

Gen UI: Build for the era of agents.



Alan Blount
The Friendly Product Guy



Vishal Motwani
The Unfriendly Product Guy

Three times now, where users meet software has changed what we build.

The browser. The app store. Now the assistant.

LIVE DEMO [travel-a2ui](#) BUILT ON [A2UI](#) · Express · macros · renderers for every stack

Agenda

- 01 Where users are**
The browser, the app store, now the assistant.
- 02 Why UI in Agents**
Chat is a great input. Prose is a bad output.
- 03 Prebuilt UI**
MCP Apps inside someone else's chat. Example : Build one in Flutter
- 04 Generative UI**
Fast with Express. Reliable with macros.
- 05 Vibe Coding UI**
Vibe code screens and functionality using framework specific skills; Example : Flutter Skills for cross platform apps
- 06 Roadmap & your move**
Where A2UI goes next, and what to try today.

01

Where users are.

Every time it moved, what we build changed.

THE PATTERN, THREE TIMES

Users moved. Software followed.

WHERE USERS ARE	ERA	WHAT YOU SHIPPED
The browser	Web	A site. Then an app. Then a desktop in a tab.
The app store	Mobile	An app. Camera, location, offline, in your pocket.
The assistant	Agents	? ? ?

Each was a better front door than the last, not a better renderer.

WHAT EACH ONE GAVE YOU

Reach, then trust, now intent.

BROWSER

Reach

A URL, no install. You stopped shipping discs and started shipping deploys.

APP STORE

Trust

Review and signing bought you the camera, the GPS, the secure enclave.

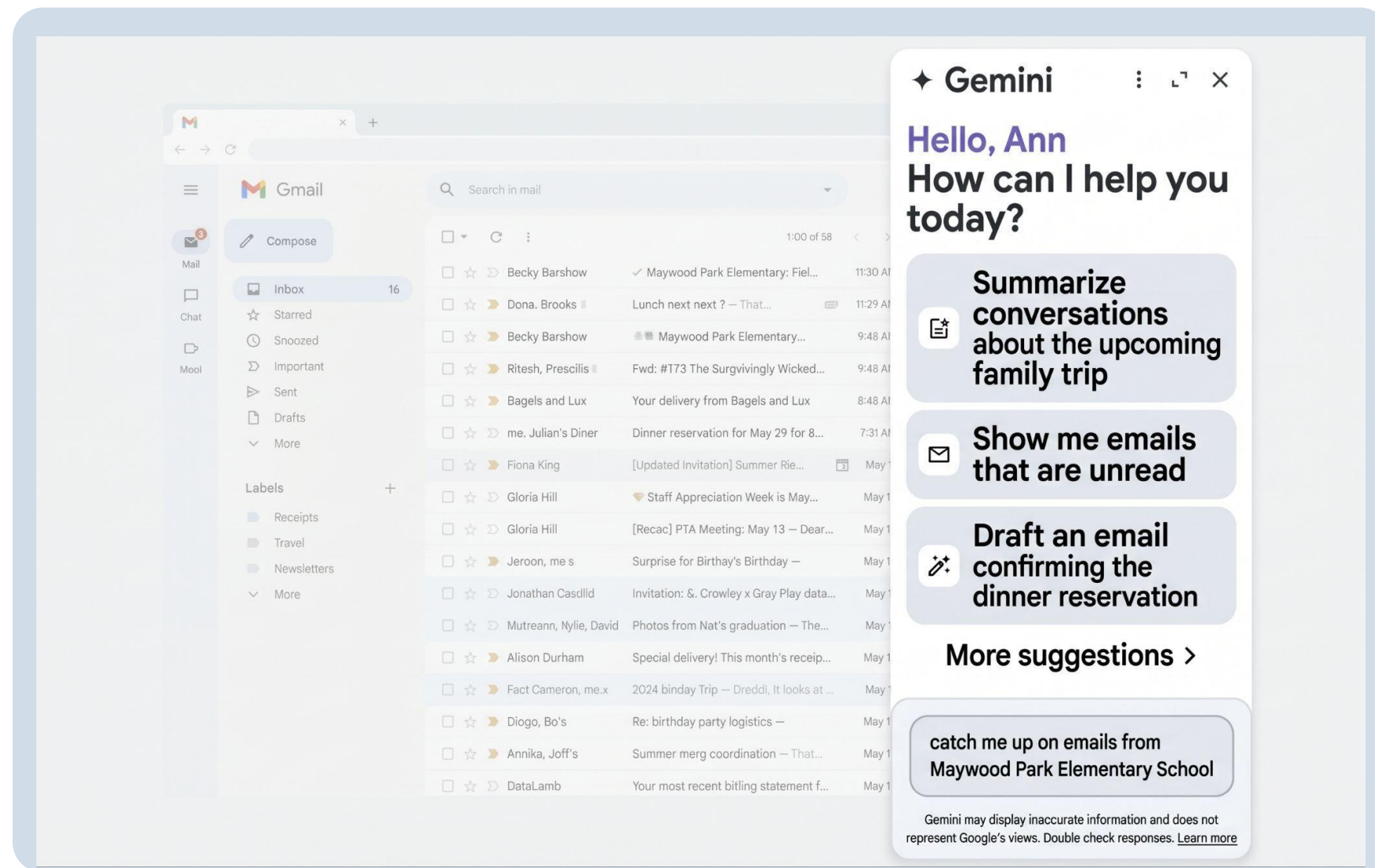
ASSISTANT

Intent

Users stop navigating. They say what they want, and the assistant picks what to reach.

SO WHAT DO YOU BUILD TODAY?

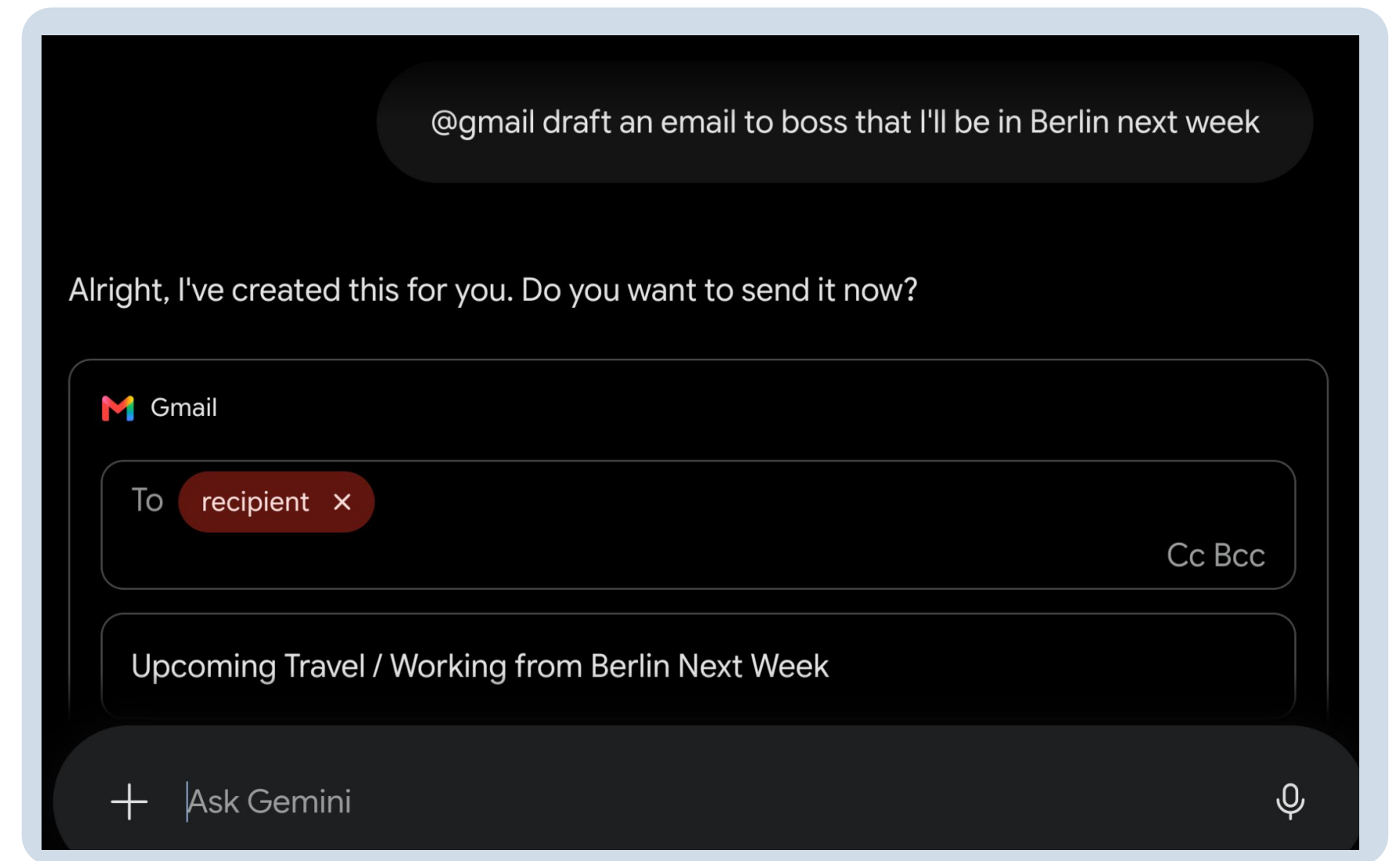
Two ways to meet users in the agent era.



01 Bring an agent into your app

e.g. Gemini inside the Gmail app

Google's own agents already answer with UI. **Most agents still answer with text.**



02 Bring your app to super-agents

e.g. Gmail inside the Gemini app

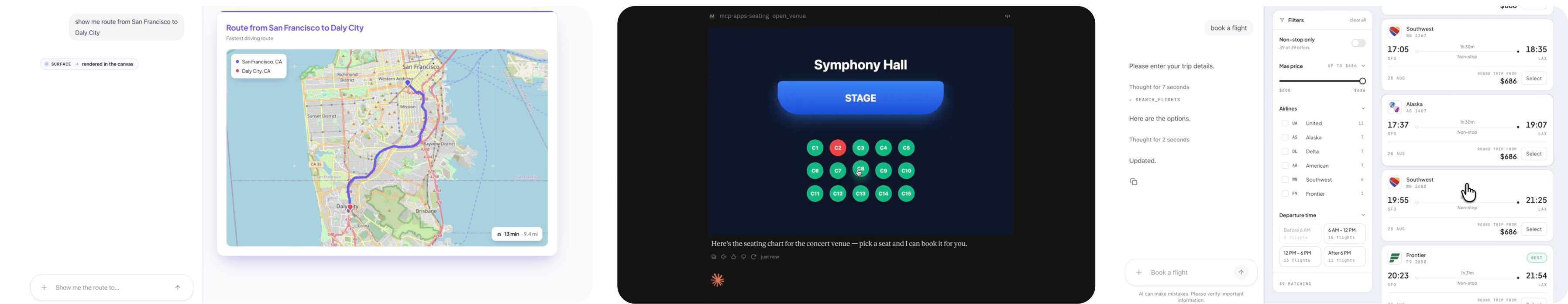
02

Why UI in Agents.

We have had this exact argument before.

WE HAVE HAD THIS ARGUMENT

“Why would you need a GUI? You have a terminal.”



