network to be optional. Almost none of these clients qualify. Using the term for them misappropriates a precise idea
Single pane of glass	IT operations and infrastructure monitoring, not reporting	Captures the consolidation intent accurately	Names a property, not an artefact. Nobody installs a single pane of glass. See below on its provenance
Agent client	Coined, 2024 onward, no settled usage	Points at what is new, which is what sits behind the interface	Presumes the backend is agentic. Most are not, and the term dates the moment it stops being

ASSESSMENT On single pane of glass, the author's prior belief was that the term came from network operations and IT service management rather than from reporting. The weight of usage supports that. IBM, Fortra and network operations centre vendors all use it for consolidating infrastructure monitoring, and the reporting and business intelligence usage is a later borrowing. The origin claim itself cannot be verified. Every source asserting a first use, almost always dated to network management in the late 1990s, is a vendor glossary page or a search-optimised definition article with no citation behind it. The prior is directionally right and the specific origin is unsourceable through ordinary search.

ASSESSMENT The finding is that the category has no settled noun, and the cleanest available term is the plainest: internal desktop client. It is unclaimed, it means what it says, and it survives contact with a procurement form. Recommending a coinage would be worse, because a term nobody else uses cannot be searched for by the next person who inherits the estate.

ASSESSMENT The absence is not cosmetic. A missing noun means no line item in a software budget, no category in an application allowlisting policy, no row in a configuration management database, and no obvious owner when the person who built it leaves. Low-code platforms have all four, because the vendor supplied the vocabulary along with the licence. That is part of what the seat rent buys, and it is the part buyers notice last.

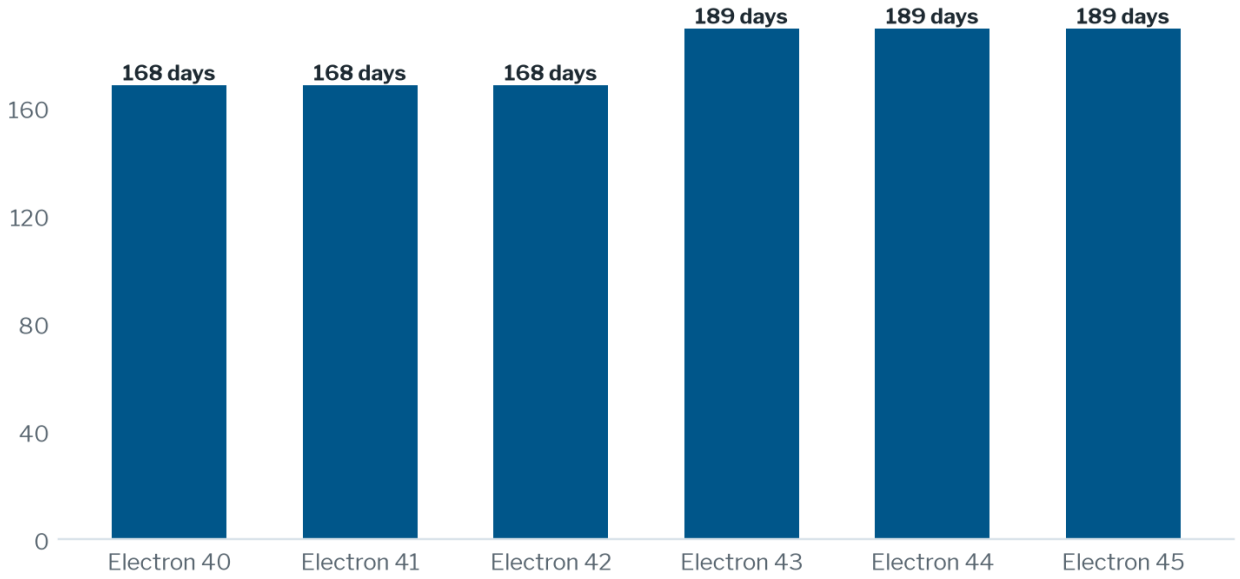
Two frameworks, and what the projects actually say

FACT Electron 44.0.0 reached stable on 25 August 2026, bundling Chromium 152 and Node.js 24.18.1. The project ships a major alongside every other Chromium release on an eight-week cadence and supports the latest three stable majors. Electron 41, stable on 10 March 2026, reached end of life on 25 August 2026. [1] [2]

FACT Tauri 2 reached stable on 2 October 2024. The core crate is at 2.11.5, published 1 July 2026. Tauri does not bundle a browser engine: the project states directly that its applications do not ship a runtime because the final binary is compiled from Rust, and that it uses WRY for webview rendering and TAO for window creation rather than wrapping a kernel. [6] [8]

ASSESSMENT The eight-week cadence is the most under-discussed operational fact about Electron. Because only three majors are supported at once, any given release is out of support roughly six months after it ships. A team that wants to stay on a supported Chromium must therefore take at least two major runtime upgrades a year, forever. That is not a defect, it is the security model working, but it is a standing labour commitment that no comparison article prices. [2]

FIGURE 1 How long an Electron major stays supported



FACT Days from stable release to end of life, computed from the project's published release schedule. Electron 45 is scheduled, not shipped, so its dates may move. The pattern, not the individual figure, is the point: a supported runtime requires at least two major upgrades a year. [2]

Security models, and their histories

FACT Electron's secure defaults arrived over seven years and three major versions. Node integration in renderers has defaulted to off since Electron 5, context isolation has defaulted to on since Electron 12, and renderer sandboxing has been on by default since Electron 20. The project's own security documentation states that displaying arbitrary content from untrusted sources poses a severe security risk that Electron is not intended to handle. [3]

FACT Context isolation is a boundary that has been crossed repeatedly and was crossed again this year. On 27 July 2026 the Electron project published thirteen security advisories in a single batch. They include a high-severity context isolation bypass via prototype hijack, a contextBridge object copy that incorrectly honours setters on the prototype, a DevTools embedder handler that executes arbitrary files via shell open, a spoofable parent-process code-signature check, and a sandboxed iframe that can launch external protocol handlers. [5]

FACT Tauri enforces a trust boundary between Rust and the webview, and gates every crossing through capabilities declared in application configuration, with permissions and scopes enforced at the command implementation. The design intent is that webview code has access only to system resources explicitly exposed through the inter-process communication layer. [11]

FACT Both Tauri generations have been audited independently by Radically Open Security under NLnet and Next Generation Internet funding, and both reports are published in the project repository. The 2022 pre-release audit of version 1 found that the mechanisms protecting against an adversary with script execution were ineffective, that the invoke key could be leaked, and that the default examples encouraged lax content security policy values. The August 2024 audit of version 2 found eleven high, two elevated, three moderate, five low and two informational issues, including the ability to call inter-process communication methods from any displayed origin or frame, and an unauthenticated directory traversal in the development server that exposed the developer's disk to the local network. [14] [15]

ASSESSMENT Publishing an audit that bad before a stable release is the strongest single signal in favour of Tauri's security posture, and it should be read that way rather than as a strike against it. Electron has no equivalent published independent audit, which is not evidence that Electron is less secure; it is evidence that the two projects have different funding structures. Tauri's audits were grant-funded by a European public programme. Electron is maintained under the OpenJS Foundation by contributors whose employers ship the applications, which produces continuous scrutiny and no report to cite. A reader comparing on audit history alone is comparing funding models.

Two engines, not three

This is the technical spine of the report, and the widely repeated framing of it is wrong in a way that matters. Tauri is not a browser wrapper, and it does not use whatever browser the user has set as default.

FACT Tauri binds to the operating system's webview component. On Windows that is WebView2, which Tauri's own reference states is based on Microsoft Edge and therefore Chromium and which can update itself. On macOS it is the system WebKit, which the same reference notes is a core operating system component updated with regular operating system updates, so an unsupported macOS version does not receive WebKit updates. On Linux it is WebKitGTK, version 4.1 per the current prerequisites. [9]

FACT WebView2 is a redistributable runtime and not the Edge browser. Microsoft's distribution documentation states that a production release of a WebView2 application can only use the WebView2 Runtime as the backing web platform, not Microsoft Edge, and gives its reasoning: Edge is not guaranteed to be present, and an administrator pinning the browser version should not pin the application. The same page states that there are separate update policies for Microsoft Edge and the WebView2 Runtime, and that disabling updates for Edge does not affect the availability of the latest WebView2 APIs. [16]

ASSESSMENT The consequence is that a Tauri estate has two rendering engines rather than three, and the split does not follow the lines most teams expect. Chromium runs on Windows and Android. WebKit runs on macOS, iOS and Linux. The practical implication is that the Linux build behaves like the Mac build, not like the Windows one, which is the reverse of how most cross-platform testing is organised. A team that tests on Windows and Linux and assumes macOS will follow has tested one engine twice. [9] [16]

What actually diverges

FACT The device access divergence is not marginal and it is not closing. Web Serial has been supported in Chrome and Edge since version 89 and is not supported in any version of Safari. WebUSB is unsupported across macOS, iPadOS and iOS, and the WebKit standards positions list records the specification as opposed. Web Bluetooth is unsupported in every Safari version. The File System Access picker methods are unsupported in WebKit, which ships only the Origin Private File System. [39] [46]

FACT Storage behaviour diverges in a way that is easy to miss until data disappears. WebKit's published storage policy removes all script-writeable storage for an origin with no user interaction in the last seven days of browser use, which covers IndexedDB, LocalStorage, session storage, media keys, and service worker registrations and caches, and deletes all of an origin's data at once rather than in parts. [40]

Capability	Chromium engine	WebKit engine	What it means for an internal application
Web Serial	Supported from Chrome and Edge 89	Not supported in any version	Any device integration over a serial port has to move to a Rust command. Reachable, but it is work
WebUSB	Supported in Chromium	Not supported on macOS, iPadOS or iOS; the WebKit	Scanner and instrument integration will not port. Design for a native path from the start

