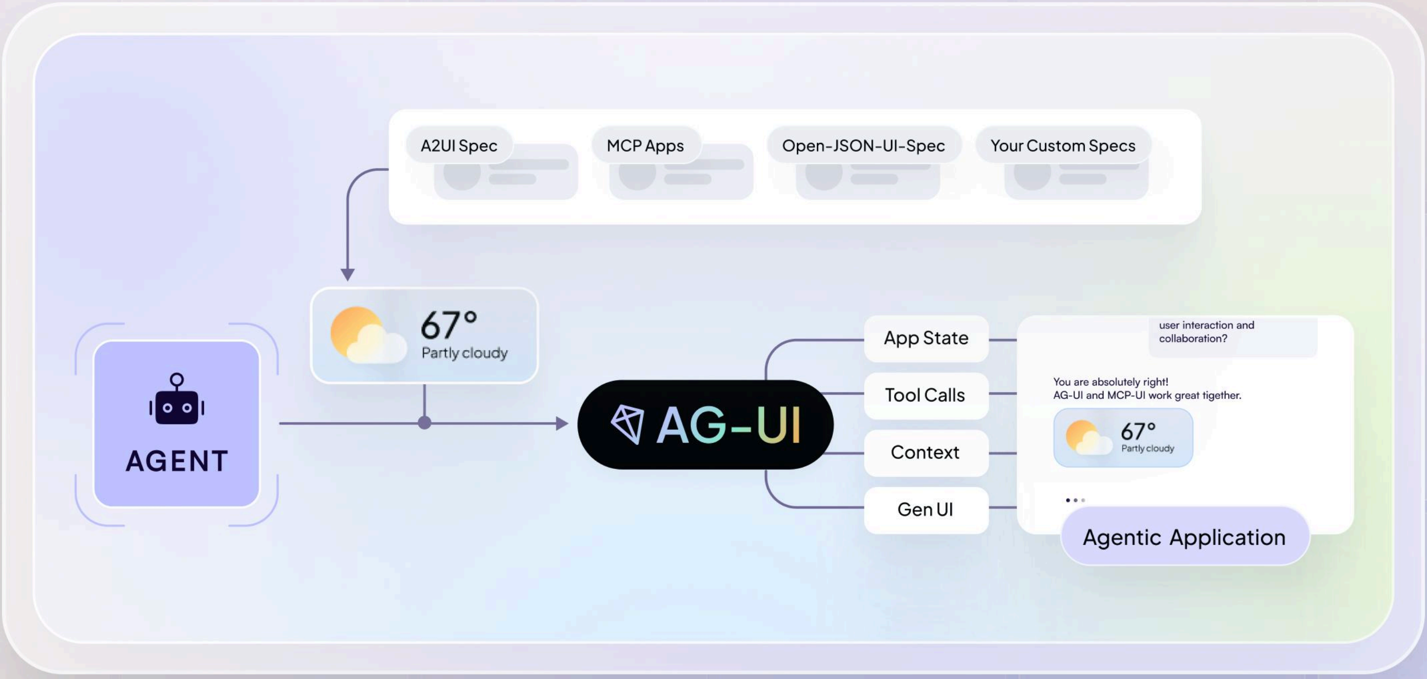
CopilotKit

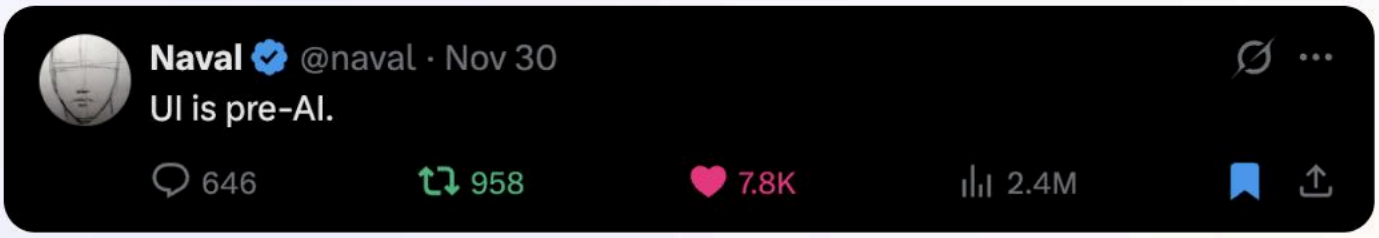
Generative UI

The 2026 guide for building modern UI in the agentic era



Swipe to next page

Why UI is Changing



Naval @naval · Nov 30

UI is pre-AI.