A terminal makes you hold the state in your head. A screen shows it to you.

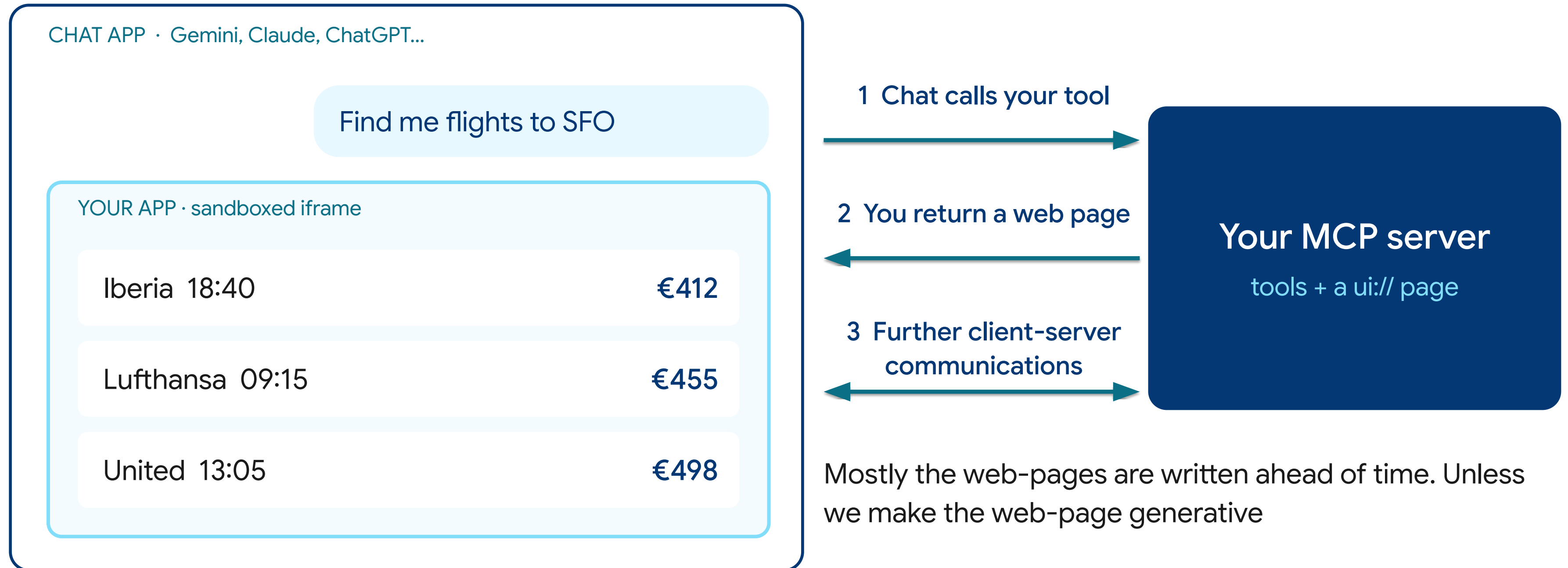
03

Prebuilt UI.

MCP Apps that run inside someone else's chat. You can build one in Flutter today.

FIRST, WHAT IS AN MCP APP?

A web page your server hands to someone else's chat.



CREDIT WHERE IT IS DUE

Why an MCP app is a good idea.

01

It's a webview. That's freedom.

Anything you built for the web ships into a conversation: your charts, your 3D viewer, last year's screen.

02

You decide what the model learns.

The page reports back only what you choose. The model's context stays small.

Go build one. Then find the ceiling.

MCP APPS IN FLUTTER, TODAY

It's just a web page In Flutter : flutter build web --wasm

BUILD

Your widgets, no rewrite.

SERVE

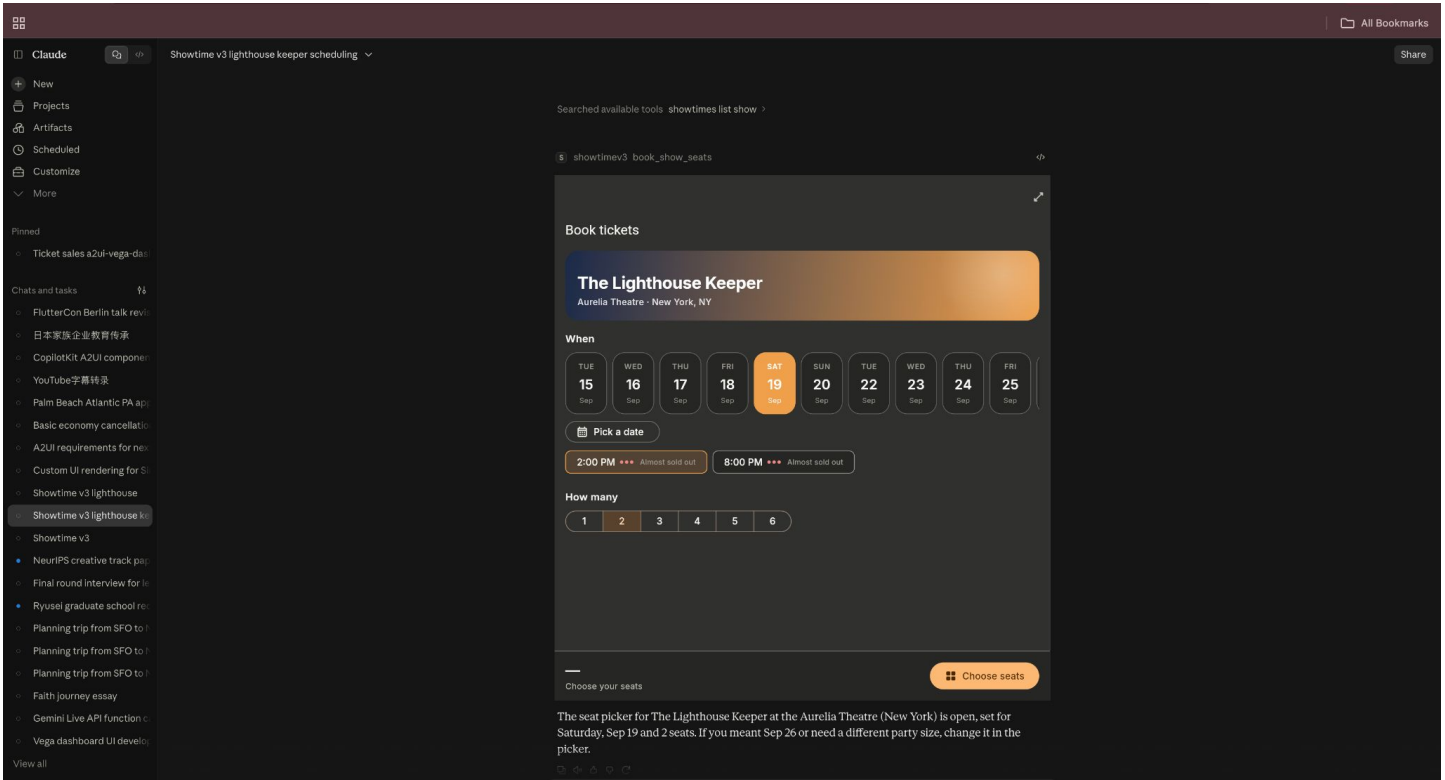
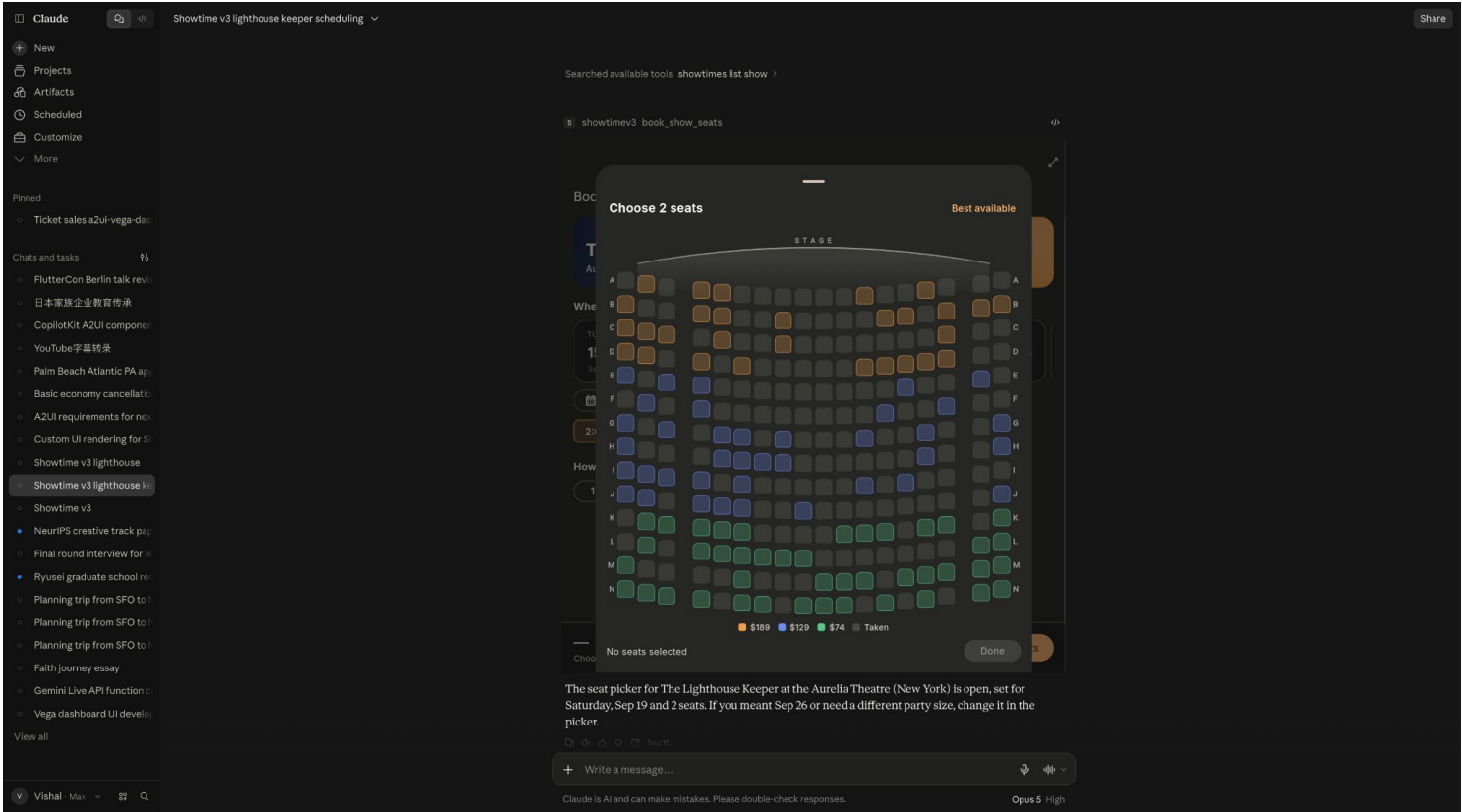
Serve it as the ui:// page.

BRIDGE

Handshake, then results.

No A2UI. No catalog. No model writing UI.
Your widgets, delivered through someone else's chat.

Web: [here](#)



THE CEILING

What Prebuilt UI in MCP Apps can't do.