Capability	Chromium engine	WebKit engine	What it means for an internal application
		standards position records opposition	
Web Bluetooth	Supported in Chromium	Not supported in any Safari version	Same conclusion. No public WebKit position, so this could change, and nothing suggests it is imminent
File System Access pickers	showSaveFilePicker and showOpenFilePicker supported	Not supported; WebKit ships only the Origin Private File System	Use the Tauri dialog and filesystem plugins rather than the web API. This one has a clean mitigation
Script-writeable storage lifetime	Standard quota and eviction	WebKit deletes IndexedDB, LocalStorage, service worker registrations and caches after seven days without user interaction on an origin	Never treat webview storage as durable. Persist through the Rust side or the backend
Camera and microphone	Standard permission prompt	Requires an Info.plist entitlement and prompts at both application and webview level on macOS 14	Workable, but it is a packaging step a generated project will not have written for you
Media codecs, WebGPU, service worker lifetime	Chromium behaviour, uniform across Windows and Android	Tracks the macOS or distribution WebKit version, which the user does not control	Test against the oldest macOS version and the oldest distribution you support, not against the newest

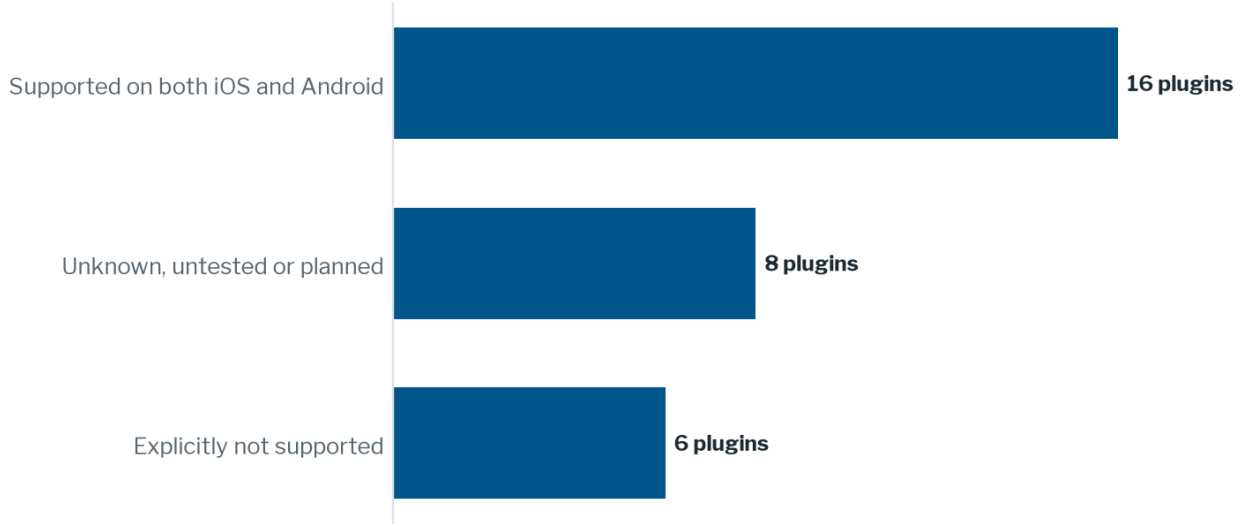
FACT Tauri ships official Rust plugins that bypass the webview entirely for filesystem, HTTP, clipboard, notifications, dialogs, shell, global shortcuts, store, SQL and updates. Because these are Rust commands invoked over the inter-process communication layer rather than web platform APIs, their behaviour does not depend on which engine renders the interface. [12]

ASSESSMENT The architectural rule this implies is worth stating plainly, because it converts a scary compatibility matrix into a design constraint a team can hold in its head. The webview is a rendering surface. Anything that touches the machine goes through Rust. Adopted at the start of a project this rule removes most of the divergence table above at no cost, because a team that never reaches for the File System Access API never discovers that WebKit lacks it. Adopted after the interface is written it is a rewrite. This is the single highest-value decision in a Tauri project and it is made in the first week.

FACT The rule holds on desktop and not yet on mobile. The plugin repository marks the updater, command line interface, autostart, positioner, single instance and window state plugins as explicitly unsupported on both iOS and Android. Filesystem, global shortcut, localhost, persisted

scope, process, shell, stronghold and websocket are marked unknown, untested or planned on mobile. The project states that a tick in that table means partially supported. [12]

FIGURE 2 **Official Tauri plugins by mobile support status**



FACT Counted from the plugins-workspace repository README on 25 August 2026, across 30 official plugins. The supported column is generous: the project defines its tick as partially supported, and five of the sixteen are mobile-only plugins whose desktop status is itself marked unknown. Read this as the shape of the gap, not as a precise score. [12]

ASSESSMENT The absence of a mobile updater is the item that should change a plan rather than merely inform it. In-app updates are the reason to ship an installed client instead of a web page, and on iOS and Android the mechanism does not exist in the official plugin set. Tauri's mobile targets are real and the applications work; the distribution story on mobile routes through the app stores and their review, which is a different operating model from the desktop story this report is about. [12] [13]

Linux behaves like the Mac build

FACT Tauri's own reference documentation states that the diverse nature of the Linux ecosystem makes it very hard to compile accurate information about WebKitGTK across distributions, and publishes an explicitly incomplete table of distribution-to-WebKit-version mappings, recommending developers check their own distribution's repository. [9]

FACT The blank window and hardware acceleration failures are documented by the project, not merely present in issue trackers. Tauri publishes a Linux Graphics Issues page describing windows that open but stay blank or white and windows that flicker while resizing, attributes most of them to the WebKitGTK DMABUF renderer requesting buffer formats the Nvidia driver does not provide, and gives a graduated set of workarounds ending in disabling accelerated

compositing entirely. The same page records that WebGL2 context creation succeeds even when the result is backed by a software rasteriser. [10]

ASSESSMENT That last sentence deserves separate weight. A silent fallback to software rendering is worse than a crash, because a crash is a support ticket and a slow application is a reputation. It is also invisible to the developer, who is on a supported configuration. Any Tauri project shipping Linux needs a non-WebGL fallback path and a way to detect which path a given user is on, and neither of those is generated for you. [10]

ASSESSMENT Electron's counterposition is straightforward and should be priced honestly rather than dismissed on footprint. One bundled Chromium on every platform deletes this entire problem class: the divergence table above becomes empty, the Linux graphics page becomes irrelevant, and a bug reproduced on the developer's machine reproduces on the user's. The price is a larger download, a larger install footprint, and the two-upgrades-a-year runtime treadmill. For an internal application distributed over a corporate network to managed machines, download size is close to a non-issue and the treadmill is the real cost. Teams choosing Tauri to save 100 megabytes on an internal deployment have optimised the cheapest variable in the problem.

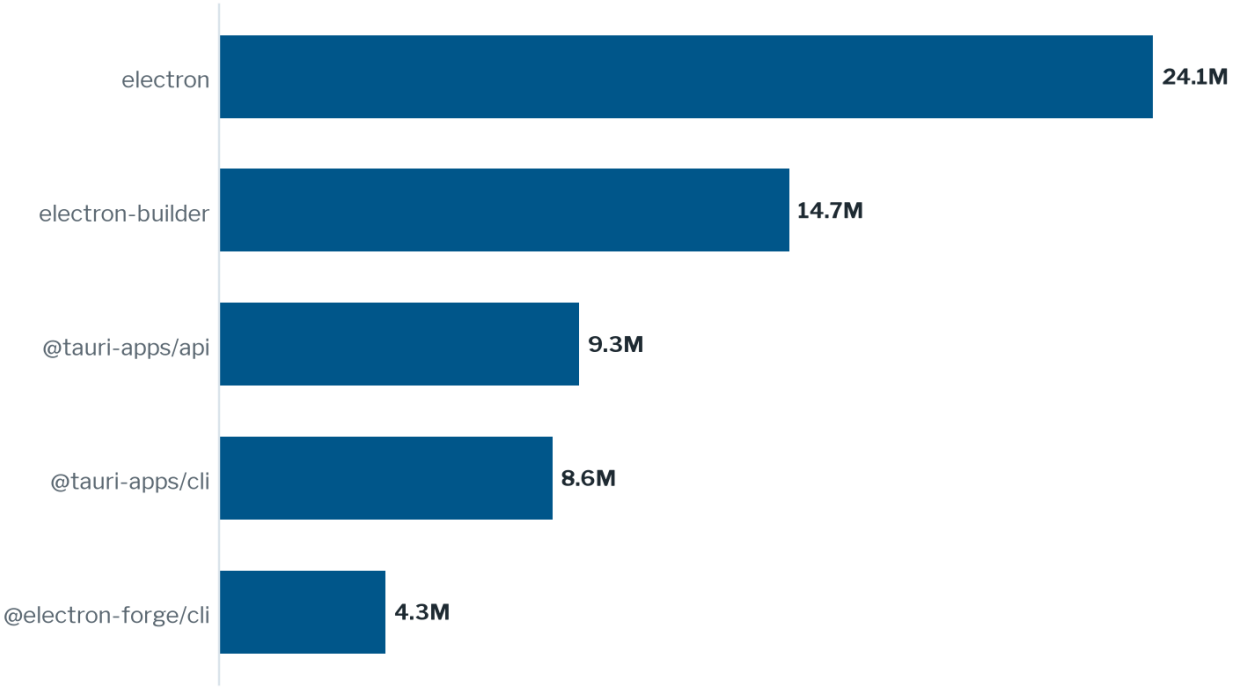
What adoption can be measured, and what cannot

ASSESSMENT There is no non-commercial count of desktop framework adoption. The Stack Overflow Developer Survey covers languages, databases, cloud platforms, web frameworks and development environments, and has no desktop application framework category. The JetBrains State of the Developer Ecosystem, which surveyed 24,534 developers across 194 countries between April and June 2025, likewise does not break out Electron against Tauri. Tauri removed its own showcase page. The absence is itself the finding, and it is why the percentage figures that circulate in comparison articles have nothing behind them.