646 958 7.8K 2.4M

Traditional UI assumes that the **backend decides**, the **frontend renders**, and the **user clicks** → waits → clicks again.

Agents break this model.

Agents should interact with us

Agents don't just answer... they do things, and that requires UI that can show steps, ask for approvals, and keep state.

→ **AG-UI** solves this! It creates a shared contract between agents and UI so reasoning, state, and intent are visible and renderable.

Text is not enough

Chat was never designed for **interaction**, only conversation.

→ **A2UI** lets agents return interactive UI via structured JSON that the client can reliably render into native components.

LLMs can generate UIs

The UI layer can now be created on demand (per user intent), instead of being fully pre-designed for every workflow.

→ **MCP Apps** lets agents return external mini-apps (a literal container that has prebuilt UI in it)

New capabilities and types of apps

Apps are shifting from "screens you navigate" to "outcomes you request," where workflows are assembled dynamically by agents across tools and data.

→ **CopilotKit solves this!** You own the UI components, but CopilotKit handles Agent-UI loops, action wiring, readable state, human-in-the-loop control and more.

↑

This is how you turn an agent UI from simple demo to **real fullstack agentic app**.

What is Generative UI

Generative UI refers to any user interface that is partially or fully produced by an AI agent, rather than authored exclusively by human designers and developers.

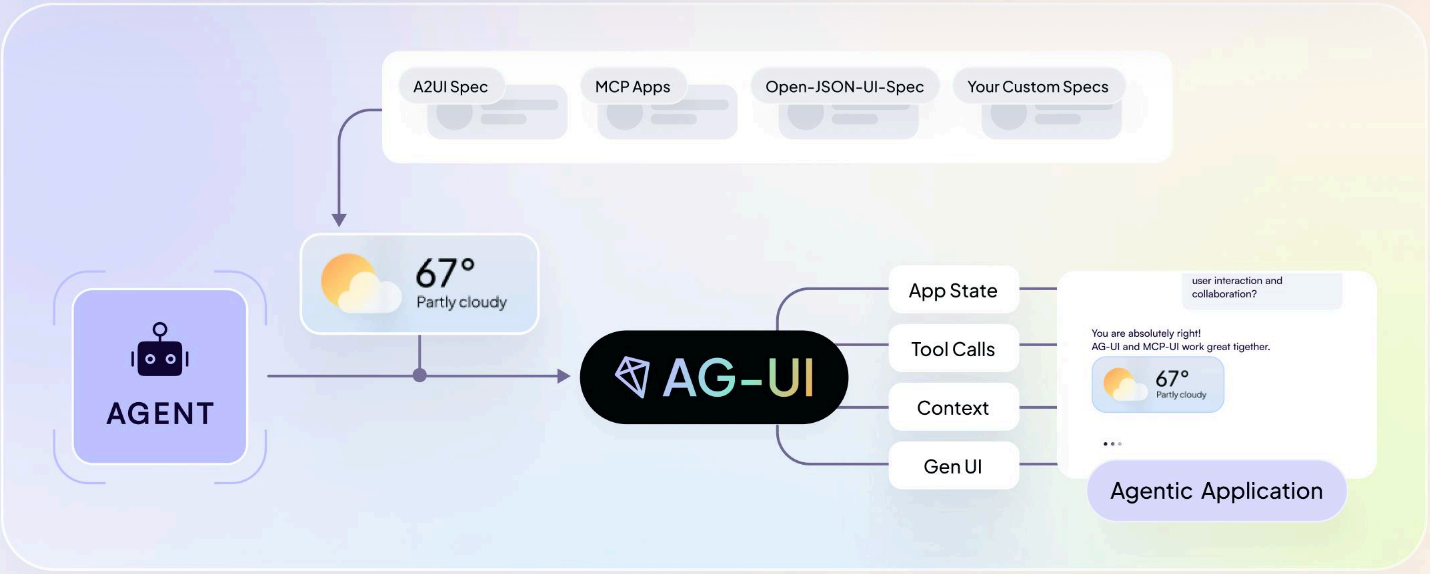
The Agent plays a role in:

- Determining what appears on the screen
- How information is structured
- And in some cases, how the layout is composed

THE CORE IDEA:

As agents become more capable, an agentic application's UI itself becomes more of a dynamic output of the system - able to adapt, reorganize, and respond to user intent and application context.