01

It isn't generative.

You wrote the page ahead of time. The model picks which app, and cannot compose it to user's demands and needs.

Someone had to build that exact screen already.

02

It isn't native.

It's HTML in a box. It never gets the host's own widgets.

Three apps in one chat: three design systems.

Next Up: Generative UI sends a description.

Either MCP Apps draws it in Web View or the Host draws it natively.

04

Generative UI.

Two principles: fast and reliable. A mechanism for each.

WHY GENERATIVE UI?

“Book a flight from Berlin to San Francisco, and two tickets back.”

Now open Google Flights.

There is no field for it. Passenger count is one number for the whole trip.

That form is doing its job. It serves the case the team could anticipate.

THE INPUT IS EASY

Chat takes that sentence perfectly.

THE OUTPUT IS THE PROBLEM

Did it hear **2 out / 2 back**, or **1 out / 2 back**?

THE WHOLE ARGUMENT

Chat is a wonderful input. Prose is a terrible output.

The model may have understood you perfectly. Prose can't show you that it did.

THE PROSE ANSWER

“I'll book two travellers from Berlin to San Francisco with return tickets.”

Two return tickets, or two returning?

A SURFACE

Out · 1 travellers · BER → SFO

Back · 2 travellers · SFO → BER

One glance. A Change button on what's wrong.

LIVE

Let's book a trip.

- 01 Type the Berlin request. The surface paints while the model is still writing.
- 02 Press Change. The choice re-opens inline, pre-filled.
- 03 Open the Wire tab: Express in, JSON out.
- 04 Then the microphone. The voice asks; the surface answers.

WHY VOICE MAKES THE CASE

Four fares read aloud is a memory test.

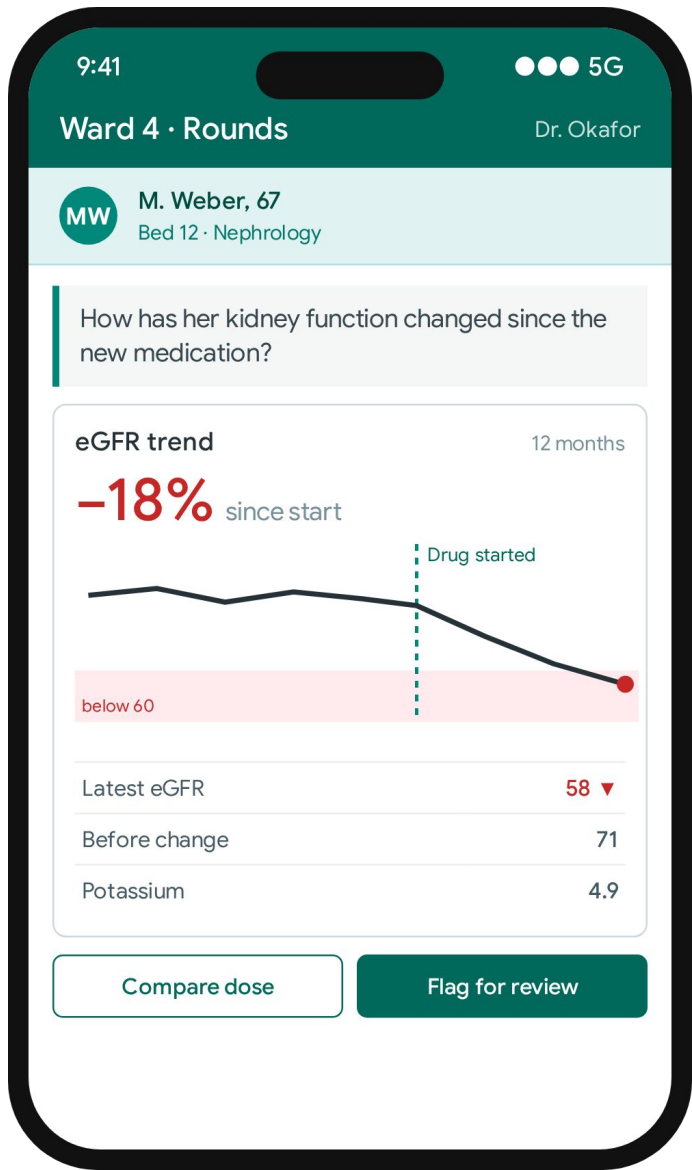
The voice takes the question. The surface is the answer.

travel-a2ui-vy7stnte2a-uc.a.run.app

github.com/vpm238/travel-a2ui

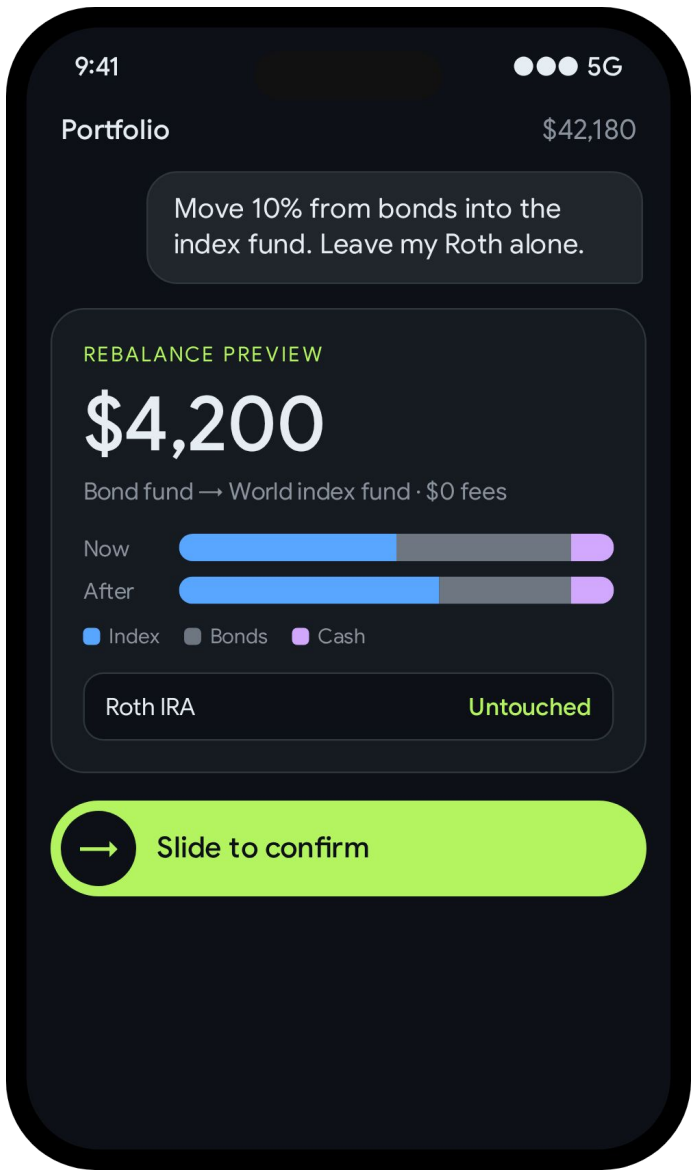
TRAVEL IS THE DEMO, NOT THE THESIS

Every domain has a long tail.