ASSESSMENT The figures that do circulate should be treated as marketing copy. A Tauri adoption increase of 35 percent year on year, repository growth of 55 percent, and applications 96 percent smaller than Electron all appear in search-optimised comparison pages and syndicated blog posts with no method, no base and no measurement. None of them traces to a primary source. They are reproduced here to identify them, not to use them. [47]

FACT The one measured signal available is package registry downloads. Over the thirty days to 24 August 2026, npm recorded 24.06 million downloads of electron, 14.68 million of electron-builder, 9.27 million of @tauri-apps/api, 8.58 million of @tauri-apps/cli and 4.28 million of @electron-forge/cli. On crates.io, the tauri crate has 27.28 million downloads in total and 10.28 million in the trailing ninety days. [6] [7]

FIGURE 3 **npm downloads, 26 July to 24 August 2026**



FACT Retrieved from the npm registry download API on 25 August 2026. These are downloads, not installations, users, or applications. Continuous integration pipelines dominate the counts and a single project can generate thousands per month. Use the ratio, roughly three to one on the primary packages, as a floor on relative ecosystem size and nothing more. [7]

FACT Named production applications are citable for Electron and largely are not for Tauri. Electron maintains an official curated showcase listing Visual Studio Code, Slack, Signal, Discord, Figma, Microsoft Teams, Notion, Obsidian, GitHub Desktop, Postman and Trello among hundreds of entries. Microsoft's own Visual Studio Code documentation refers to the Electron shell used by Visual Studio Code. Tauri directs enquiries to a community-maintained awesome list rather than an official showcase. [4] [12]

ASSESSMENT An organisation choosing Electron can point a risk committee at named applications from named companies. An organisation choosing Tauri cannot, and will have to argue from the code, the audits and the download counts instead. That is a governance cost rather than a technical one, and for a first internal deployment it may be the deciding argument in the room even though it says nothing about which framework produces a better application.

What it costs to ship a desktop client

Nobody computes this, which is why the build-versus-rent conversation usually ends in assertion. The costing below is for a signed, auto-updating Windows client distributed to managed endpoints inside one organisation. Published rates and author estimates are marked separately, and the estimates are the ones that matter.

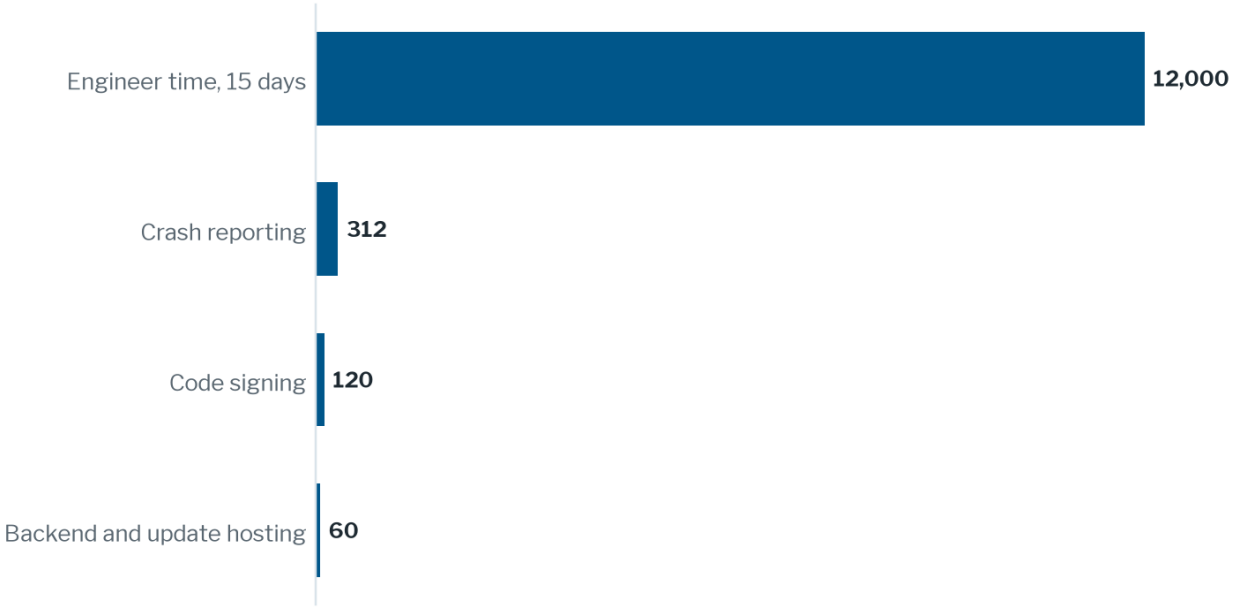
FACT Windows code signing has a hardware floor that did not exist before mid-2023. The CA/Browser Forum's Code Signing Baseline Requirements, effective 1 June 2023 under ballot CSC-17, require subscriber private keys for both Extended Validation and non-EV code signing certificates to be generated and stored in a hardware cryptographic module certified to at least FIPS 140-2 Level 2 or Common Criteria EAL 4 augmented. Software-held keys are no longer permitted. [19]

FACT Microsoft's Artifact Signing, formerly Trusted Signing, is the route around the hardware token. Microsoft's SmartScreen documentation states it starts at 9.99 US dollars per month, requires no hardware token, integrates with continuous integration pipelines, and validates the customer's identity before issuing certificates. The Azure pricing page for the service renders its rates dynamically and the published tier prices could not be verified from the page or from its archived copies, so the 9.99 figure is taken from Microsoft's documentation rather than from the rate card. [17] [21]

FACT Apple Developer Program membership is 99 US dollars per year, and organisations must supply a D-U-N-S number registered to the legal entity. Notarisation is included in membership rather than charged separately. This line applies only to a build that ships macOS or iOS. [20]

Line item	Annual	Basis	Evidence
Code signing, Artifact Signing Basic	$120	$9.99 per month, 5,000 signatures included	Published
Crash reporting, Sentry Team	$312	$26 per month billed annually	Published
Backend and update feed, Cloudflare Workers Paid	$60	$5 per month, 10M requests included	Published
Update artefact hosting	$0	GitHub Releases, or inside the Workers plan	Published
Intune packaging and distribution	$0 marginal	Plan 1 at $8 per user per month, already held via Microsoft 365 E3 or E5 in most organisations	Published
Apple Developer Program	$99	Only if the build ships macOS or iOS	Published
Runtime upgrade treadmill, 8 days	$6,400	Two Electron majors per year plus dependency patching, at $800 per day	Author estimate
Signing pipeline and installer maintenance, 3 days	$2,400	Certificate rotation, installer regressions, packaging changes	Author estimate
Client-attributable support and incidents, 4 days	$3,200	Excludes backend and application logic support	Author estimate
Total, Windows only, steady state	$12,492	Of which $492 is published and $12,000 is estimated	Mixed

FIGURE 4 **Annual cost to run a signed, auto-updating Windows client**



ASSESSMENT US dollars per year. Three of the four lines are published vendor rates and are almost invisible against the fourth, which is the author's estimate and which nobody publishes. That asymmetry is the finding: the decision rests on the one number in this chart that cannot be looked up. [17] [32] [34]

ASSESSMENT Year one is separate and larger. Getting from a generated scaffold to a signed, packaged, auto-updating client that survives an endpoint security review is estimated here at twenty to forty engineer-days, or 16,000 to 32,000 US dollars at the assumed rate. This is a one-off, it does not recur, and it is the number a finance function should be shown alongside the recurring figure rather than instead of it. A comparison that amortises it over a single year makes the platform look better than it is; a comparison that omits it makes the client look better than it is.

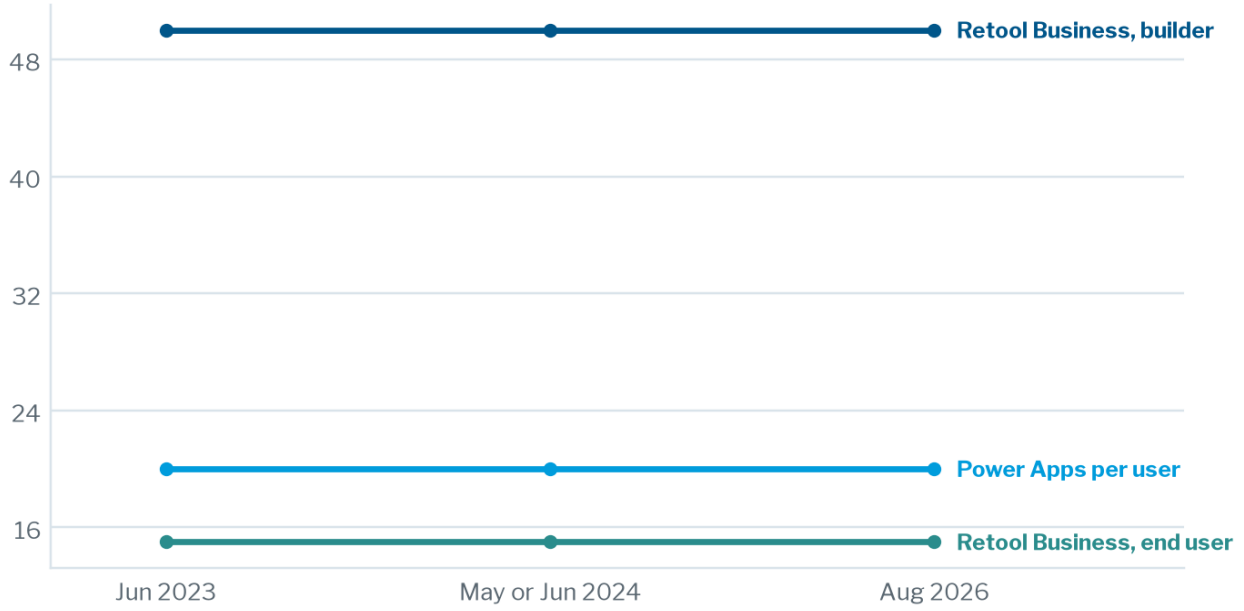
Did low-code pricing move

This was the most checkable question in the brief and the answer is unusually clean. Archived copies of the same vendor pricing pages were compared against the current ones. The question was narrow: did the pricing dimension change, did the rate change, or did neither.