Note: This can be done in very different ways, each with its own tradeoffs… (more on next page)



As agents reason and act, the UI becomes a dynamic artifact of the system… not a static screen.

3 Types of Generative UI

Generative UI approaches fall into three broad categories, each with distinct tradeoffs in developer experience, UI freedom, adaptability, and long-term maintainability.

Types of Generative UI

Static

UI is chosen from a fixed set of hand-built components.

Declarative

A structured UI specification (cards, lists, forms, widgets) is used between agent and frontend.

Open-Ended

Arbitrary UI (HTML, iframes, free-form content is passed between agent and frontend.

Any of the types can be controlled by the app programmer, or left up to the agent to define.

Some Generative UI Specifications have added richness (and confusion) to generative UIs… such as **MCP Apps**, **Open JSON UI**, and **A2UI**.

Check out this generative UI **interactive 'playground'** to see the differences!

Ecosystem Mapping

No single approach is superior. The best choice depends on your apps priorities, surfaces, and UX philosophy.

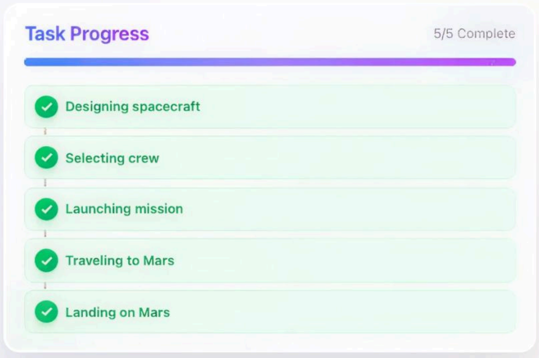
Approach	Examples	Strengths	Weaknesses
Static	AG-UI, CopilotChat, useAgent	Fidelity, reliability, brand control	Engineering intensive, linear growth
Declarative	Open-JSON-UI, A2UI	Balanced, scalable, multi-renderer	Limited full customization
Open-Ended	MCP Apps	Unlimited creativity	Hard to secure, web-first

Application Surfaces

Chat (Threaded Interaction)

Slack-like conversational interface where the app brokers each turn. Generative UI appears inline as cards, blocks, or tool responses.

Examples: Slack bots, Intercom AI Agent, GitHub Copilot Chat, Notion AI Chat.



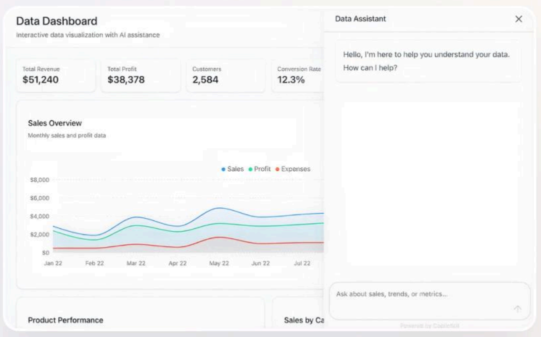
Task Progress 5/5 Complete

- ✓ Designing spacecraft
- ✓ Selecting crew
- ✓ Launching mission
- ✓ Traveling to Mars
- ✓ Landing on Mars

Chat+ (Co-Creator Workspace)

A side-by-side or multi-pane layout: chat in one pane, a dynamic canvas in another. The canvas becomes a shared working space where agent-generated UI appears and evolves.