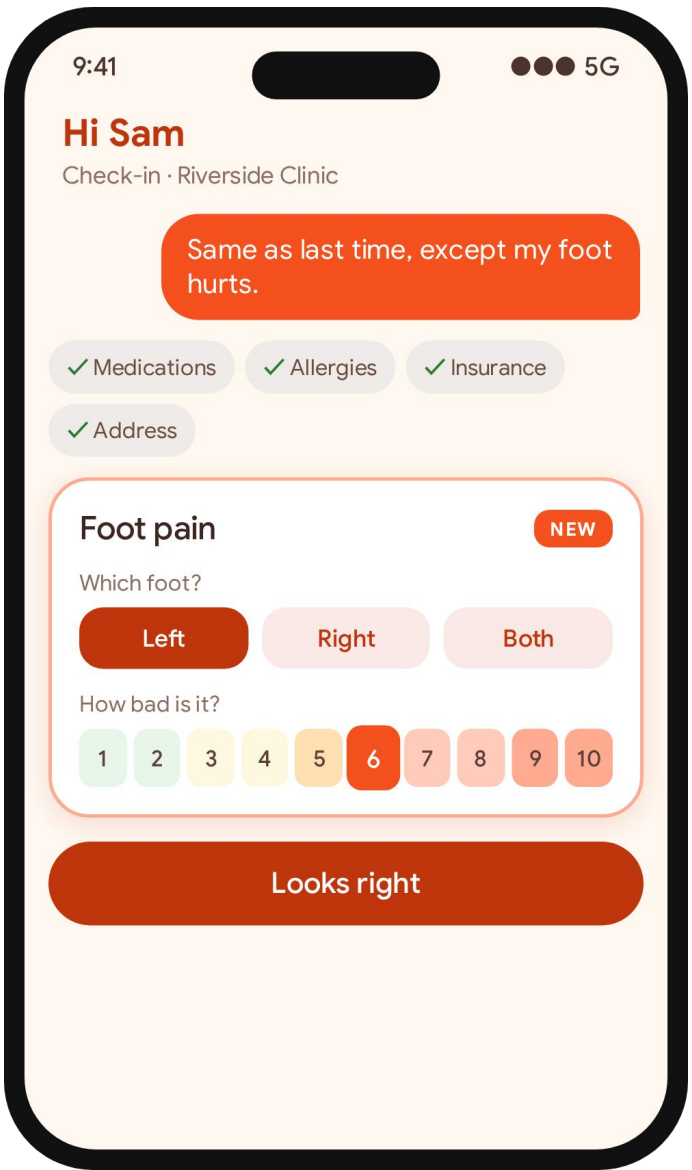
Clinical

The chart for this question



Investing

A form to check



Returning patient

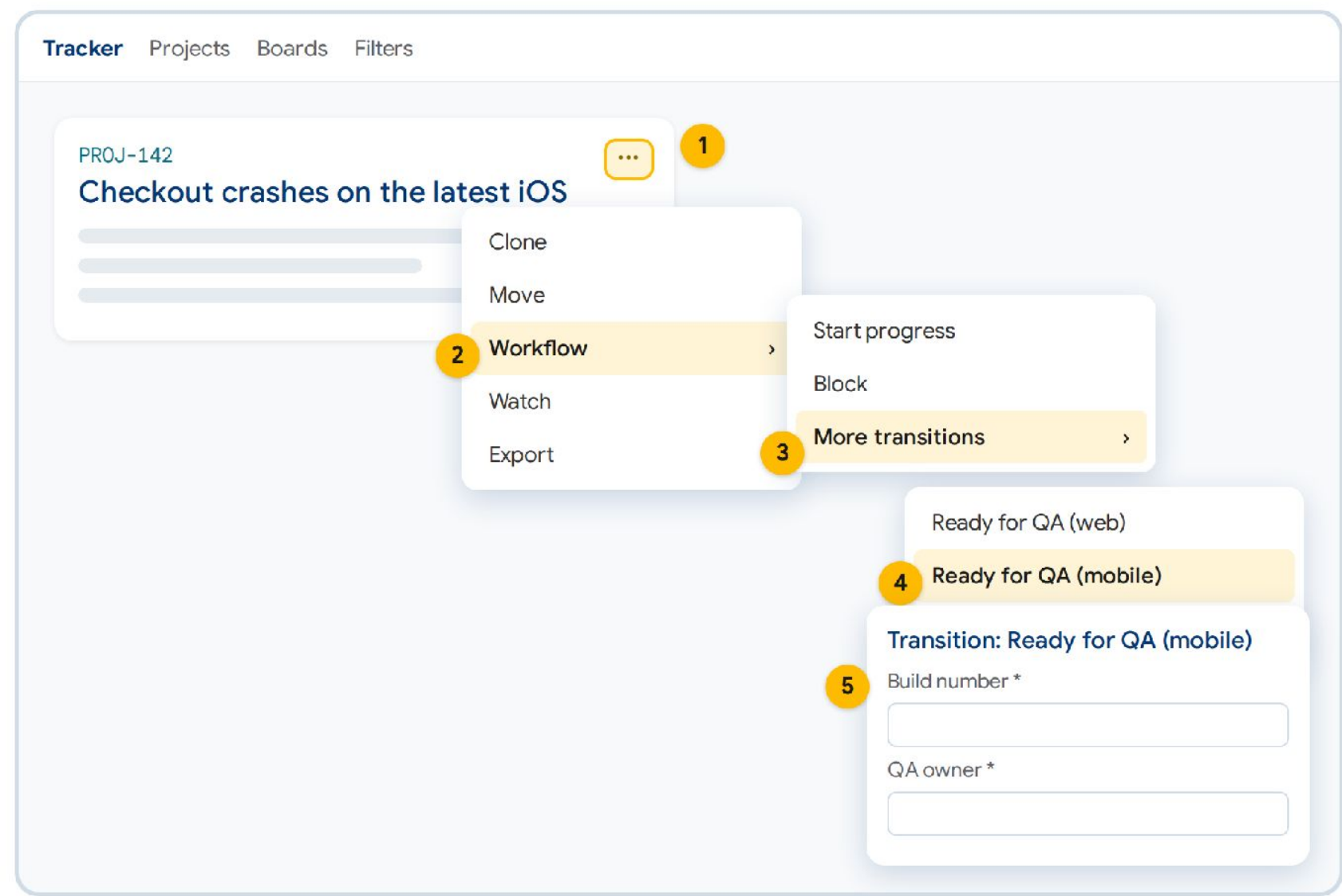
Only what changed

Open-ended in.
Verifiable out.

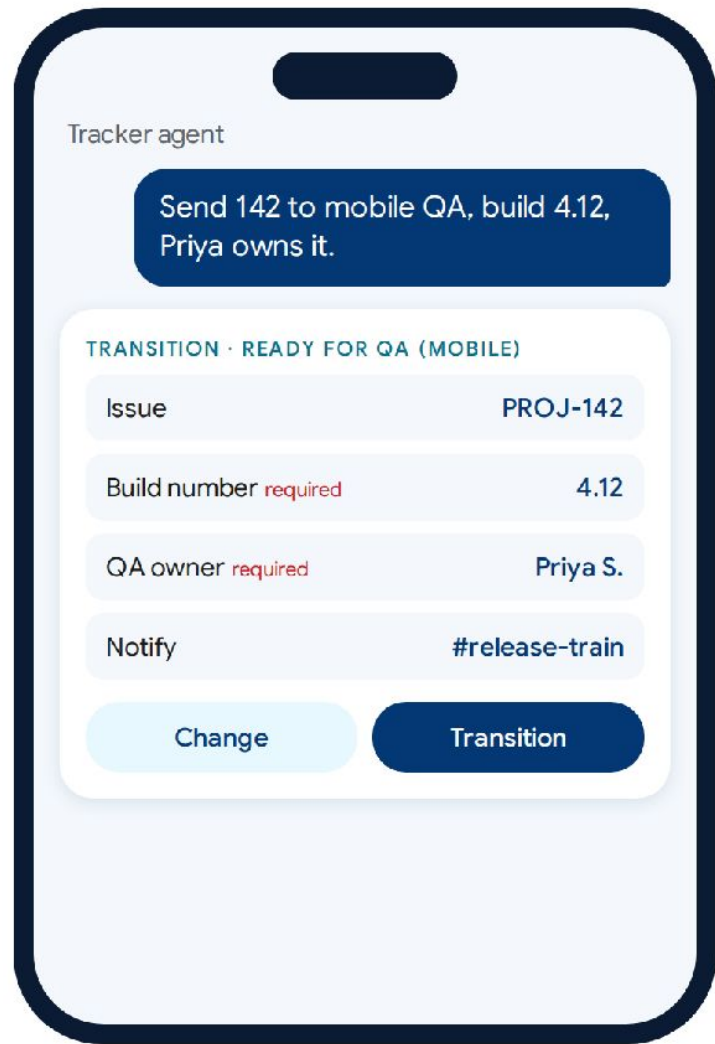
The question is new every time.
The answer is something you
can check at a glance.

OR THE OPTION EXISTS, BUT YOU CAN'T FIND IT

Sometimes the field is there. Five menus deep.



The option exists. Five clicks deep.
Different in every workspace's workflow.



One sentence. The exact form.
Built from this workspace.

Configurable products
hide their power.

Custom workflows, fields and
transitions. Every workspace
differs, so no fixed screen
shows the right option.

The agent knows this
workspace. It asks for the
exact form.

Config-driven UI is not new. LLMs driving it is.

SERVER-DRIVEN UI

A human writes the config.

At build time. For a population. You A/B it for a quarter.

GENERATIVE UI

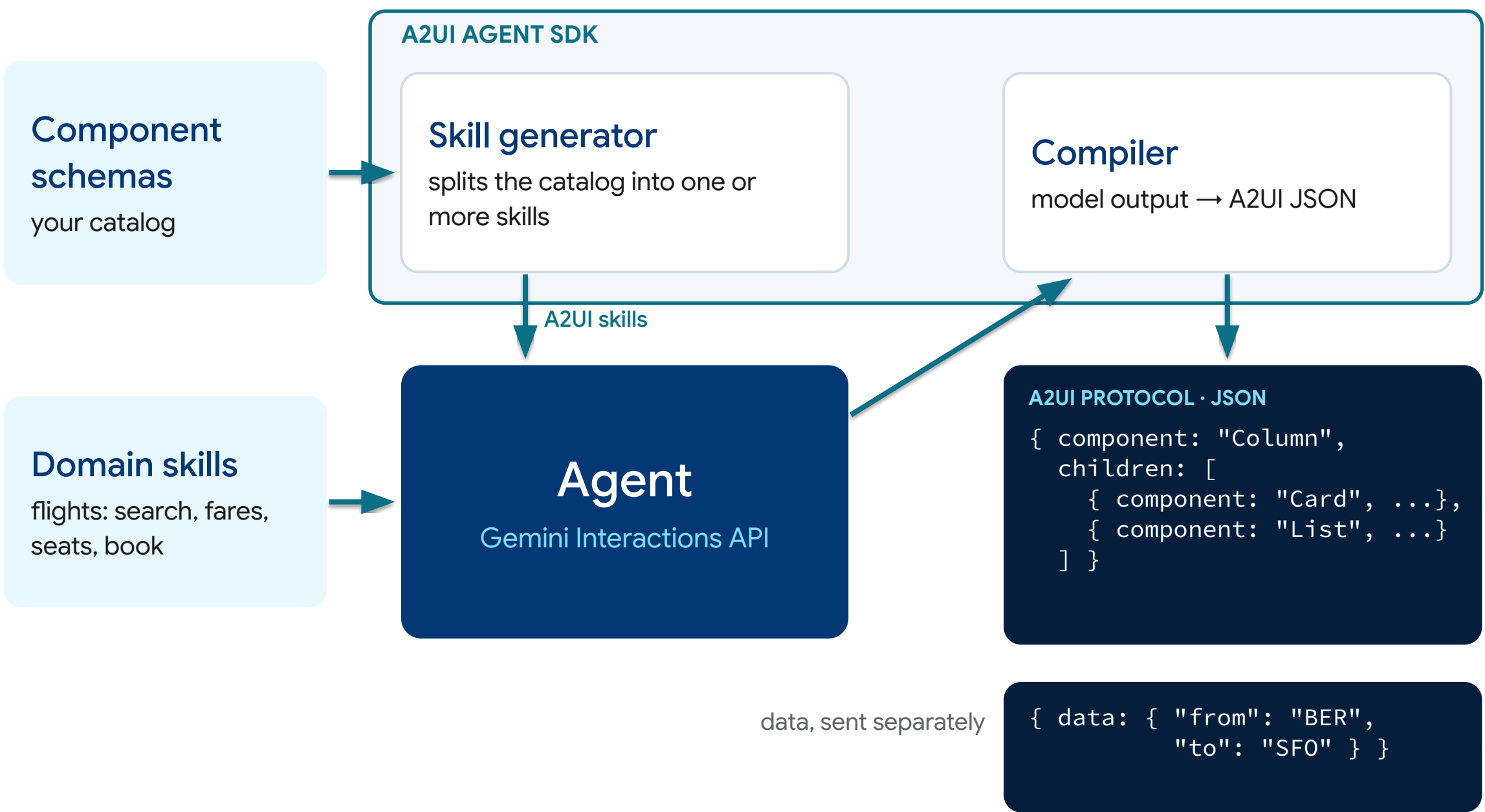
A model writes the config.

Per turn. For this person. From a catalog you control.

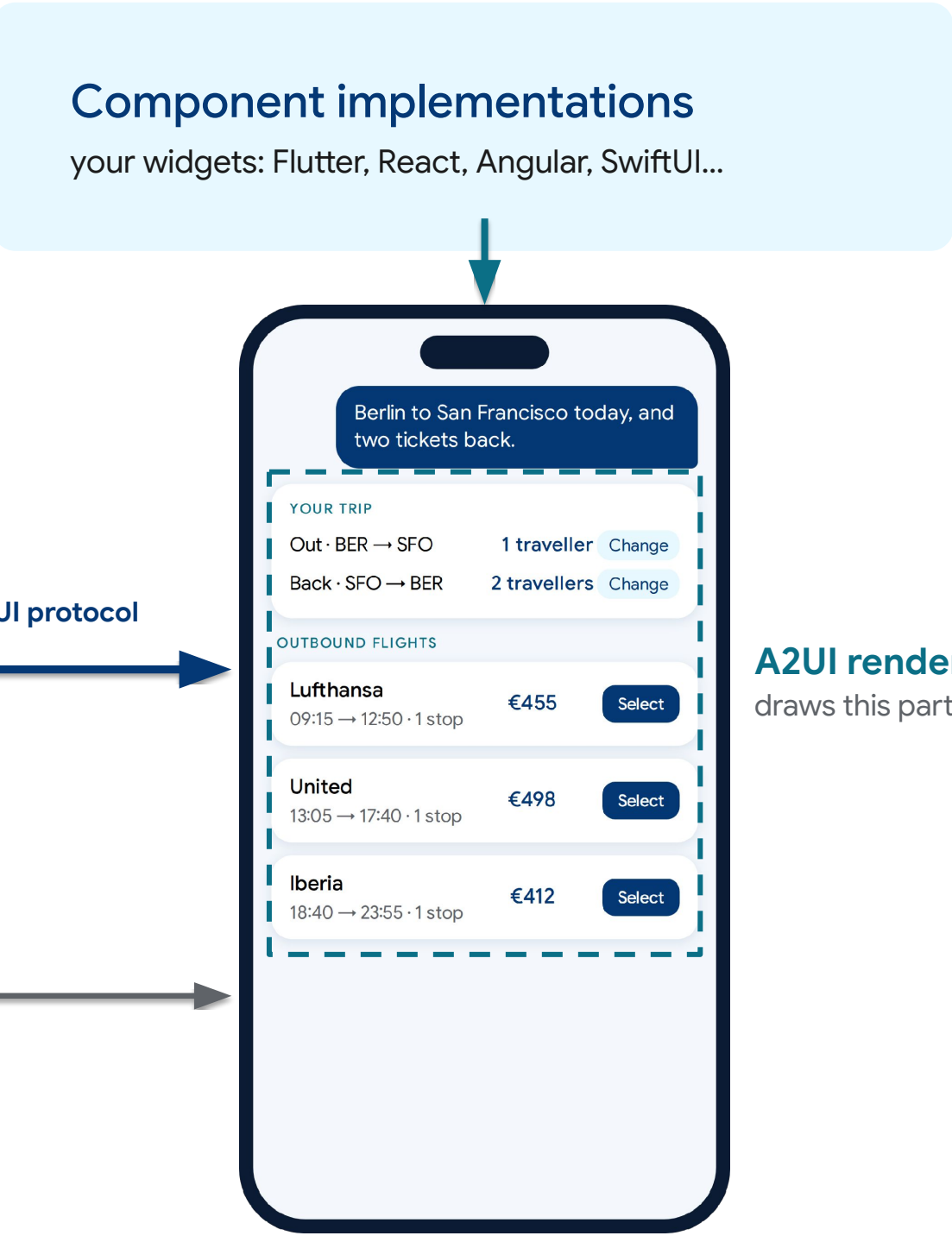
Similar to Server-Driven UI; Generative UI has to be **fast**, and it has to be **reliable**.

