PUBLISHED: SEPTEMBER 25TH · BY SHREYAS KALYANPAD

A guest surface is a piece of your product that runs inside someone else's app. Not a link out to your site. Your actual interface, in their window.

Since 26 January 2026 there is an official spec for this. MCP Apps is the first official extension to the Model Context Protocol, and it shipped marked stable. Claude and Goose render it today, Visual Studio Code has it in Insiders, and ChatGPT followed in the same week. A tool can now hand back a working interface instead of a wall of text.

Most teams read that and start porting their dashboard into it. Wrong move.

> *You are not shipping a product in there. You are shipping one decision, in a room you do not own.*

Five things the host takes off you

Everything you designed has to fit inside a box somebody else taped out.

1. The type and the colour are not yours

What teams ship: the brand font, the brand gradient, the brand button.

Why it fails: the host hands your app a set of CSS variables and expects you to use them. The spec fixes the names, things like --color-text-primary, --font-sans and --border-radius-md. OpenAI goes further in its own UI guidelines: inherit the system font stack, do not use custom fonts even in fullscreen, and do not put brand colour behind text.

Ship instead: read the host variables, keep a sensible fallback for each one, and spend your brand on a single accent.

2. There is no second screen

What teams ship: tabs, a back button, a drill-in from the list to the detail.

Why it fails: an inline card has no navigation model. OpenAI's rules are blunt about it: no deep navigation or multiple views in a card, no nested scrolling, and two actions at most, one primary and one optional secondary.

Ship instead: pick the one decision the person came for, and cut everything that is not it. If you need a second view, that is a second tool call.

3. Your logo is already on the screen

What teams ship: a header with the logo, the product name and a settings cog.

Why it fails: the host draws that part. OpenAI tells developers not to include their logo in the response, because the app name and icon get added before the widget renders. You are paying rent on space that is already labelled.

Ship instead: delete the header. Start at the content.

4. Your app cannot reach the internet by default

What teams ship: a webfont from a CDN, images from an asset host, one small analytics call.

Why it fails: if you declare no origins, the host applies connect-src 'none' and blocks outside sources. Your CDN font, your asset host and your analytics call all fail. Camera, microphone, location and clipboard are requests rather than rights, and the spec tells apps not to assume they were granted.

Ship instead: declare the origins you really need, inline the rest, and check every permission in code before you use it.

5. Someone else decides whether you render at all

What teams ship: a flow that assumes the action runs the moment the button is pressed.

Why it fails: a host can require explicit approval for a tool call your interface starts. On Claude, Team and Enterprise owners can switch off the tool calls that render interactive connectors for everybody in the company.

Ship instead: treat the approval as part of the flow, and make sure the card still says something useful when the action is refused.

The trade

WHAT YOU GIVE UP	WHAT TO DO ABOUT IT
Fonts and colour	Read the host variables, fall back, keep one accent
Navigation	One decision per card, second view is a second call
Branding space	Drop the header, start at the content
Open network access	Declare the origins you need, inline the rest
Control of the action	Design the approval, handle a refusal

A ten minute audit

Open the thing you want to put inside a chat window. Write down the single decision a person opens it for. Now delete the header, the tabs and the second screen, and see whether what is left still answers it. If it does not, the feature is not ready to be a guest.

Checked against

Read on 25 September 2026.

- Model Context Protocol blog: MCP Apps is live as the first official MCP extension, with the client list and the security model
- SEP-1865, the MCP Apps specification: stable on 2026-01-26, the default content security policy, the theme variables and the permission model
- OpenAI developer docs, UI guidelines: display modes, two actions per card, no deep navigation, no custom fonts, no logo in the response
- Claude support: interactive connectors, inline cards and fullscreen view, and the admin switch for tool calls that render them

---