Examples: Figma AI, Notion AI workspace, Google Workspace Duet side-panel



Data Dashboard

Interactive data visualization with AI assistance

Total Revenue	Total Profit	Customers	Conversion Rate
$51,240	$38,378	2,584	12.3%

Sales Overview

Monthly sales and profit data

Data Assistant

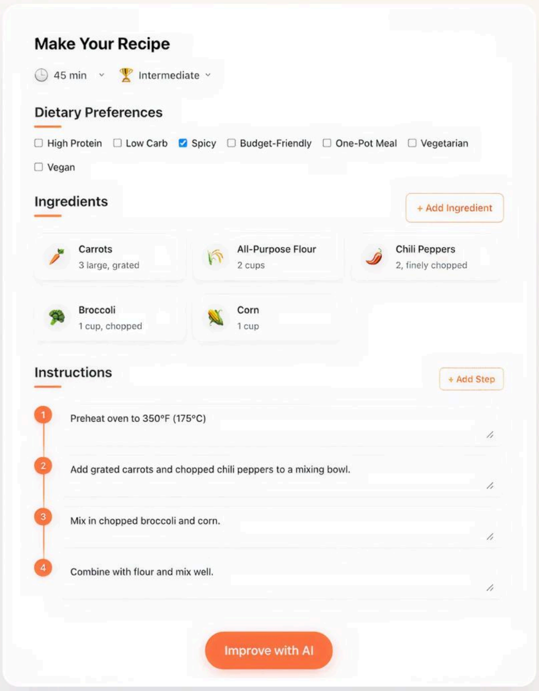
Hello, I'm here to help you understand your data. How can I help?

Ask about sales, trends, or metrics...

Chatless (Generative UI integrated into application UI)

The agent doesn't talk directly to the user. Instead, it communicates with the application through APIs, and the app renders generative UI from the agent as part of its native interface.

Examples: Microsoft 365 Copilot (inline editing), Linear Insights, HubSpot AI Assist



Make Your Recipe

45 min · Intermediate

Dietary Preferences

☐ High Protein ☐ Low Carb ☑ Spicy ☐ Budget-Friendly ☐ One-Pot Meal ☐ Vegetarian ☐ Vegan

Ingredients

Carrots	3 large, grated
All-Purpose Flour	2 cups
Chili Peppers	2, finely chopped
Broccoli	1 cup, chopped
Corn	1 cup

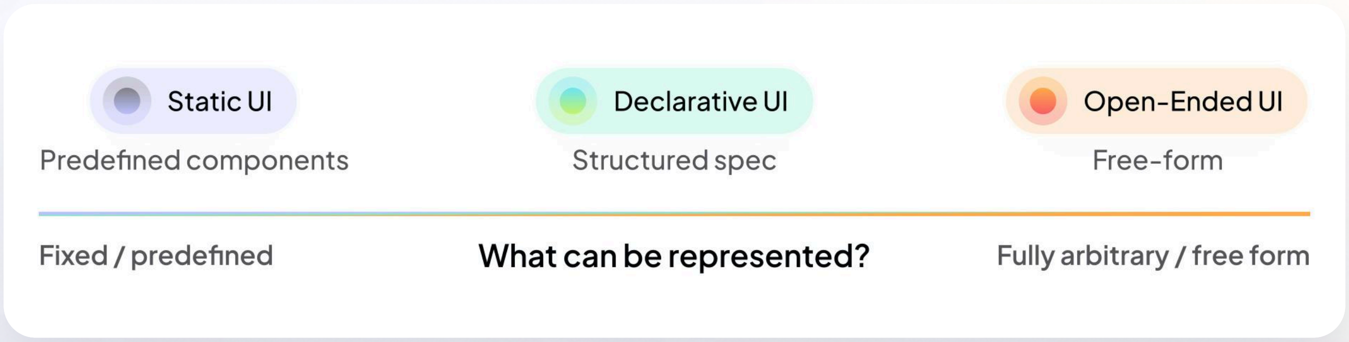
Instructions

1. Preheat oven to 350°F (175°C)
2. Add grated carrots and chopped chili peppers to a mixing bowl.
3. Mix in chopped broccoli and corn.
4. Combine with flour and mix well.

Attributes of Generative UI

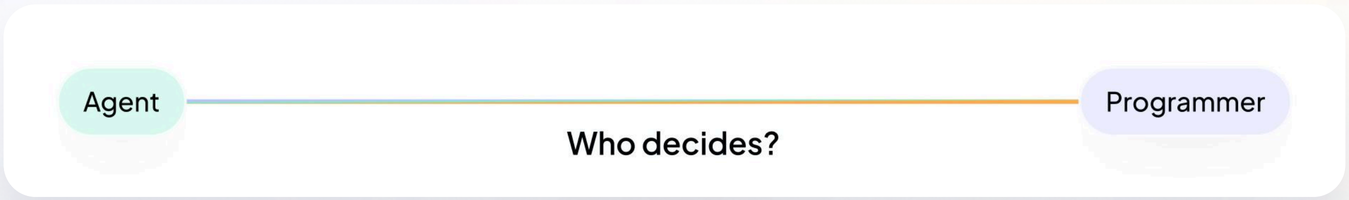
Freedom of Expression

Generative UI spans a spectrum of expressive freedom: fixed components → declarative UI → fully open-ended HTML



Who has Control

Are UI decisions made by the agent or constrained by the developer?