A2UI: an SDK for your agent, a renderer for your app.

SERVER · YOUR AGENT

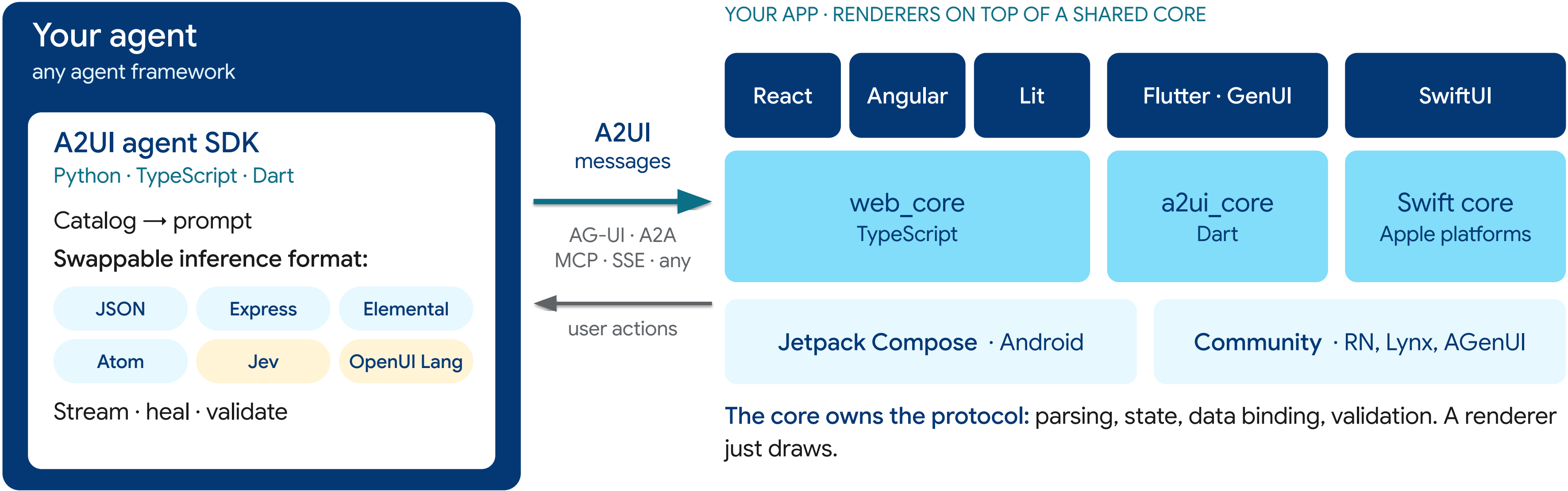


CLIENT · YOUR APP



HOW A2UI FITS TOGETHER

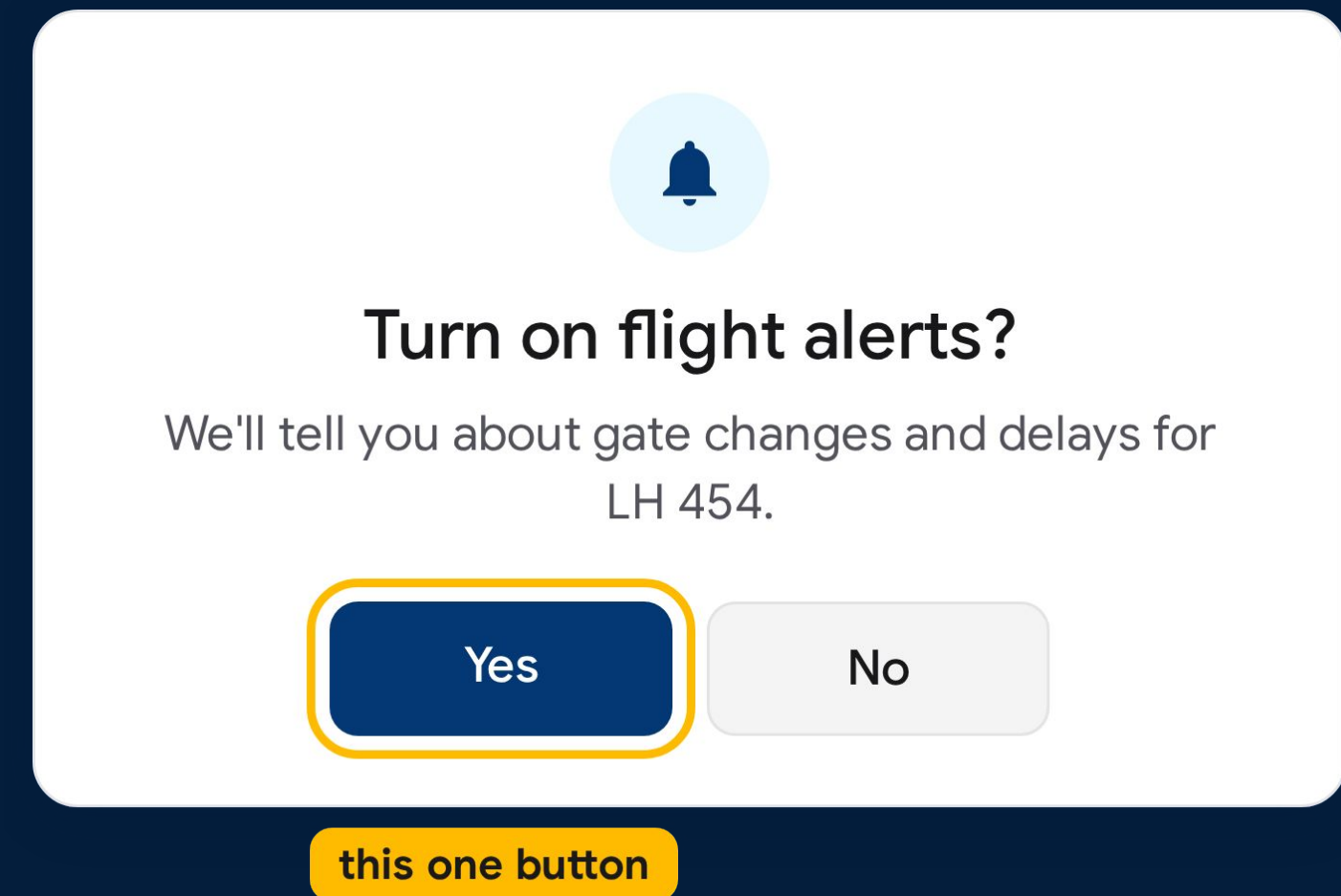
One protocol. An SDK on each side.



Next Up : Evolution of A2UI to v1.0

A2UI JSON is written for **renderers**, not models

```
{ "id": "yes_btn_text",  
  "component": "Text",  
  "text": "Yes" },  
{ "id": "yes_btn",  
  "component": "Button",  
  "child": "yes_btn_text",  
  "variant": null,  
  "action": {  
    "event": {  
      "name": "accept",  
      "context": {} } } }  
...
```



Verbose, and tough on small models

A simple permission card: 1,771 characters.

A2UI Express: same card, 2–3× fewer tokens

WHAT THE MODEL WRITES · THE WHOLE CARD · 397 CHARS

```
root = Card(main_column)
main_column = Column([icon, title,
    description, actions], _, "center")
icon = Icon($/icon)
title = Text($/title, "h3")
description = Text($/description, "body")
actions = Row([yes_btn, no_btn], "center")
yes_btn_text = Text("Yes")
yes_btn = Button(yes_btn_text, _, Event("accept"))
no_btn_text = Text("No")
no_btn = Button(no_btn_text, _, Event("decline"))
```

Express compiler

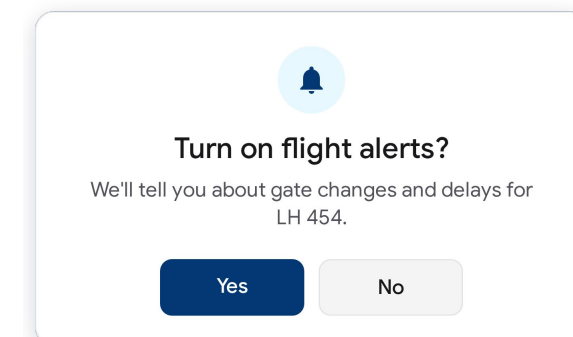
Agent side. Maps arguments using your catalog's schema.

Standard A2UI v1.0 JSON

The same format as the last slide.

Your renderer

Untouched. It never sees Express.



UNCHANGED

Renderer

Catalog

Data bindings

Events

Wire protocol

Primitives are the wrong words for your domain

ONE FLIGHT RESULT, FROM PRIMITIVES

```
root = Card(col)
col = Column([top, times, stops, sel])
top = Row([airline, price])
airline = Text($/flights/0/airline, "h3")
price = Text($/flights/0/price, "h3")
times = Row([dep, arrow, arr])
dep = Text($/flights/0/depart)
arrow = Text("→")
arr = Text($/flights/0/arrive)
stops = Text($/flights/0/stops,
"caption")
sel_t = Text("Select")
sel = Button(sel_t, _, Event("select"))
```

Lufthansa €455

09:15 BER → 12:50 SFO

1 stop · FRA

Select

United

13:05 BER

17:40 SFO

1 stop · EWR

€498

Choose

Iberia · 1 stop

18:40 → 23:55

€412

Book

Wrong vocabulary: Card and Text, not flights and fares.

