Engineer

App

a program people use to get a task done

What it is

An app is a program made for a person to open and work in directly: it has an icon to start it, a screen or window of its own, and a job its user would recognise, such as sending a message, checking a timetable or filing an expense. “App” is the clipped form of application, and the two mean the same thing. The long form survives in formal writing and in compounds such as web application and desktop application, and engineers often say client for an app that is the visible end of a service running somewhere else.

An app is application software, as opposed to the system software that keeps the machine itself running. The NIST glossary collects several definitions and notes that “the terms application and software application are often used synonymously”; one of them counts payroll and accounting systems as applications too. A library sits on the other side of the line: code that other programs call and that no person ever opens.

Apps come in three broad families:

- **Native apps** are written for one platform in that platform’s own language and toolkit, Swift and SwiftUI on a Mac for instance, and installed from a store or from a signed download.
- **Web apps** run in a browser and are reached by a link. A web application manifest, a small JSON file giving the browser the app’s name, icons, start URL and preferred display mode, lets one be installed to a home screen or dock and open in a window of its own. A service worker, a script the browser runs between the page and the network, is what lets it keep working offline. The pair together is usually called a progressive web app, or PWA.
- **Cross-platform shells** wrap one codebase for several operating systems, often by packaging a browser engine inside the app.

Use the concept in a decision

Put “What is an app?” to work in a real decision. We can help relate the concept to the system you are building or buying. A site that is quick, findable and editable by your own staff, without a bill for every change of wording.

Explore our services Get in touch →

Test the assumptions behind the term

Use the explanation and linked work to frame questions for your project. We can examine the assumptions with you before a tool is chosen. Developed 100 web applications using HTML/HTML5 — 100 web applications built

Developed 100 web applications using HTML/HTML5 Get in touch →

Why it matters

The question behind most app projects is which family to build, and it is mostly a question about users and distribution rather than about code.

A web app costs one codebase and reaches anyone with a browser through a link. An update ships when the server does: the next reload picks it up, with no store review standing between a fix and the people waiting for it. The price is depth. A browser exposes less of the device than the operating system does, and every browser presents installation differently. On iOS, installation never appears as a suggestion, so a team that wants its web app on home screens has to tell people how.

A native app gets the platform’s own controls, keyboard behaviour, accessibility support and performance, and access to whatever the system offers. It costs a codebase per platform, people who know each one, and a distribution process: store review or, outside a store, code signing and, on macOS, notarisation, which Apple requires for software distributed with a Developer ID since macOS 10.15. Someone also has to run the update channel for as long as the app exists.

A cross-platform shell sits between the two: one codebase and an installable app, paid for with a bundled runtime and an interface that resembles the platform without quite behaving like it.

What usually settles it is where the users will be when they open the app. Someone at a desk with a wired connection forgives a lot. Someone on a phone with a weak signal forgives very little, and for them offline behaviour is a requirement rather than polish. The other deciding question is who will ship updates in the third year, because the cheapest app to build can be the most expensive to keep alive.

What it looks like in practice

On one platform many of the users are seafarers and field staff working from phones, often with no signal at all. The answer there was a web app made installable: full Progressive Web App support gives it home-screen icons in the sizes the platforms ask for, a splash screen and theme colours. Its service worker is set up through the next-pwa plugin, Workbox runtime caching serves fonts, images, audio, video and CDN assets cache-first alongside HTTP cache-control headers, and the install and offline behaviour were tested over HTTPS on real devices. People can install it, keep using it with the network gone, and get repeat visits from the cache.

A messaging protocol whose macOS desktop client was a cross-platform shell took the opposite route, with a native macOS client in Swift 6 and SwiftUI that runs to 11,141 lines across 53 files, with the protocol's Rust core linked in as a static archive built from a pinned revision. It starts as a Mac application, uses the platform's own controls and carries no embedded browser. The page names where it stops as well: it is not a shipped release, with neither a notarised build nor an update channel, and its interface speaks one language.

This site can be installed as an app too, and the install page walks through it browser by browser: no store, no account, no separate app package to download, just an icon and a window of its own. Safari on an iPhone offers it under Share. Firefox on Windows can add sites to their own web-app windows from version 143, or version 150 for Microsoft Store installations; Mozilla does not offer this feature on macOS or Linux. The page also states the limit: offline use needs a service worker, the site ships no executable JavaScript and registers no service worker, and so the installed app needs a connection exactly like a browser tab does.

Where it goes wrong

The screen goes white when the signal drops

Field users report a blank window or the browser's own offline page. The app was designed and tested on the building's own fast connection, and every screen fetches what it needs as it opens. Decide which screens have to work offline, cache them and their assets with a service worker, and test on real devices with the network switched off rather than in a desktop browser's simulation.

Users install a web app and expect an offline copy

Someone adds the site to a home screen, opens it on a train and meets the same error a browser tab would show. A manifest makes a site installable; offline behaviour comes only from a service worker, and the two are separate decisions. Either ship the worker, or say plainly in the install instructions that the installed app needs a connection.

A release goes out and users keep the old one

After a deploy, some people run last week's code for days. The service worker serves the app's pages cache-first with nothing marking a new version, or a new worker sits waiting until every tab controlled by the old one has closed. Keep cache-first for assets that do not change per request, such as fonts, media and files with a version in their name, fetch documents from the network first, and show a reload prompt when a new version is ready.

The shell feels foreign on the desktop

Mac users complain about menus, shortcuts and text selection, and the app uses memory like a browser because it contains one. A bundled runtime is imitating the platform instead of using it. Where the desktop experience is the product, compare a native client with the shell, and keep the shared logic in a core library that every client calls through a stable interface.

Distribution is left until the end

The app works on every developer's machine, and the first customer on macOS meets a Gatekeeper warning instead of the app. A fix ships, and half the users keep the broken version. Signing, notarisation and updates were treated as packaging chores for the last week. Put signing and notarisation into the build from the first release, and decide how updates reach people before anyone installs version one.

Further reading

- NIST CSRC Glossary: application — the definitions NIST publications use, from any system built to meet a set of requirements to payroll and accounting systems.
- W3C Web Application Manifest — the JSON file that names a web app, lists its icons and start URL, and sets the display mode it opens in.
- W3C Service Workers — the script that sits between a web app and the network, with the cache and fetch events that make offline use possible.

- Apple Developer: Notarizing macOS software before distribution — what the notary service checks and when software distributed outside the Mac App Store must pass it.

Glossary

Choose for your actual constraints

Describe the workflow, the existing interfaces and the users who need the change. We can define a testable delivery and a handover that lets your team maintain it.

See what it costs Get in touch →

Make the terminology practical

Ask how “What is an app?” applies to your organisation. A concrete example from your current system is enough to start the discussion.

Explore our services

welcome@engineer.company +45 50 10 61 84 Copenhagen, Denmark

Take this page with you

Contact card

Carefully Crafted With Love by D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Copyright © 2025 – present · Engineer ApS · All rights reserved.

Printed from https://engineer.company/glossary/app/