Static Generative UI

Static generative UI allows engineers to hand-craft specific visual components, and agents simply decide which of those components to use. The agent does not generate arbitrary UI; instead, it maps generated data to existing UI components.

In this model, the front end defines every detail of the experience- the layouts, styles, interaction patterns, and constraints. The backend or agent contributes information and intent, but the rendering ultimately comes from a predefined set of components.

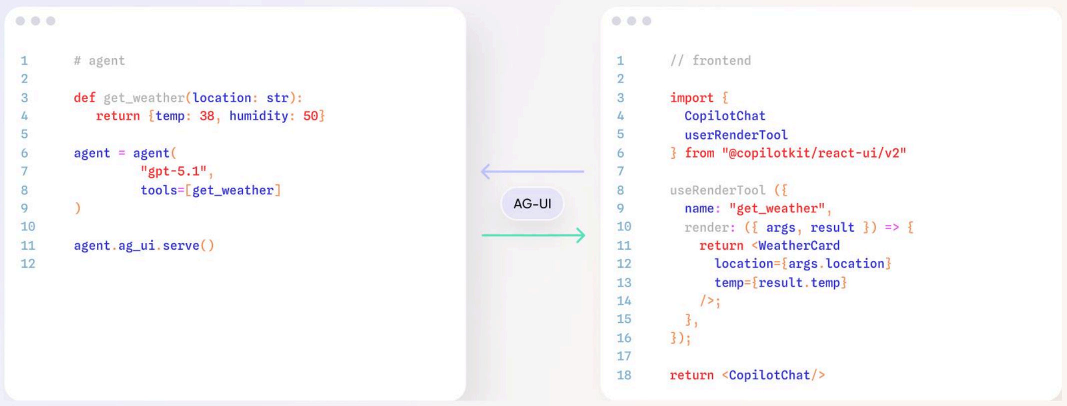
Why teams use it:

- ✓ Guarantees high visual polish and consistency
- ✓ Ideal for high-traffic, mission-critical surfaces where predictability matters

Tradeoffs:

- ⚠ The frontend codebase grows proportionally to the number of agent capabilities
- ⚠ The more use cases, the more components you must build and maintain

Example: Using CopilotChat with AG-UI



```python
# agent

def get_weather(location: str):
   return {temp: 38, humidity: 50}

agent = agent(
         "gpt-5.1",
         tools=[get_weather]
)

agent.ag_ui.serve()
```

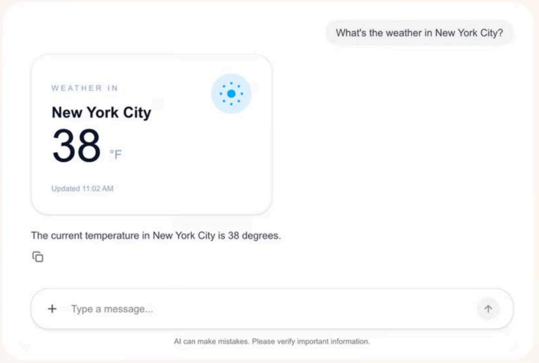
```jsx
// frontend

import {
  CopilotChat
  userRenderTool
} from "@copilotkit/react-ui/v2"

useRenderTool ({
  name: "get_weather",
  render: ({ args, result }) => {
    return <WeatherCard
      location={args.location}
      temp={result.temp}
    />;
  },
});

return <CopilotChat/>
```

The agent defines tools, AG-UI handles the connection, and the frontend renders predefined components.


What's the weather in New York City?
WEATHER IN
New York City
38 °F
Updated 11:02 AM
Type a message...
AI can make mistakes. Please verify important information.

The current temperature in New York City is 38 degrees.

Declarative Generative UI

Declarative generative UI balances structure and flexibility by having agents return a structured specification rather than arbitrary UI code. Instead of free-form HTML, agents emit a well-defined schema — such as a collection of cards, lists, forms, or widgets defined by a declarative standard.

This approach preserves consistency while giving agents far greater expressive power than purely static component libraries. It creates a middle ground where UI is not handcrafted for each use case, but is also not fully free-form.