Drift: the same result, built differently each time.

Fixed catalog: you can't add a widget to someone else's renderer.

Give the model bigger words

inference catalog

FlightCard

FareCompare

SeatMap

TripSummary

@macro
expands on your agent,
before anything is sent

rendering catalog

Card

Row

Column

Text

Image

Button

The model writes
`@FlightCard(flight={"path": "/flights/0"})`

The renderer gets
Card · Row · Text · Button,
still bound to `/flights/0`

Lufthansa €455
09:15 → 12:50
1 stop · FRA
Select

United €498
13:05 → 17:40
1 stop · EWR
Select

Everything else new in 1.0.

Call server-side functions

`callAgentFunction`: the UI calls your agent exposed tools and gets a typed result back.

Web components

`web_core` enables standard web components based catalog. Drop A2UI into any web page.

Extensions

Attach your own metadata to surfaces, components and catalog entries.

Mix catalogs

One surface can combine components and functions from several catalogs.

Catalogs that teach

Design instructions and allowed parent/child rules live in the catalog itself.

One-message surfaces

Components and starting data arrive together in a single `createSurface`.

Also: validation with severity, ARIA live regions, functions that require a user tap.

[What's new in 1.0 →](#)

05

Vibe Coding UI.

Your coding agent can already write Flutter. Give it Flutter's skills.

VIBE CODE APPS

Example: Flutter skills, in the agent you already use.

Antigravity

Claude

Cursor

Codex

GitHub Copilot

WHAT YOU GET

Official Flutter & Dart skills.

Layouts, state, widget tests: loaded on demand, so the agent writes Flutter the way the Flutter team would.

INSTALL · CLAUDE CODE

```
claude plugin marketplace add  
  flutter/agent-plugins  
claude plugin install  
  dart-flutter@dart-flutter
```

Every agent, one page: docs.flutter.dev/ai/get-started

Pick by latency budget.

	PREBUILT UI	GENERATIVE UI	VIBE CODING UI
Who writes it	you, ahead of time	the model, per turn	an agent, with you
Latency	~0, cached	1–2 seconds	minutes
Failure mode	wrong for the case	wrong component	a broken build
Trust	highest	bounded by catalog	read the diff
Build with	flutter build web	a2ui-flutter (genui)	Flutter skills
Good for	the predictable	the long tail	new screens, tools

06

What's next, and your move.

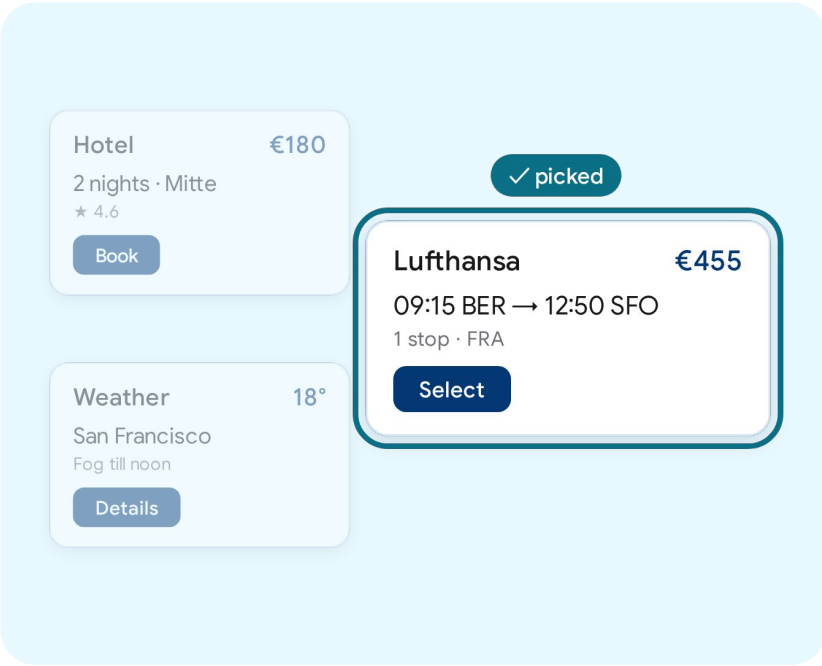
Where A2UI is going, and what to try.

THE SPECTRUM · HOW MUCH DOES THE MODEL DECIDE?

Four ways to put UI in front of an agent.

More control the developer decides

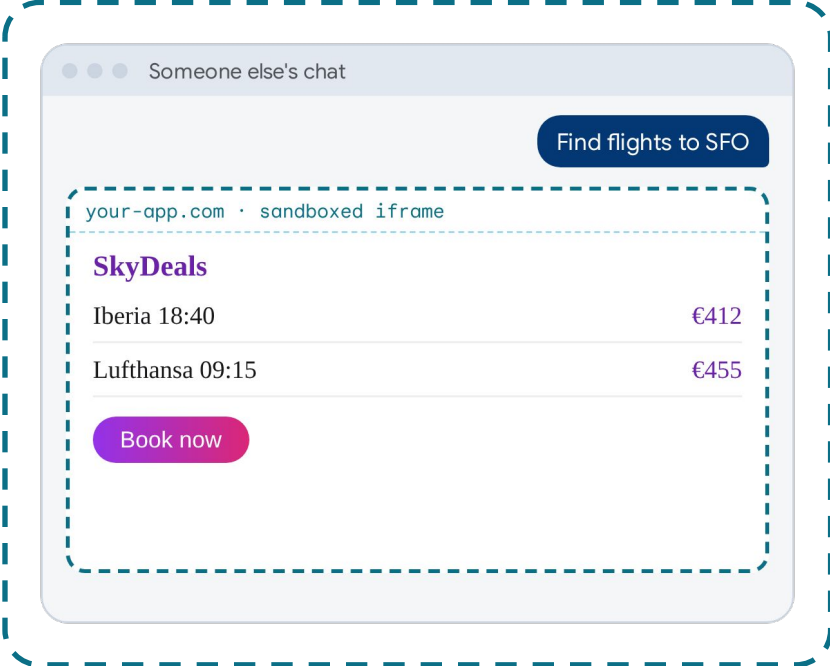
the model decides More freedom



CONTROLLED · PREBUILT UI

You built the screen.

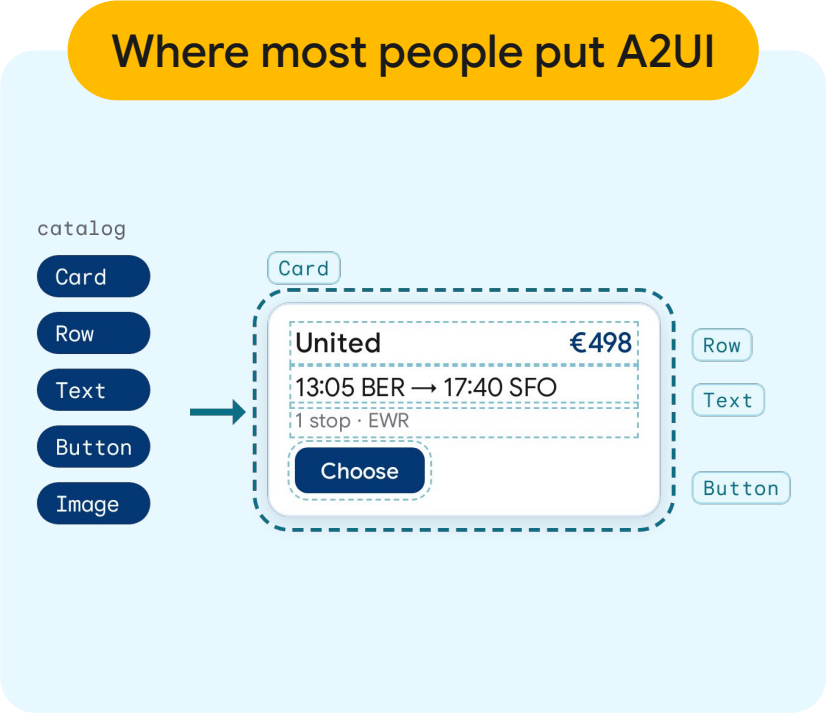
The agent picks one of your components. Polished and instant; every new case is a new component.



IN BETWEEN · MCP APPS

Your page, in their chat.

A web page your server hands to someone else's assistant, in a sandboxed iframe.



DECLARATIVE · GENERATIVE UI

Composed from a catalog.

The model assembles approved blocks on each turn. Covers the long tail.



OPEN-ENDED · VIBE-CODED UI

The model writes the UI.

New code every time. Anything goes, including off-brand and broken.

THE CORRECTION

A2UI isn't one point. It's the whole line.

A2UI one catalog · one renderer · every point on the line

```
// the model writes one line (Jev)
FlightCard(flight: /flights/0)
```

Lufthansa€455

09:15 BER → 12:50 SFO

1 stop · FRA

Select

your component · fastest

CONTROLLED

Your components, picked fast.

Finished components in the catalog. The model writes one line, and Jev keeps it fast.

Any MCP host

Find flights to SFO

MCP App · A2UI renderer (React / web components)

Iberia€412

18:40 BER → 23:55 SFO

1 stop · MAD

Select

application/a2ui+json

MCP APPS

A2UI inside MCP Apps.

Send A2UI over MCP for the host to draw natively, or ship a React or web-components renderer in the app.

```
# Express · 2-3× fewer tokens
root = Card([top, times, sel])
top = Row([Text("United"), Text("€498")])
sel = Button("Choose") # or @FlightCard macro
```

United€498

13:05 BER → 17:40 SFO

1 stop · EWR

Choose

DECLARATIVE

The one you expect.

The model composes from your catalog. Express keeps it fast. Macros keep it reliable.

```
coding agent
> add a seat picker to checkout
✓ a2ui skill · reading your catalog
✓ built from: Card · SeatMap · Button
```

Pick your seat12C

Confirm

your catalog · your design language

OPEN-ENDED

Vibe code with your components.

A coding agent with the A2UI skill builds new screens from your catalog, in your design language.

One catalog, one renderer. You choose how much the model decides.