Vendor and line	Jun 2023	May or Jun 2024	Aug 2026	What changed
Retool Team, per builder, annual	$10	$10	$10	Rate unchanged across all three snapshots

Vendor and line	Jun 2023	May or Jun 2024	Aug 2026	What changed
Retool Team, per end user, annual	$5	$5	$5	Rate unchanged
Retool Business, per builder, annual	$50	$50	$50	Rate unchanged
Retool Business, per end user, annual	$15	$15	$15	Rate unchanged
Retool workflow allowance	1 GB throughput per month	5,000 runs per month	5,000 runs per month	Dimension changed between 2023 and 2024, from data throughput to run count. Rate unchanged
Retool AI	Not present	Not present	250 to 3,000 credits by tier, plus hourly agent billing	New meter added on top of the unchanged seat
Power Apps per user, per month	$20	$20	$20	Rate unchanged across all three snapshots
Power Apps at 2,000 or more seats	40 percent discount	$12	$12	Same effective rate, expressed differently
Power Apps per app, per user per app	$5	Absent	Absent	Cheapest option withdrawn, not repriced
Dataverse database capacity	Not on this page	$40 per GB per month	$40 per GB per month	Rate unchanged
Copilot Studio pack	Not present	$200 per 25,000 messages	$200 per 25,000 credits	Unit renamed, rate explicitly unchanged

FIGURE 5 **Published seat rates, per user per month on annual billing**



FACT US dollars. 2023 and 2024 figures read from Internet Archive snapshots of the vendors' own pricing pages, current figures read from the live pages on 25 August 2026. Three flat lines is the entire result. List rates only: enterprise agreements discount off these and the discount is not observable from outside. [22] [23] [24] [25] [26] [27]

ASSESSMENT The claim under test was that these platforms became execution platforms for AI-generated work without their rate cards changing. The archives support it. Neither Retool nor Microsoft moved a seat price in three years across a period in which the cost of producing the applications those seats host fell substantially. Where a dimension did move, it moved in ways that did not reduce the bill: Retool replaced an uncapped run count constrained by data throughput with a hard run cap, and Microsoft withdrew its cheapest per-app option. [22] [23] [24] [25] [26] [27]

FACT Bubble is the genuine counterexample and it should be reported as one. Bubble replaced server capacity with workload units as its billing dimension, effective 1 May 2023 for new applications with all applications migrated by 1 October 2024. That is a real move from a capacity dimension to a usage dimension. Bubble also disclosed in the same announcement that customers remaining on legacy plans faced roughly a 10 percent increase and that agency plans saw a comparable per-seat increase. [30]

ASSESSMENT Bubble does not overturn the position, for two reasons that should be stated rather than assumed. It moved from capacity to usage, not from seats to usage, because Bubble never charged per seat for application users in the way Retool and Power Apps do. And Bubble is not the platform an enterprise is renting for internal line-of-business applications, which is the comparison this report is about. The overturning condition named at the outset was a major low-code vendor repricing away from seats toward usage. It has not happened. [30]

ASSESSMENT For OutSystems, Mendix, Appian and ServiceNow, pricing is quote-only. Third-party figures for these vendors circulate widely and are not used here. A reader evaluating those

four has information this author does not, namely their own quote, and should run the crossover calculation in this report against it directly.

Paying rent twice

FACT Retool meters AI generation on top of seats. Current plans allocate AI credits by tier, 250 per month on Free, 1,000 on Team, 3,000 on Business, and on Enterprise 1,000 per builder plus a 4,000 pooled allowance. Credits are pooled at account level, renew monthly and do not roll over. Additional credit packs are sold to paid plans. Agent time is billed by the hour at rates that vary by model, with the free tier carrying up to 20 hours per month. [22]

FACT Microsoft is moving AI from bundled to metered on a published schedule. AI Builder capacity add-ons reached end of sale on 1 November 2025. On 1 November 2026 both the add-ons and the AI Builder credits seeded into premium licences such as Power Apps Premium reach end of life, and seeded credits are removed from those licences for all new and existing customers, including those on Enterprise Agreements. There is no automatic conversion. Customers are directed to purchase Copilot Credits instead, at one US cent per credit on the pay-as-you-go meter, with unused credits not carrying over month to month and technical enforcement, including service denial, applied on overage. [28] [29]

ASSESSMENT This is the finding that strengthens the original position rather than qualifying it. The claim being tested was that organisations pay twice, once for the assistant that writes the application and again at a multiple for the seats that host it. That is now understating it. From November 2026 a Power Apps Premium customer pays the unchanged 20 US dollars per seat, loses the AI allowance that seat previously carried, and buys AI capacity as a third line. The seat price did not fall to reflect the removal. Paying rent twice has become the published architecture of the product rather than an inference drawn from it. [28] [29]

Where the backend lives

The shell does not remove the backend. It relocates the bill, and the relocation is worth pricing at published rates for one representative internal application: 200 users, roughly 40 requests per user per working day, about 176,000 requests per month, with 50 milliseconds of compute per request.

Option	Annual at this workload	What the published rate includes	Note
Cloudflare Workers, Paid plan	$60	$5 per month, 10 million requests and 30 million CPU milliseconds included	The workload sits entirely inside the included allowance. The monthly floor is the whole bill

Option	Annual at this workload	What the published rate includes	Note
AWS Lambda with function URLs	Under $5	$0.20 per million requests, $0.0000166667 per GB-second, 1 million requests and 400,000 GB-seconds free monthly	Effectively free at this volume. Adding API Gateway rather than function URLs would add about $2 a year
Power Apps included capacity	$0 marginal	250 MB database and 2 GB file per Premium licence, so 50 GB and 400 GB across 200 seats	The capacity is free only in the sense that $48,000 of seats has already been bought. Additional Dataverse is $480 per GB per year
Retool included capacity	$0 marginal	5 GB database and 5 GB file storage, 5,000 workflow runs per month on Team	Same structure. The run cap is the constraint that bites first on scheduled work

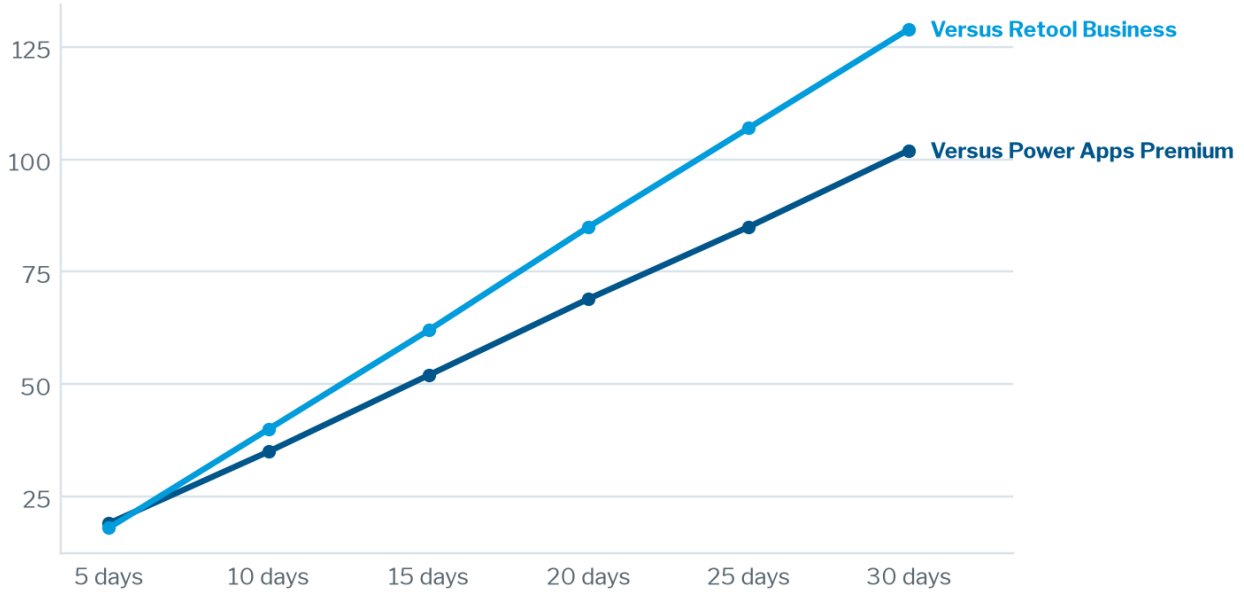
ASSESSMENT At this scale the compute bill is a rounding error on both serverless platforms and the entire question is the seat rent sitting on top of the included capacity. That changes with data movement rather than with compute, and the egress argument is not re-derived here; see the author's prior work on cloud pricing. [32] [33] [44]

ASSESSMENT The line that should worry a buyer is Dataverse at 40 US dollars per gigabyte per month, which is 480 US dollars per gigabyte per year. That is roughly 600 times the annual cost of a gigabyte in Cloudflare R2 or object storage generally. For a document-heavy internal application this single line can exceed the seat cost, and it does not appear in any build-versus-rent comparison this author has seen. [25] [32]

The crossover

The comparison reduces to one arithmetic question. A desktop client costs a fixed amount per year regardless of how many people use it. A per-seat platform costs a variable amount that scales with headcount. Somewhere they cross.

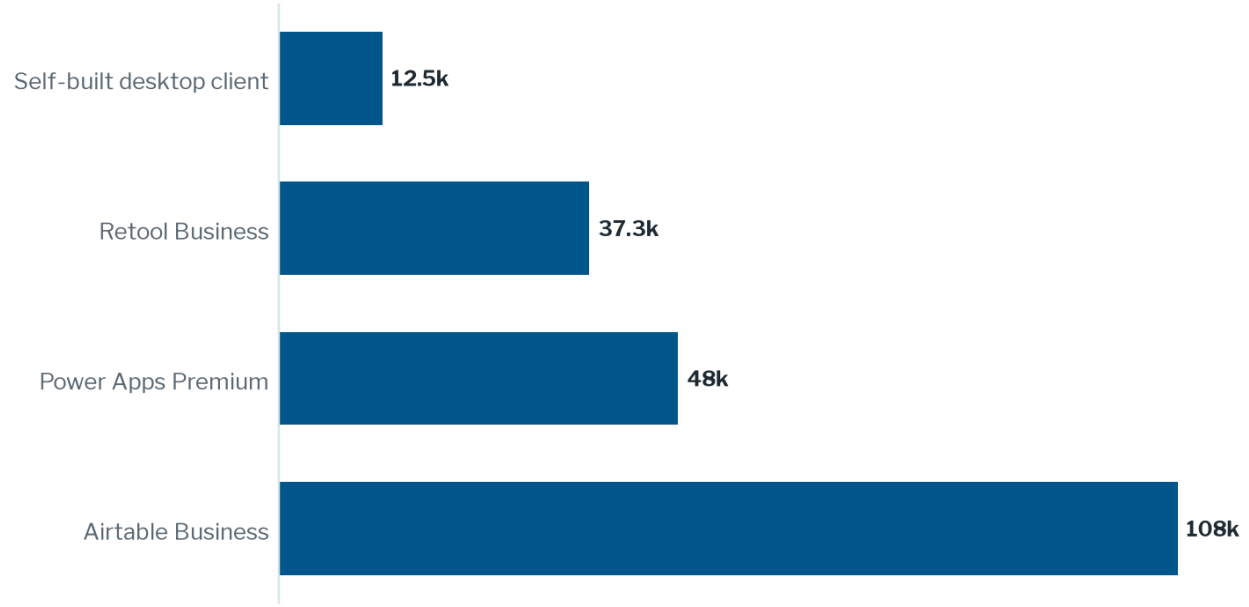
FIGURE 6 **Seats at which the desktop client becomes cheaper than the licences**



ASSESSMENT Horizontal axis is the author's assumed annual maintenance labour in engineer-days at $800 per day, plus $492 of published fixed costs. Vertical is the seat count above which building is cheaper. Power Apps at $240 per seat per year. Retool Business at three builders at $600 and end users at $180. List rates only, steady state only, year-one build cost excluded. The width of this band is the honest answer. [22] [25]

ASSESSMENT At the base case of fifteen engineer-days a year, the client becomes cheaper than Power Apps Premium at 52 seats and cheaper than Retool Business at 62. At an optimistic six days it crosses under 20 seats against both. At a pessimistic thirty days it crosses above 100. The band is wide because the deciding variable is unmeasured, and any report that gives a single number here is hiding that. [22] [25]

FIGURE 7 **Annual cost at 200 seats**



ASSESSMENT Thousands of US dollars per year, steady state, list rates. The client figure is the base-case estimate from the costing above and is the only bar that does not scale with headcount. Retool assumes three builders and 197 end users on annual billing. Airtable is included as a reference point for how far the per-seat model stretches, not as a like-for-like substitute. [22] [25] [31]

ASSESSMENT The gap widens with scale in a way that is easy to miss. At 2,000 seats Power Apps Premium at the volume rate of 12 US dollars per user per month is 288,000 US dollars a year, against a client cost that has not moved from roughly 12,500. The self-built client does not become 23 times better at 2,000 seats than at 200; it becomes 23 times cheaper relative to the alternative, because one side of the comparison is a fixed cost and the other is not. Conversely, below about twenty seats the platform is simply the right answer and no amount of architectural preference changes that. [25]

Distribution is the gate, not the framework

This is where the thesis is most likely to break, so it gets the most careful treatment. The conclusion is that the obstacle is real, that it is organisational rather than technical, and that Microsoft's own documentation names the way through it.

FACT SmartScreen evaluates two signals, publisher reputation from the signing certificate and file hash reputation from download history. Microsoft states that even a signed, newly created binary can show a warning until its hash or publisher certificate accumulates sufficient evidence of positive reputation, that there is no exact threshold, and that it can take several weeks and hundreds of clean installs from a wide audience. [17]

FACT Extended Validation certificates no longer help with this. Microsoft's documentation states directly that EV certificates no longer bypass SmartScreen, that this behaviour existed years ago and no longer exists, and that paying a premium for EV solely to avoid SmartScreen

warnings is no longer justified. It also notes that on Windows 11, Smart App Control may supersede SmartScreen application reputation and will block execution of unsigned files unless the file has a positive reputation, applying to all executables rather than only downloaded ones. [17]

ASSESSMENT Taken alone, those two facts look fatal for an internal application. A client with sixty users will never generate hundreds of clean installs from a wide audience, so on the consumer reputation path it warns forever, and the expensive certificate that buyers assume solves this does not.

FACT The same Microsoft page names three enterprise routes that bypass the problem entirely. Enterprise environments may distribute files from trusted intranet locations not subject to SmartScreen review. Enterprise IT administrators may submit files for review through the Microsoft Security Intelligence portal, which Microsoft states can accelerate trust for internal or managed deployments. And enterprise policy can itself change SmartScreen behaviour, including removing the ability to bypass a warning. [17]

FACT Application control works the same way. Windows Defender Application Control enforces allowlisting in the kernel before code is loaded, and supports managed installers: when a deployment tool such as Microsoft Endpoint Configuration Manager or Intune is configured as a managed installer, items it deploys are added to the allow list automatically. Microsoft recommends WDAC over AppLocker, noting that AppLocker enforcement runs in user mode and can be disabled by an attacker with local administrator rights. [18]

FACT The deployment tool required for that path is usually already paid for. Microsoft Intune Plan 1 lists at 8 US dollars per user per month, and is included in Microsoft 365 Business Premium, E3, E5, F1 and F3 and in Enterprise Mobility and Security E3 and E5, so for most organisations running managed Windows endpoints the marginal cost of distributing one more application through it is zero. [41]

ASSESSMENT So the answer to who approves installation of a newly signed binary on managed endpoints is: the same team that already approves every other binary, using the same mechanism, at a one-time cost. It is a ticket, a signing identity registered once, and an Intune deployment ring. What it is not is a per-seat charge, and this is the crux. The seat rent was purchasing an implicit exemption from that internal conversation, because a low-code application is a URL and a URL needs no allowlisting entry. The exemption is real and it has been priced at hundreds of dollars per user per year without anyone naming it as the thing being bought.

ASSESSMENT On endpoint detection specifically, the honest position is that the evidence available is anecdotal. Practitioner reports of endpoint detection tools flagging unfamiliar signed executables recur across vendors and are consistent enough to establish that the complaint is real, but no vendor publishes false-positive rates by application category and no independent test measures them. Treating an unfamiliar Electron or Tauri executable as likely to generate at least one detection event during a first rollout is the prudent planning assumption. Quantifying that rate is not possible from public sources and is named in what this document cannot answer.

ASSESSMENT The practical consequence for anyone acting on this report is a sequencing rule. Get the signing identity, the Intune deployment ring and the endpoint security exception agreed

before writing the application, not after. The framework decision can wait; the distribution decision cannot, because it is the one with a queue in front of it and the one that can return a no.

The end-user argument, and what the evidence supports

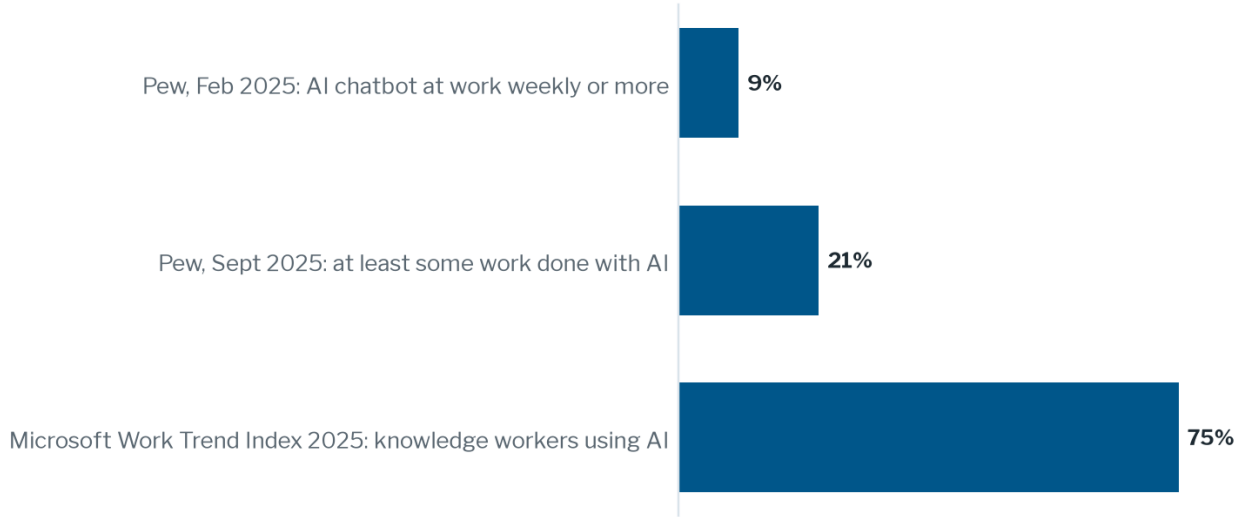
The proposition is that most employees do not want to learn prompting and should not have to, and that hiding AI behind an ordinary interface is the productivity path for organisations that are not AI-native. The evidence bears on the first half and not the second.