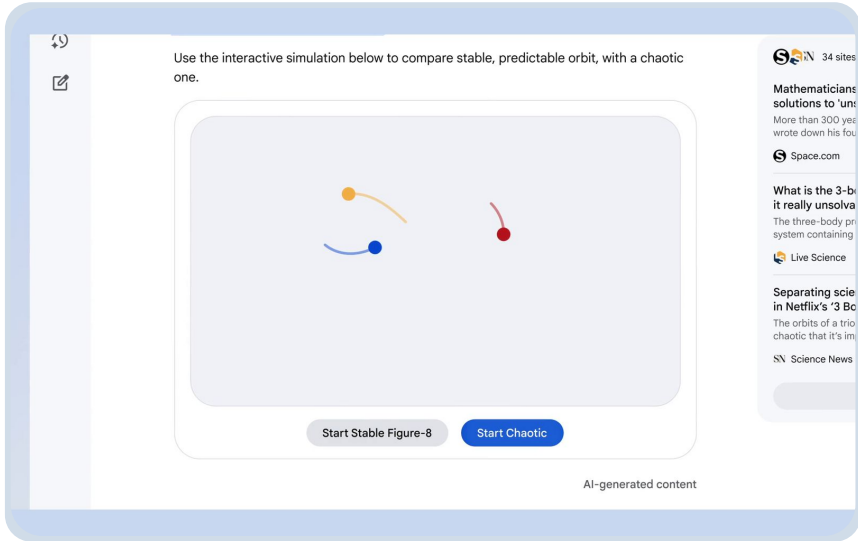
[A2UI + MCP Apps](#) · Google Developers Blog

NOT A PREDICTION

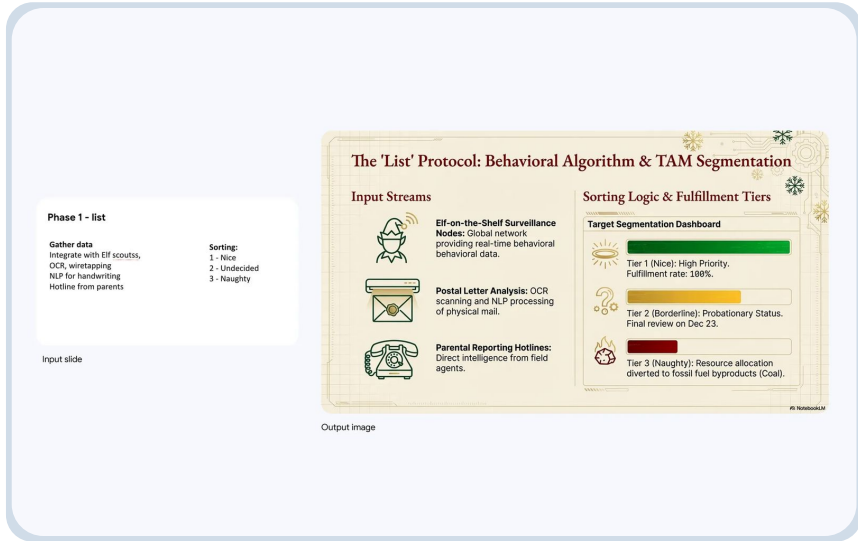
Generative UI is already shipping across Google.



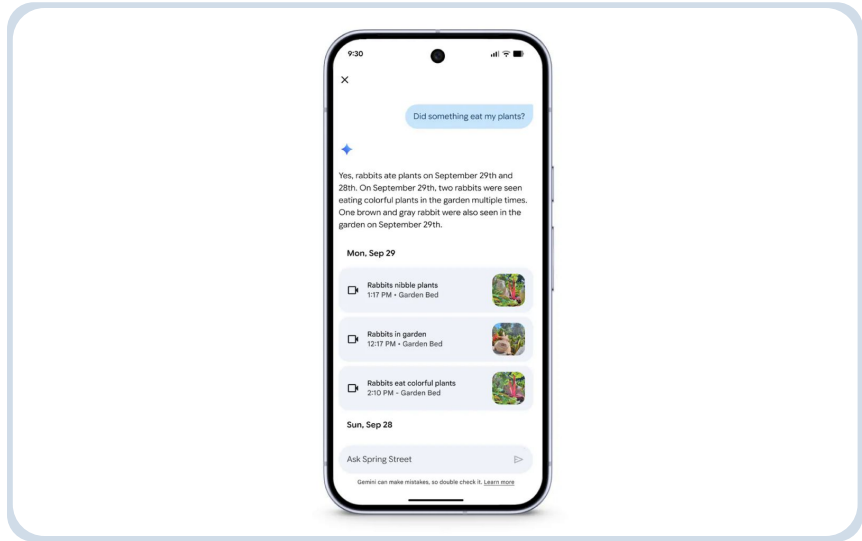
Gemini app Dynamic View



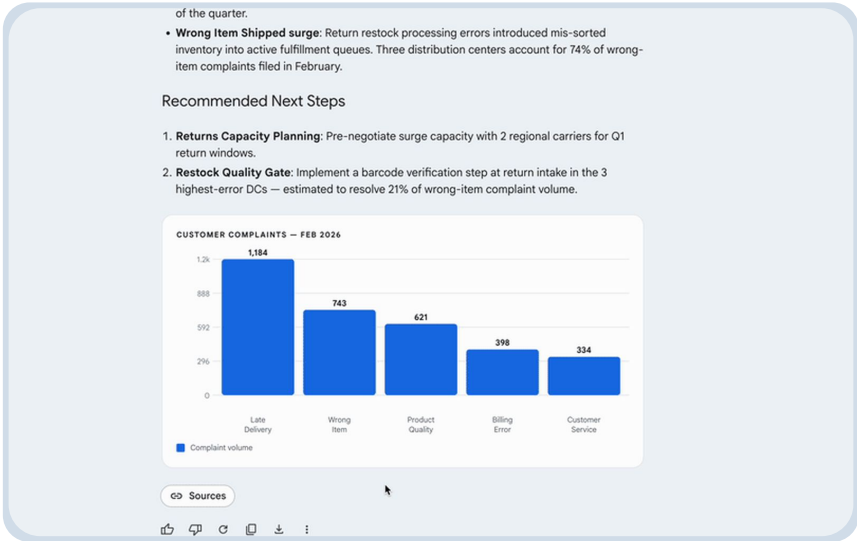
Search AI Mode simulation



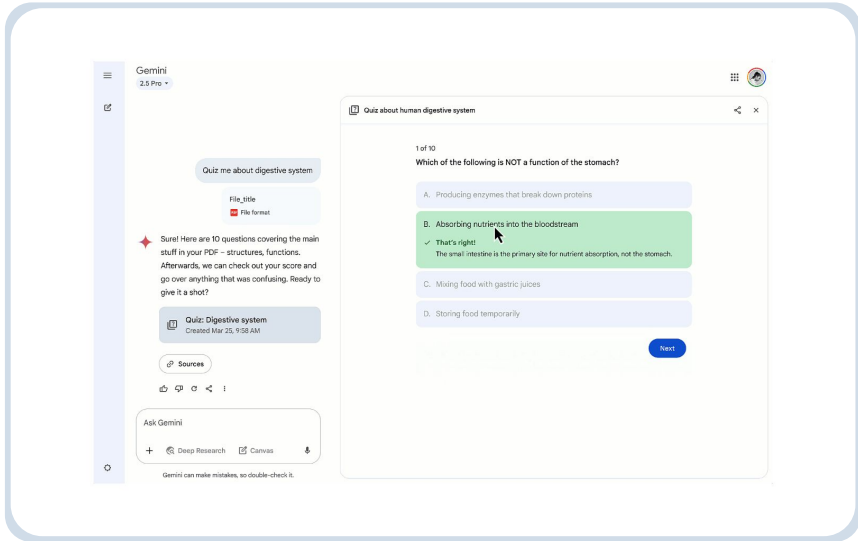
NotebookLM Slide Decks



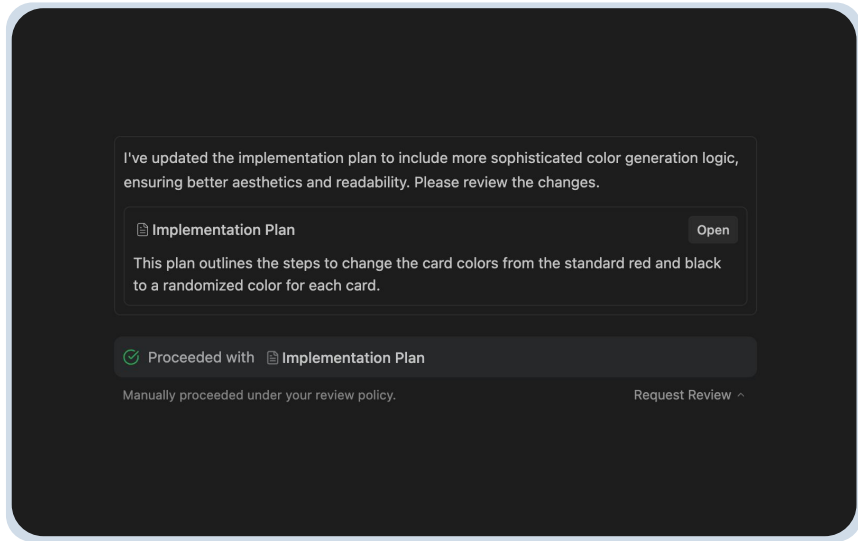
Google Home Ask Home



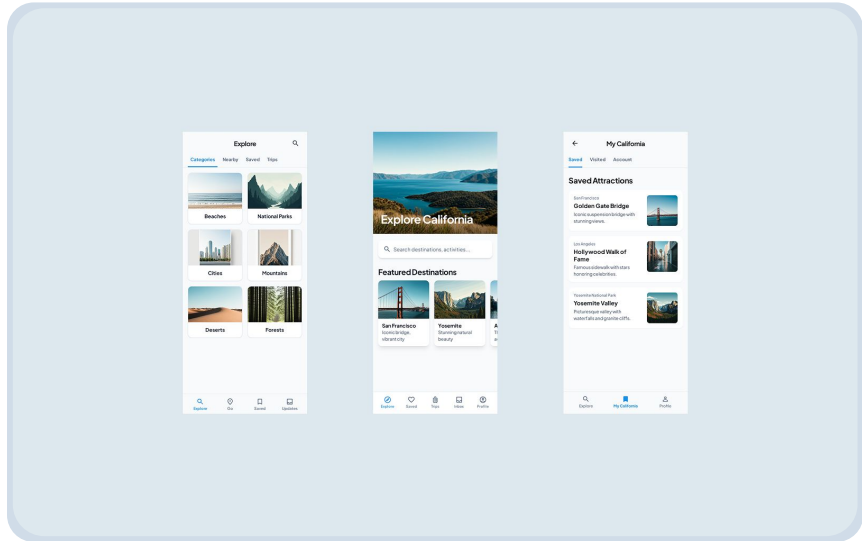
Gemini Enterprise Data agent



Gemini app Canvas quiz



Antigravity Plan to review



Stitch Generated app screens

BACK TO THE QUESTION WE STARTED WITH

Users moved. Software followed.

WHERE USERS ARE	ERA	WHAT YOU SHIPPED
The browser	Web	A site. Then an app. Then a desktop in a tab.
The app store	Mobile	An app. Camera, location, offline, in your pocket.
The assistant	Agents	???

So what do we build for the assistant?

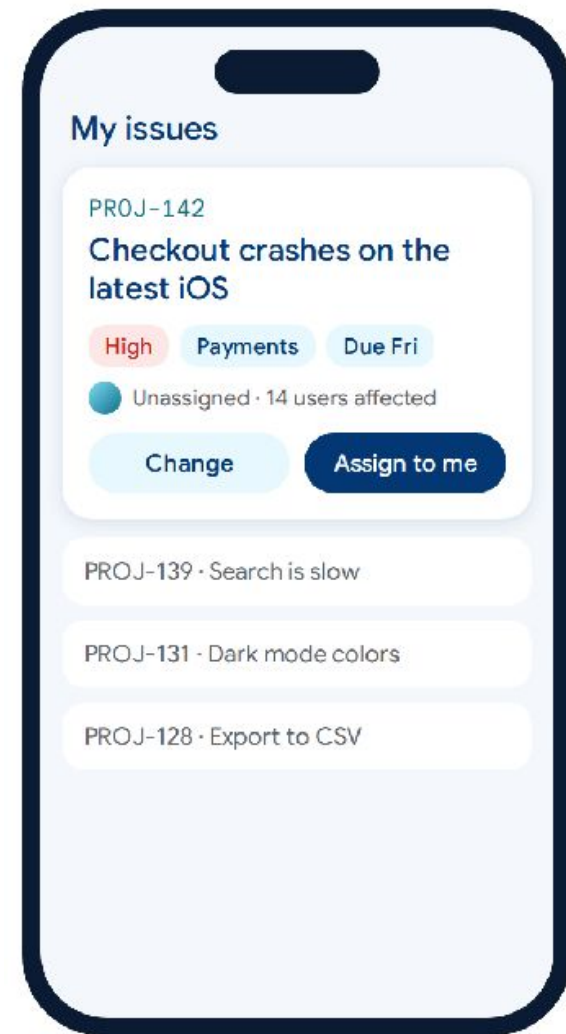
THE HONEST ANSWER

Nobody knows the final shape. One bet is safe.

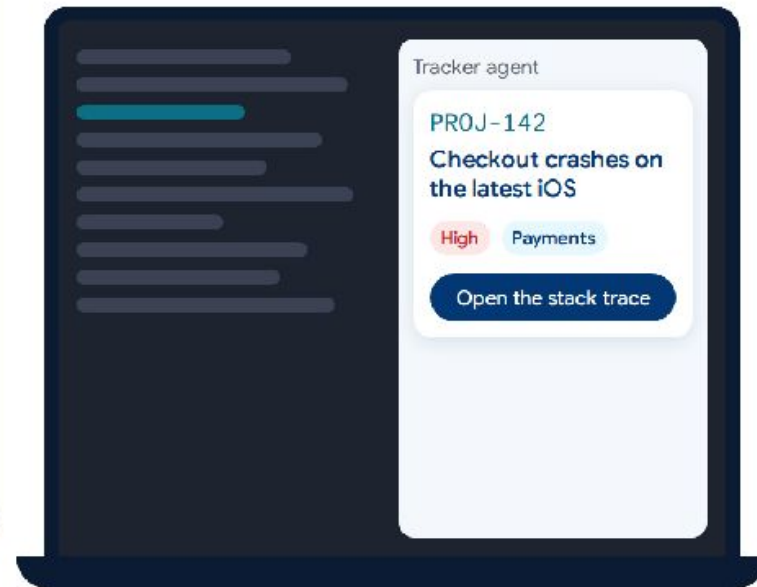
WHERE USERS ARE	ERA	WHAT YOU SHIPPED
The browser	Web	A site. Then an app. Then a desktop in a tab.
The app store	Mobile	An app. Camera, location, offline, in your pocket.
The assistant	Agents	Services and components an agent can use.
01 Services & components Your capabilities and your design system. Not screens.	02 Reachable by any agent MCP, A2A, skills, App Intents, WebMCP: the door varies by agent.	03 Rendered anywhere One UI description, drawn natively on each surface.

THE VALUE PROPOSITION

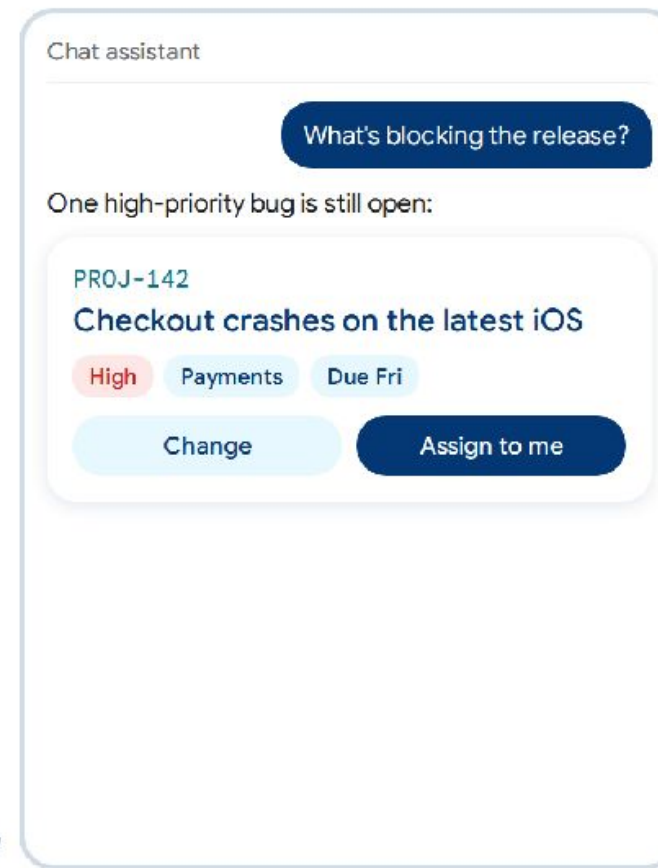
One agent. Every surface. Rich UI everywhere.



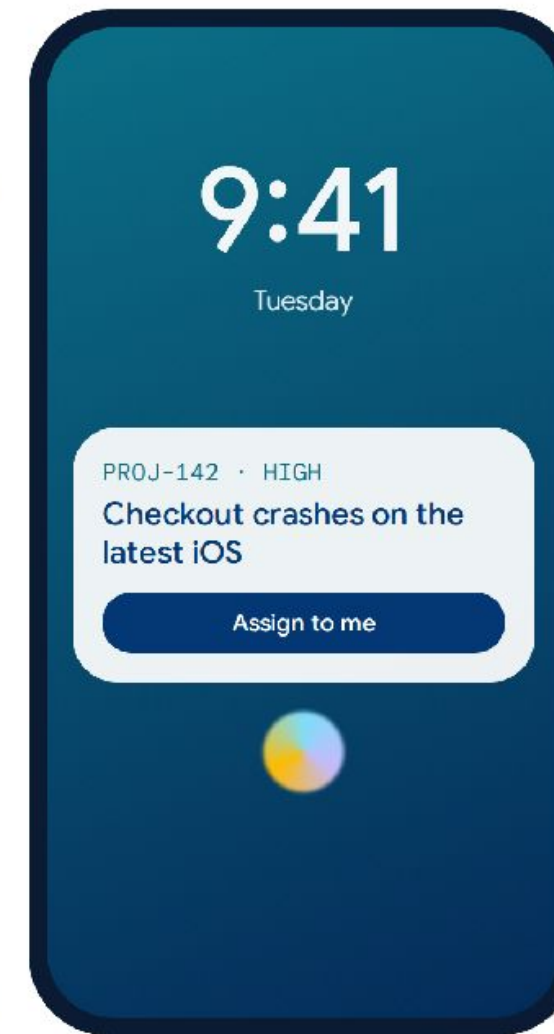
Your own app
web · iOS · Android



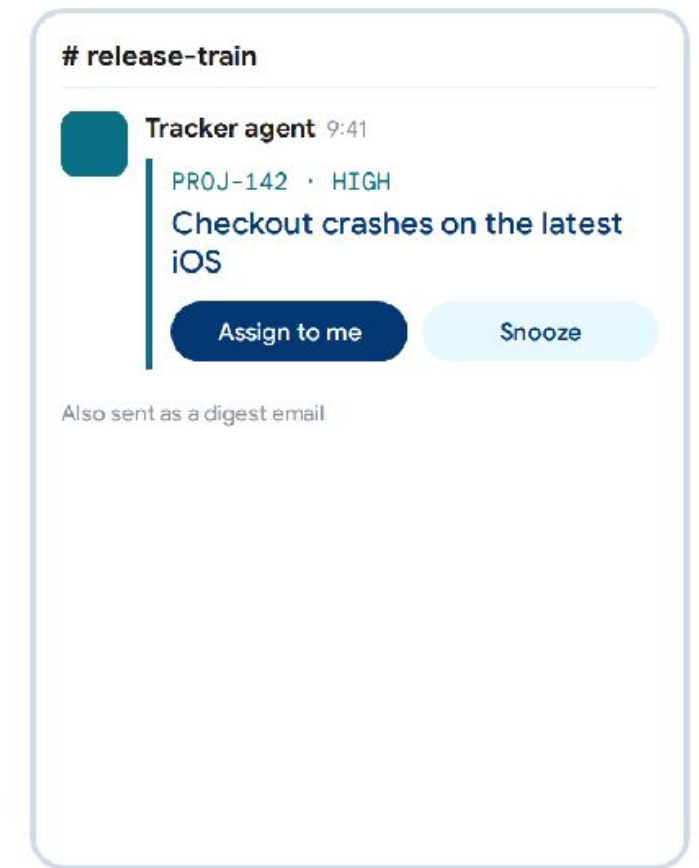
IDEs & coding agents
side panes



Chat assistants
inside MCP Apps



Voice assistants
e.g. a Siri snippet



Where teams work
Slack · Teams · email

Describe the UI once. **Every surface draws it natively.**

How A2UI reaches each surface.

Your web & mobile apps	React, Angular, Lit, Flutter, SwiftUI, Compose renderers	Today
Chat assistants	A2UI inside MCP Apps: Claude, ChatGPT, VS Code...	Today
IDEs & coding agents	MCP Apps where supported; web renderer in an Antigravity side pane	Emerging
Siri & Spotlight	App Intents snippet (ShowsSnippetView) + A2UI SwiftUI renderer	Emerging
Slack, Teams, email	Server-side renderers: A2UI to HTML or native message blocks	Roadmap

Start now: [A2UI in MCP Apps guide](#) →

Surfaces that can be remembered, and selectively teach agents.

PERSIST

An interaction is a String.

Save it with the conversation. Bring it back later without asking the model again.

TALK BACK

The UI updates the agent's context.

What the user did becomes something the agent knows, like MCP Apps do today. UI decides what model learns

Get to know what's new in A2UI.

01

Macros

Give the model your domain's building blocks. The renderer still gets primitives.

[Macros docs →](#)

02

Swappable inference

Keep the renderer. Try Express. Explore Jev.

[Express →](#)

03

Completeness in 1.0

Surfaces call server-side functions. Extensions carry your own metadata.

[What's new in 1.0 →](#)

A2UI 1.0 spec: a2ui.org/specification/v1.0-a2ui

Build something cool. Share it.

01 Fork the demo

github.com/vpm238/travel-a2ui

02 Add an A2UI renderer to your app

[Web, Flutter, SwiftUI, Compose and more](#)

03 Contribute evals

[Help us measure what good UI looks like](#)

04 Become a GenUI consultant

Build a standout industry demo; we'll feature you

Post it with **#A2UI**. We read every one.

Thank you.

Build services and components. Let any agent use them.

A2UI 1.0 spec

a2ui.org/specification/v1.0-a2ui

What's new in 1.0

a2ui.org/specification/v1.0-evolution-guide

A2UI Express

a2ui.org/specification/proposals/express/a2ui_express

Macros docs

github.com/a2ui-project/a2ui/tree/main/python/a2ui_agent/src/a2ui/transformers/macros

Jev + A2UI

github.com/dglazkov/jev2ui

A2UI in MCP Apps

a2ui.org/guides/a2ui-in-mcp-apps

A2UI renderers

a2ui.org/reference/renderers

MCP Apps spec

modelcontextprotocol.io/extensions/apps

Demo

github.com/vpm238/travel-a2ui

Vishal Motwani · Lead PM, Generative UI, Google



a2ui.org
Start here