FACT Pew Research Center, surveying 5,010 employed US adults between 2 and 8 September 2025 as part of a nationally representative sample of 8,750 adults on its American Trends Panel, found that 21 percent of US workers say at least some of their work is done with AI, up from 16 percent a year earlier. Two percent said all or most of their work is, unchanged year on year. Sixty-five percent said they do not use AI much or at all. [35]

FACT An earlier Pew survey of 5,273 employed US adults fielded in February 2025 found 9 percent using AI chatbots at work daily or a few times a week, 7 percent a few times a month, 55 percent rarely or never, and 29 percent who had not heard of them. Among non-users, 36 percent cited lack of relevance to their job, 22 percent lack of interest, 10 percent not knowing how, and 9 percent employer restrictions. [36]

VENDOR CLAIM Microsoft's 2025 Work Trend Index, a survey Microsoft commissioned and published covering 31,000 workers across 31 countries and combined with LinkedIn labour signals and Microsoft 365 telemetry, reported that 75 percent of knowledge workers use AI. Microsoft sells the products this figure describes. The population is knowledge workers rather than all workers, which accounts for part of the gap and not for all of it. [37]

FIGURE 8 **Two measurements of worker AI use, four years into the category**



FACT The three questions are not identical and the populations differ, which explains some of the spread. It does not explain a factor of eight between the first and third. The first two come from a probability-based panel run by a research

organisation that sells nothing in this category. The third comes from a vendor measuring demand for its own products. Both figures are reported accurately; the gap between them is the finding. [35] [36] [37]

ASSESSMENT The first half of the proposition survives contact with this evidence. A workforce in which 55 percent rarely or never use an AI chatbot and 29 percent have not heard of one is not a workforce that is going to be trained into prompt literacy on any useful timescale. Building interfaces that require it is building for a minority and calling it a rollout.

HYPOTHESIS The second half, that hiding AI behind an ordinary interface produces measurable productivity gains in organisations that are not AI-native, is reasoning rather than measurement and is offered as a hypothesis. No study located for this report compares outcomes for the same AI capability delivered through a chat interface against the same capability delivered through a conventional form-and-list application. What would confirm it: a controlled comparison inside one organisation, same task, same model, two interfaces, measuring completion rate and time on task. What would refute it: evidence that the gains come from the model rather than the interface, so that the delivery mechanism is neutral and this whole argument is decoration on a decision made elsewhere.

ASSESSMENT Nothing in this report rests on the age of the user. Pew reports higher chatbot use among workers aged 18 to 29 and among those with postgraduate degrees, and reports that workers aged 50 and over are more likely than younger cohorts to cite not knowing how as their reason for not using them. That is the extent of what a named source supports and this report goes no further, because the generational claim in this category is almost always asserted without one. [36]

What the shell adds to the attack surface

ASSESSMENT A generated Electron or Tauri project is a repository full of configuration that executes on folder-open: build scripts, task definitions, editor settings, dependency manifests, and in Tauri's case a capability file that determines what the frontend can reach. This enlarges the surface described in the author's prior work on agent speed and the detection gap rather than creating a new one. The mechanism there was that automated change outpaces the controls designed for human-rate change. A desktop client project adds a second copy of that problem on the endpoint side, because the artefact of the automation is now a signed executable running with user privileges rather than a row in a hosted platform. [45]

ASSESSMENT The specific risk is not that AI writes insecure Electron code, though it may. It is that the security-relevant settings in an Electron or Tauri project are configuration rather than code, they are set once in the first hour of a generated scaffold, and nobody reads them again. Whether nodeIntegration is on, whether the sandbox is disabled to make something work, whether a Tauri capability was widened to a glob during debugging: all of these are one-line changes that pass review because they are not in the part of the diff anyone reads. [3] [11]

ASSESSMENT This argues for a specific control rather than a general warning. The webview configuration and the capability file should be treated as security-controlled artefacts with

named owners and change review, in the same way a firewall rule is, and separately from application code review. That control costs nothing and is not in any generated project.

ASSESSMENT Against that, the shell removes a surface too, and the comparison should be two-sided. A low-code platform is a hosted multi-tenant runtime holding the organisation's data, reachable from the internet, with a shared identity plane. A desktop client over a backend the organisation already runs adds no new tenancy and no new internet-reachable data store. Which surface is larger depends on the organisation, and asserting that the self-built option is obviously riskier is as unexamined as asserting the reverse.

The alternatives, priced

Option	Cost to ship	Where it wins	Where it fails
A browser tab	$0	Any application that is read, filter, enter, submit. No install, no signing, no allowlisting, no update mechanism to build	No offline, no local filesystem beyond downloads, no protocol handling, no tray or background work, no presence in the taskbar
Installable web application	$0 beyond hosting	Adds a dock or taskbar entry, a window without browser chrome and an icon. Supported in Chrome and Edge on Windows and macOS, and in Safari on macOS 14 and later via Add to Dock	Still a web page. Storage remains subject to browser eviction policy, and on WebKit that means the seven-day script-writeable storage rule applies
Tauri	As costed above	Small binaries, a real Rust privilege boundary, published independent audits, and mobile targets that exist	Two rendering engines to test, Linux graphics failures the project documents itself, and a mobile plugin set with named gaps including the updater
Electron	As costed above, plus a heavier upgrade treadmill	One Chromium everywhere, which deletes the divergence problem. Named production applications a risk committee will recognise	Larger footprint, at least two forced major upgrades a year, and a security history in which context isolation has been bypassed repeatedly, most recently in July 2026
Wails	Similar to Tauri	A natural fit for a team that already writes Go and does not want Rust in the stack	Version 3 is beta as of mid-2026. Choosing pre-release infrastructure for a system somebody has to own for five years needs a reason beyond language preference
.NET MAUI Blazor Hybrid	Similar, plus Visual Studio tooling	A Microsoft-supported path for a Microsoft-shop team, sharing Razor components	Ties the client to the .NET release cadence and the Microsoft toolchain, which is

Option	Cost to ship	Where it wins	Where it fails
		with an existing web application	either the point or the objection depending on the organisation
Deno on the desktop	Not assessable	Attractive if the backend is already Deno	No stable, documented desktop distribution story comparable to the above. Not recommended for a system with a five-year owner

ASSESSMENT Two of the alternatives are ruled out on maturity rather than on merit, and the reason should be stated rather than implied. Wails version 3 remains pre-release as of mid-2026, published as beta with alpha release tags, which is a poor foundation for a system somebody has to own for five years even though production applications ship on it today. Deno has no stable, documented desktop distribution story comparable to Electron or Tauri. Neither is a bad technology; both are bad choices for a first internal deployment that has to survive a handover. [42]

ASSESSMENT For a meaningful share of internal applications the answer is a browser tab, and saying so is more useful than the rest of this report. If the application is read, filter, enter, submit against a backend the user is always online to reach, an installed client buys a taskbar icon and a signing bill. The installed client earns its cost when at least one of four things is true: the application must work without the network, it must reach the local filesystem or a device, it must run or notify in the background, or it must be present as an application rather than as a tab the user closes by accident. If none of those is true, ship a web page and spend the fifteen engineer-days elsewhere. [43]

The counter-arguments

Installability buys less than people think

ASSESSMENT Conceded, and it is the strongest of the objections. Most of what teams want from an installed client is available from an installable web application at zero marginal cost on Chrome, Edge and, since macOS 14, Safari. The answer is not that installability is valuable in general; it is that it is valuable for a specific and enumerable set of requirements listed above. A team that cannot name which of those four applies to its application should not be building a client. [43]

IT will not let a business unit ship binaries

ASSESSMENT Conceded, and correctly so. A business unit should not be able to ship binaries to managed endpoints, and any organisation where it can has a bigger problem than its low-code bill. The answer is that this is an argument about who ships, not about whether the artefact is viable. IT can ship it through the same managed installer path it uses for everything else, at a one-time cost. What the objection correctly identifies is that this pattern requires an IT partnership that the low-code model was specifically sold as a way to avoid, which means the political cost of the change lands on whoever proposes it. [18]

Generated code is still code, and somebody owns it

ASSESSMENT Conceded, and this is the objection the costing in this report exists to answer rather than dodge. Somebody signs it, patches it, upgrades the runtime twice a year and eventually leaves. That is exactly what the fifteen engineer-days represent, and it is the largest line in the estimate by an order of magnitude. The seat rent was absorbing that invisibly, which is a genuine service. The question this report answers is what that service costs per user, and the answer at list rates is 240 US dollars a year at Power Apps and 180 at Retool for people who never build anything. Whether that is good value for absorbed maintenance is now a decision a reader can make with a number in front of them.

Seat rent buys governance the shell does not replace

ASSESSMENT Conceded, and this deserves to be itemised rather than waved at, because it is the part of the argument that survives everything else in this report. A per-seat platform supplies a permission model administered outside the application, an audit trail of who changed what and when, a hosted runtime with an availability commitment, a vendor with a support contract and a security questionnaire already on file, environment separation and release management, and a data loss prevention integration that the organisation's compliance function already accepts. A self-built client replaces none of these. It replaces the interface.

ASSESSMENT The honest version of the build case is therefore narrower than the one usually made. It is not that the shell replaces the platform. It is that for an application where the governance requirement is modest, an interface is expensive to rent and cheap to build, and the crossover is a seat count. For an application handling regulated data, or one where an auditor will ask who approved a change in March, the seat rent is buying something real and this report's arithmetic does not apply.

Webview divergence is a QA cost on a team with no QA function

ASSESSMENT Conceded and largely unanswerable. A team small enough to build its own client with an AI assistant is a team without a test matrix, and Tauri's two-engine reality demands one. The mitigation is the architectural rule stated earlier, adopted at the start: keep everything that touches the machine in Rust, and the divergence surface shrinks to rendering and layout, which is testable by looking. The residual risk is Linux, which is why the recommendation for a team with no QA function is to pick Electron if Linux is in scope and Tauri if it is not. [10] [12]

The shell does not remove the backend, it relocates the bill

ASSESSMENT Conceded, and the relocation is priced in this report at published rates. At the reference workload the backend is 60 US dollars a year on Cloudflare Workers and effectively free on Lambda, against 48,000 US dollars of Power Apps seats sitting on top of the included capacity. The relocation is favourable at this scale. It stops being favourable when data volume rather than request volume drives the bill, which is a different question and one the author has addressed elsewhere. [32] [33] [44]

Scenarios to 2028

Probabilities below are subjective, are the least reliable content in this document, and should be read as relative weights rather than measurements. The tripwires are the useful part, because any reader can watch them without private information.

SCENARIO A The rate card holds and the meter grows 55 percent

Seat prices stay where they have been since 2023 while AI consumption becomes a separate and growing line on every low-code invoice, following the AI Builder to Copilot Credits pattern already scheduled for November 2026.

Consequence. The build case strengthens every year without any vendor doing anything wrong, because the denominator in the comparison keeps rising while the numerator does not.

Earliest visible sign. A second major vendor announcing that AI capacity previously bundled with a seat will be sold separately, with no corresponding reduction in the seat price.

SCENARIO B A major vendor reprices toward consumption 25 percent

Microsoft, Retool or a comparable enterprise vendor replaces or substantially discounts per-seat licensing for internal applications with a usage meter, in the way Bubble moved from capacity to workload in 2023.

Consequence. The crossover seat count rises sharply for low-usage applications and the build case narrows to high-headcount, low-activity deployments, which is most internal software.

Earliest visible sign. A published price list that charges nothing for a user who opens an application twice a month. Watch the read-only or end user tier first, because that is where the seat model is least defensible.

SCENARIO C Endpoint policy hardens faster than the tooling 20 percent

Smart App Control and kernel-level application control become default-on rather than opt-in across managed Windows estates, and reputation requirements tighten for publishers without volume, making self-published binaries genuinely undeliverable outside a formal managed installer path.

Consequence. The distribution gate moves from a one-time ticket to a standing programme, adding real recurring cost to the client side of the comparison and pushing the crossover well past 100 seats.

Earliest visible sign. Microsoft changing the default state of Smart App Control on new Windows installations, or publishing a reputation threshold that cannot be met by an application with a fixed internal audience.

Leading indicators

If this happens	It means	Moves toward	Where to watch
Power Apps Premium list price changes for the first time since 2023	The rate card is no longer frozen and the central premise of this report needs rechecking	Scenario B	The Power Apps pricing page, compared against the archived 2023 and 2024 snapshots cited here
A vendor introduces a genuinely free read-only or viewer tier for internal applications	The seat model is being defended at its weakest point rather than extended	Scenario B	The end user or internal user row on Retool's plan comparison, and Microsoft's licensing guide changelog
November 2026 passes and seeded AI Builder credits are removed as scheduled	The bundled-to-metered transition is executing on the published timetable	Scenario A	Microsoft's End of AI Builder credits page and the Power Platform admin centre capacity view
Stack Overflow or JetBrains adds a desktop framework category to its survey	Adoption in this category becomes measurable for the first time	Neither	The technology sections of the annual survey results
Tauri ships a supported mobile updater	The mobile story stops being a desktop story with mobile targets attached	Neither	The plugins-workspace README support table
Smart App Control ships default-on for new Windows installations	The distribution gate hardens materially for publishers without download volume	Scenario C	Windows release notes and the SmartScreen reputation documentation
Electron changes its supported-versions policy from three majors	The upgrade treadmill, which is the largest cost line in this report, changes length	Neither	The Electron release schedule and timelines documentation

Implications

For CIOs and CTOs approving the spend

ASSESSMENT The question to put to your own organisation is not which framework. It is how many people are paying an internal application seat rent for software they open twice a week. Pull the seat counts by application from the platform's own admin console, apply 240 US dollars per user per year at Power Apps list or 180 at Retool for non-builders, and compare each application against roughly 12,500 a year. Everything above about 60 seats is a candidate. Everything below about 20 is not, and should be left alone. The November 2026 removal of seeded AI credits is a natural forcing date for that review because it changes the bill without any decision on your part. [25] [28]

For enterprise architects

ASSESSMENT The decision that matters is made in the first week of the first project and it is not the framework. Write down the rule that the webview is a rendering surface and everything

touching the machine goes through the native layer, and enforce it in review. That single constraint converts the entire webview divergence problem from an ongoing test burden into a design rule, and it is the difference between a Tauri estate that ports cleanly and one that has to be rewritten when somebody asks for Linux. Then pick the framework on two questions: does anything need a capability WebKit lacks, and is Linux in scope. Two noes means Tauri. Either yes means Electron. [10] [12]

For low-code Center of Excellence leads

ASSESSMENT This is the most exposed role in the report and the recommendation is not to defend the platform. It is to become the function that runs the crossover calculation, because somebody in the organisation is going to and it is better that it is the person who understands the governance the platform supplies. The defensible position is a portfolio one: the platform keeps the applications where the audit trail, the permission model and the vendor accountability are worth the seat, and the small number of high-headcount low-governance applications move off it. A Center of Excellence that argues everything belongs on the platform will lose that argument on arithmetic within two budget cycles.

For finance and procurement

ASSESSMENT Three specific things to check. First, whether your enterprise agreement discount off list is large enough to move the crossover, because at a 50 percent discount every seat count in this report doubles and the conclusion may reverse. Second, the Dataverse line at 40 US dollars per gigabyte per month, which is 480 a year per gigabyte and is the single most mispriced item in the category. Third, whether your renewal already assumes AI capacity that ceases to be included on 1 November 2026, because that is a price increase arriving without a price change and it will not be flagged. [25] [28]

For IT endpoint and desktop security engineering

ASSESSMENT You are the gate and you will be asked. The useful preparation is to define, before the first request arrives, what an internal desktop client has to satisfy: a single organisational signing identity rather than one per team, deployment only through the managed installer path so that application control allowlisting is automatic, a named owner for the webview configuration and capability file as security-controlled artefacts, and an endpoint detection exception process that does not require a new ticket per release. Every one of those is cheaper to define once than to negotiate per project, and defining it converts your team from an obstacle into the reason the pattern is viable at all. [17] [18]

For solo builders and small agencies

ASSESSMENT The crossover arithmetic runs against you and you should know it before you sell the work. Below about twenty seats, a client's low-code bill is genuinely cheaper than a client you build and maintain, and pretending otherwise is a bill they will discover in year two. Where the pattern is strong for you is the opposite end: clients with hundreds of light users, and clients whose data cannot go into a multi-tenant hosted platform. Price the maintenance explicitly as a

retainer rather than folding it into the build, because the maintenance is the product and the build is the sample.

For operations leaders whose staff will use the result

ASSESSMENT The evidence supports your instinct that your staff will not learn to prompt. A probability-based survey found 55 percent of US workers rarely or never using an AI chatbot at work and 29 percent who had not heard of one, and the year-on-year movement in serious use has been slow. Insist on an interface your team recognises rather than a chat box, and insist on it early, because it is a two-week decision at the start of a project and a rewrite at the end. What the evidence does not yet support is a productivity number for making that choice, and you should not let anyone quote you one. [35] [36]

What this document cannot answer

Five questions matter to the conclusion and cannot be settled from public sources. Each would need primary research.

What annual maintenance actually costs. Every crossover figure in this report turns on an estimate of fifteen engineer-days a year, and nobody publishes the real number. A survey of twenty organisations running self-built desktop clients, recording engineer-days by category over two years, would replace the widest band in this document with a measurement and is the single highest-value study in this area.

How often endpoint detection actually flags a newly signed internal client. Practitioner reports establish that it happens. No vendor publishes false-positive rates by application category and no independent test measures them, so the planning assumption in this report is a judgment rather than a rate.

What enterprise low-code buyers actually pay. Every low-code figure here is list price. Real transaction prices for Power Platform, Retool, OutSystems, Mendix, Appian and ServiceNow are under non-disclosure. A buyer-side survey of realised per-seat cost would change the crossover for everyone reading.

Whether interface delivery changes outcomes independently of the model. The end-user argument in this report is reasoning, explicitly labelled as such. A controlled comparison of the same AI capability delivered through chat and through a conventional application would settle it in either direction.

How many organisations have actually done this. There is no count of internal applications delivered as self-built desktop clients over an organisation's own backend, because the category has no name and therefore no way to be counted. That is the terminology finding turning into a measurement problem.

Half-life of this assessment

Decaying within six months. Every version number, release date and end-of-life date for Electron and Tauri, given an eight-week major cadence. Retool's AI credit allowances and its limited-time credit offer. Bubble's plan prices. The specific Electron advisories cited, which will be superseded by the next batch. The state of Wails version 3.

Decaying within twelve months. The pricing comparison itself, which should be re-run against fresh archive snapshots in mid-2027 and which is the claim most worth rechecking. The November 2026 AI Builder transition, which will have either executed as documented or not. The Tauri mobile plugin gaps, which are the fastest-moving part of that project. The SmartScreen and Smart App Control default behaviours.

Decaying within twenty-four months. The crossover arithmetic, which holds as long as seat rates and labour costs move together, and which needs redoing if either moves alone. The adoption download ratios. The end-user survey figures, where Pew's year-on-year movement suggests the numbers shift slowly but do shift.

Architectural, and unlikely to decay. That Tauri binds to the operating system webview and Electron bundles Chromium, which is a design commitment rather than a version fact. That this produces two engines rather than three, with Linux behaving like macOS. That a per-seat cost meets a fixed cost at some seat count, so the decision is arithmetic rather than preference. That the distribution gate is organisational rather than technical. And that the category has no settled noun, which will remain true until somebody with distribution supplies one.

Limitations

The author built and led a Low-Code/No-Code Center of Excellence at Concentrix and holds Microsoft Power Platform credentials, so has commercial history in the category this report argues against. The author also sells AI platform consulting and publishes paid automation training, which is an interest pointing the other way. Both are stated because the reader should weigh the argument knowing that the author has been paid by both sides of it. No vendor named in this document commissioned, funded, reviewed or saw it in advance, and no vendor was contacted during its preparation.

This report reads documentation, pricing pages, package registries, security advisories and published research. It does not benchmark either framework, measure any deployment outcome, install or test either toolchain, or interview any buyer. Every performance, footprint and reliability characteristic discussed here is taken from a project's own documentation or from the recurrence of a complaint, never from measurement performed for this report. A hands-on evaluation would produce different and complementary findings, and would be the natural next piece of work.

The evidence base has four specific weaknesses. All low-code figures are list prices and enterprise buyers pay less by an unknown margin. All labour figures are the author's estimates at an assumed rate and drive the central conclusion. The archived pricing comparison covers Retool

and Power Apps cleanly; Airtable's and Bubble's archived pages render prices through scripts that the Internet Archive did not capture, so no historical comparison is offered for either beyond Bubble's own published announcement. And adoption is measured only through package registry downloads, which count automated pipelines as readily as people.

One structural caution about the argument itself. This report finds that the evidence supports the position it set out to test, which is the outcome most likely to reflect the framing of the question rather than the state of the world. The specific check applied was to look for the three named overturning conditions rather than for confirmation. One of the three, a vendor repricing away from seats, was found in a smaller vendor and reported. The other two were not found. A reader who wants to stress this document should start with the maintenance estimate, because it is the load-bearing number and it is the one the author produced rather than located.

Sources

[1] Electron. "Releases: Stable." Retrieved 25 August 2026. *Vendor primary* releases.electronjs.org

[2] Electron. "Release Schedule." Retrieved 25 August 2026. *Vendor primary* releases.electronjs.org

[3] Electron. "Security." Documentation, latest. Retrieved 25 August 2026. *Vendor primary* electronjs.org

[4] Electron. "Showcase." Retrieved 25 August 2026. *Vendor primary* electronjs.org

[5] Electron. Security Advisories, GitHub. Batch published 27 July 2026, including CVE-2026-70601 and CVE-2026-70610. Retrieved 25 August 2026. *Vendor primary* github.com

[6] crates.io. "tauri" crate metadata. Version 2.11.5, published 1 July 2026. Retrieved 25 August 2026. *Vendor primary* crates.io

[7] npm registry download counts API, period 26 July to 24 August 2026. Retrieved 25 August 2026. *Vendor primary* api.npmjs.org

[8] Tauri. "Architecture." Version 2 documentation. Retrieved 25 August 2026. *Vendor primary* v2.tauri.app

[9] Tauri. "Webview Versions." Version 2 reference. Retrieved 25 August 2026. *Vendor primary* v2.tauri.app

[10] Tauri. "Linux Graphics Issues." Version 2 documentation. Retrieved 25 August 2026. *Vendor primary* v2.tauri.app

[11] Tauri. "Security." Version 2 documentation. Retrieved 25 August 2026. *Vendor primary* v2.tauri.app

[12] Tauri. plugins-workspace repository README, platform support table for 30 official plugins. Branch v2, retrieved 25 August 2026. *Vendor primary* github.com

[13] Tauri. "Updater" plugin documentation, supported platforms. Retrieved 25 August 2026. *Vendor primary* v2.tauri.app

[14] Radically Open Security. "Penetration Test Report Tauri 2.0." Amsterdam, 7 August 2024. Funded by NLnet via Next Generation Internet. *Independent security audit* github.com

[15] Radically Open Security. "Retest Report: The Tauri Programme." Amsterdam, 3 February 2022. Funded by the NGI Assure grant fund. *Independent security audit* github.com

[16] Microsoft. "Distribute your app and the WebView2 Runtime." Microsoft Edge developer documentation, updated 15 July 2026. *Vendor primary* learn.microsoft.com

[17] Microsoft. "SmartScreen reputation for Windows app developers." Windows apps documentation, updated 17 August 2026. *Vendor primary* learn.microsoft.com

[18] Microsoft. "Windows Defender Application Control." Windows IoT Enterprise documentation. Retrieved 25 August 2026. *Vendor primary* learn.microsoft.com

[19] CA/Browser Forum. "Baseline Requirements for the Issuance and Management of Publicly-Trusted Code Signing Certificates." Hardware crypto module requirement effective 1 June 2023, ballot CSC-17. *Standards body primary* cabforum.org

[20] Apple. "Compare Memberships." Apple Developer Support. Retrieved 25 August 2026. *Vendor primary* developer.apple.com

[21] Microsoft. "Artifact Signing, formerly Trusted Signing: Pricing." Tier rates render dynamically and were not present in the page source or in Internet Archive snapshots of 9 November 2024 and 16 February 2025. Retrieved 25 August 2026. *Vendor primary* azure.microsoft.com

[22] Retool. "Pricing." Retrieved 25 August 2026. *Vendor primary* retool.com

[23] Retool. "Pricing." Internet Archive snapshot, 9 June 2023. *Vendor primary via archive* web.archive.org

[24] Retool. "Pricing." Internet Archive snapshot, 23 May 2024. *Vendor primary via archive* web.archive.org

[25] Microsoft. "Power Apps pricing." Retrieved 25 August 2026. *Vendor primary* microsoft.com

[26] Microsoft. "Power Apps pricing." Internet Archive snapshot, 2 June 2023. *Vendor primary via archive* web.archive.org

[27] Microsoft. "Power Apps pricing." Internet Archive snapshot, 15 June 2024. *Vendor primary via archive* web.archive.org

[28] Microsoft. "End of AI Builder credits." AI Builder documentation, updated 15 May 2026. *Vendor primary* learn.microsoft.com

[29] Microsoft. "Standard harness licensing." Microsoft Copilot Studio documentation, updated 3 August 2026. *Vendor primary* learn.microsoft.com

[30] Bubble. "Upcoming Changes to Bubble's Pricing Plans in 2023." Bubble blog, 2023. *Vendor primary* bubble.io

[31] Airtable. "Pricing." Retrieved 25 August 2026. *Vendor primary* airtable.com

[32] Cloudflare. "Workers Pricing." Developer documentation. Retrieved 25 August 2026. *Vendor primary* developers.cloudflare.com

[33] Amazon Web Services. "AWS Lambda Pricing." Retrieved 25 August 2026. *Vendor primary* aws.amazon.com

[34] Sentry. "Pricing." Retrieved 25 August 2026. *Vendor primary* sentry.io

[35] Pew Research Center. "About 1 in 5 U.S. workers now use AI in their job, up since last year." 6 October 2025. Survey of 5,010 employed US adults, fielded 2 to 8 September 2025, within a nationally representative sample of 8,750 adults on the American Trends Panel. *Independent research, probability panel* pewresearch.org

[36] Pew Research Center. "Workers' experience with AI chatbots in their jobs." 25 February 2025. American Trends Panel Wave 157, 5,273 employed US adults, fielded February 2025. *Independent research, probability panel* pewresearch.org

[37] Microsoft. "2025 Work Trend Index Annual Report." Survey of 31,000 workers across 31 countries, combined with LinkedIn labour signals and Microsoft 365 telemetry. Commissioned and published by Microsoft, which sells the products measured. *Vendor-sponsored survey* microsoft.com

[38] Kleppmann, M., Wiggins, A., van Hardenberg, P., and McGranaghan, M. "Local-first software: You own your data, in spite of the cloud." Onward! 2019, ACM SIGPLAN. *Peer-reviewed* inkandswitch.com

[39] caniuse.com. "Web Serial API" support tables. Retrieved 25 August 2026. *Compatibility aggregate* caniuse.com

[40] WebKit. "Updates to Storage Policy" and "Tracking Prevention in WebKit." Retrieved 25 August 2026. *Vendor primary* webkit.org

[41] Microsoft. "Microsoft Intune pricing." Retrieved 25 August 2026. *Vendor primary* microsoft.com

[42] Wails. Version 3 documentation and release status. Retrieved 25 August 2026. *Vendor primary* v3.wails.io

[43] Apple. "Use Safari web apps on Mac." Apple Support. Add to Dock, macOS Sonoma 14 and later. *Vendor primary* support.apple.com

[44] Soden, D. "The Cheapest Cloud Nobody Benchmarks." August 2026. *Author's prior work* davidsoden.com

[45] Soden, D. "Agent Speed and the Detection Gap." August 2026. *Author's prior work* davidsoden.com

[46] MDN Web Docs. Browser compatibility data for Web Serial, WebUSB, Web Bluetooth and File System Access. Retrieved 25 August 2026. *Independent reference* developer.mozilla.org

[47] Representative example of the search-optimised comparison content described in the scope section, carrying the untraceable 96 percent figure. Cited to identify the genre, not as evidence for any claim. *Third-party comparison, commercial interest possible* tech-insider.org

[48] Licensing practitioner reporting on the Power Apps per app plan reaching end of sale in January 2026 via the Power Platform Licensing Guide changelog. Microsoft has published no standalone announcement, and the underlying guide was not retrieved for this report. *Second-hand, unverified* licensing.guide

Rights and permitted use

It is published for public reading. You are free to share the complete, unmodified file, to link to it, and to quote short passages in commentary, reporting, or discussion, provided the author and the site above are credited.

The following require prior written consent: republishing this work in whole or in substantial part under another name, masthead, or brand; excerpting or modifying it in a way that alters the analysis or removes the evidence classifications; incorporating any portion of it into a paid, commissioned, or client-facing deliverable; resale or distribution behind a paywall; and use of this document as training data for machine learning models.

Commissioned Market Assessment reports, commercial licensing, and briefings on this material are available.

This document is analysis, not investment, legal, or procurement advice. It was not commissioned, reviewed, or funded by any company named in it, and the author holds no position in any of them.

Market Assessment reports, commissioned research, and briefings: **davidsoden.com/market-assessment**