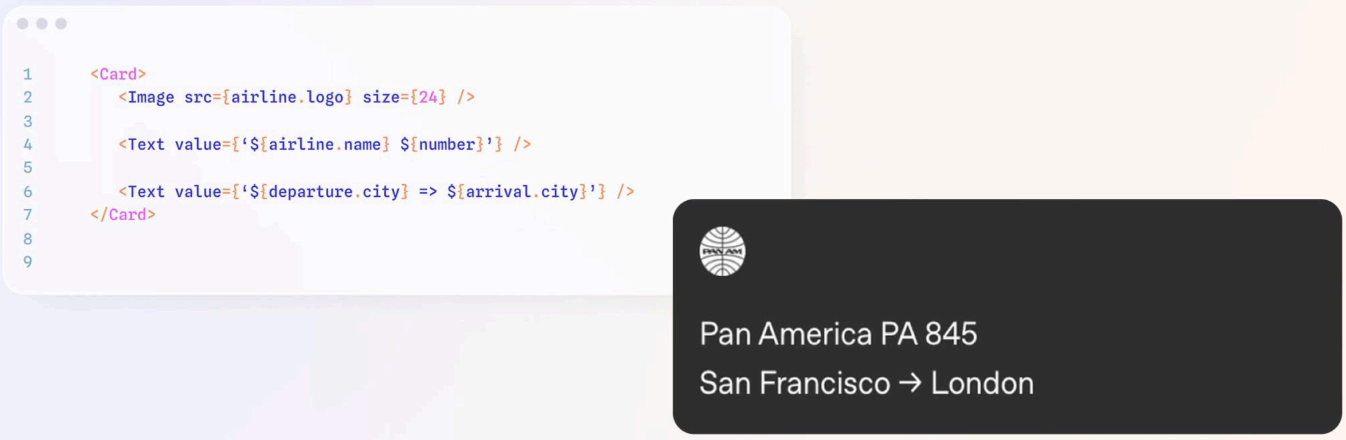
Why teams use it:

- ✓ Supports a wide range of use cases without requiring custom components for each
- ✓ Developers can render the same spec across multiple frameworks (React, mobile, desktop...)

Tradeoffs:

- ⚠ Custom UI patterns may not be possible
- ⚠ Visual differences can still occur if specs are interpreted differently

Example: Using Open-JSON-UI



```
<Card>
   <Image src={airline.logo} size={24} />

   <Text value={'${airline.name} ${number}'} />

   <Text value={'${departure.city} => ${arrival.city}'} />
</Card>
```

Pan America PA 845
San Francisco → London

The agent returns a structured spec (like a Card) that the frontend interprets and renders consistently.

Open-Ended Generative UI

Open-ended generative UI represents the opposite end of the spectrum. Here, agents generate complete UI surfaces — often as HTML, iframes, or free-form markup. Instead of choosing from predefined components, the agent can respond with an entire UI payload that the frontend simply displays.

This approach provides unparalleled flexibility of expression. Agents can render a calendar, a custom table, an animated visualization, or an interactive HTML widget, either generated by the LLM, or predefined by an application developer.

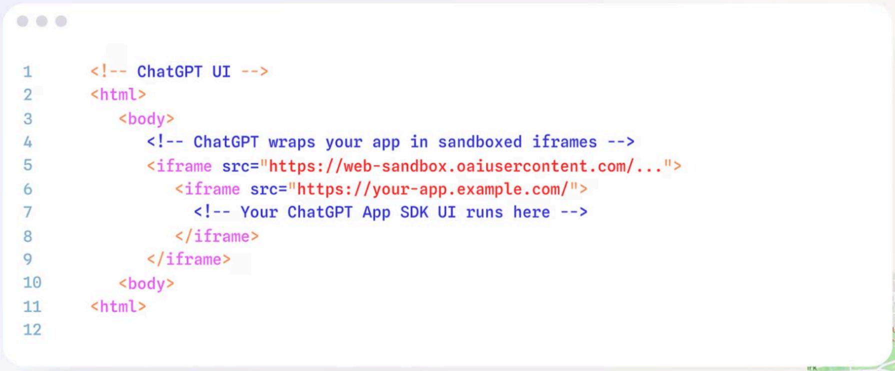
Why teams use it:

- ✓ Any type of UI can be part of an agent response, whether predefined by the programmer or generated by the agent
- ✓ Minimal coupling between frontend code and agent behavior
- ✓ Supports rapid prototyping and complex workflows without frontend engineering cycles

Tradeoffs:

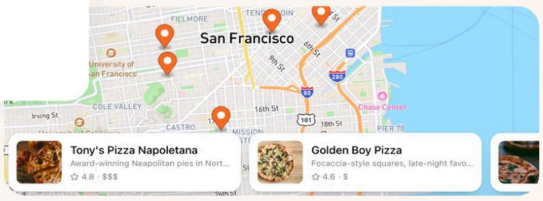
- ⚠ Security and performance considerations when rendering arbitrary content
- ⚠ Typically web-first and difficult to port to native environments
- ⚠ Styling consistency and brand alignment become challenging

Example: Using MCP-UI & ChatGPT Apps SDK



```
<!-- ChatGPT UI -->
<html>
  <body>
    <!-- ChatGPT wraps your app in sandboxed iframes -->
    <iframe src="https://web-sandbox.oaiusercontent.com/...">
      <iframe src="https://your-app.example.com/">
        <!-- Your ChatGPT App SDK UI runs here -->
      </iframe>
    </iframe>
  <body>
<html>

```


San Francisco
Tony's Pizza Napoletana
Award-winning Neapolitan pies in Nort...
4.8 · $$$
Golden Boy Pizza
Focaccia-style squares, late-night favo...
4.6 · $

The agent returns complete HTML/iframes that render arbitrary UI, giving maximum flexibility.

Why Does Generative UI Matter?

1 A new application paradigm

Once UI shifts, backend architecture, permissions, and metrics must shift too. Agentic UI reshapes how software is designed end-to-end.

2 Usefulness vs. impressiveness

Strong models don't matter if people can't operate or trust them. UI is now the main limiter of whether AI delivers real value or stays a demo.

3 Automation vs. delegation (agent-user collaboration is most efficient)

Automation still requires users to drive step-by-step. Delegation only works when UI lets users set goals, monitor plans, and intervene safely.

CopilotKit as the Central Enabler and How to Build with Generative UI

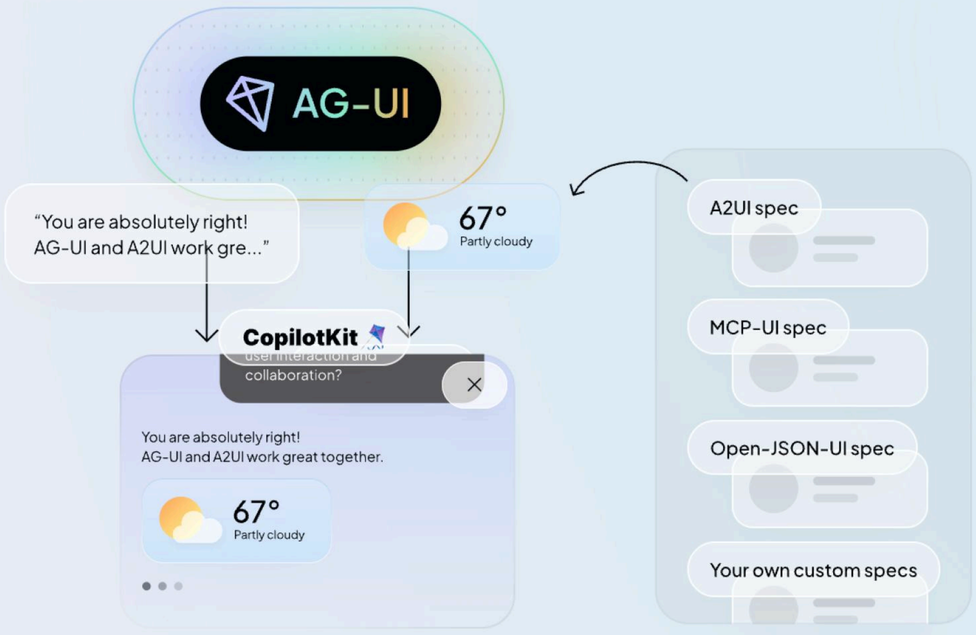
The AG-UI protocol is designed to support the full spectrum of generative UI techniques while adding important capabilities that unify them.

AG-UI integrates seamlessly with all types: static, declarative, and open-ended generative UI approaches.

But AG-UI adds shared primitives — interaction models, context synchronization, event handling, a common state framework — that standardize how agents and UIs communicate across all surface types.

CopilotKit works with any generative UI, and uses AG-UI to connect the agent to the frontend.

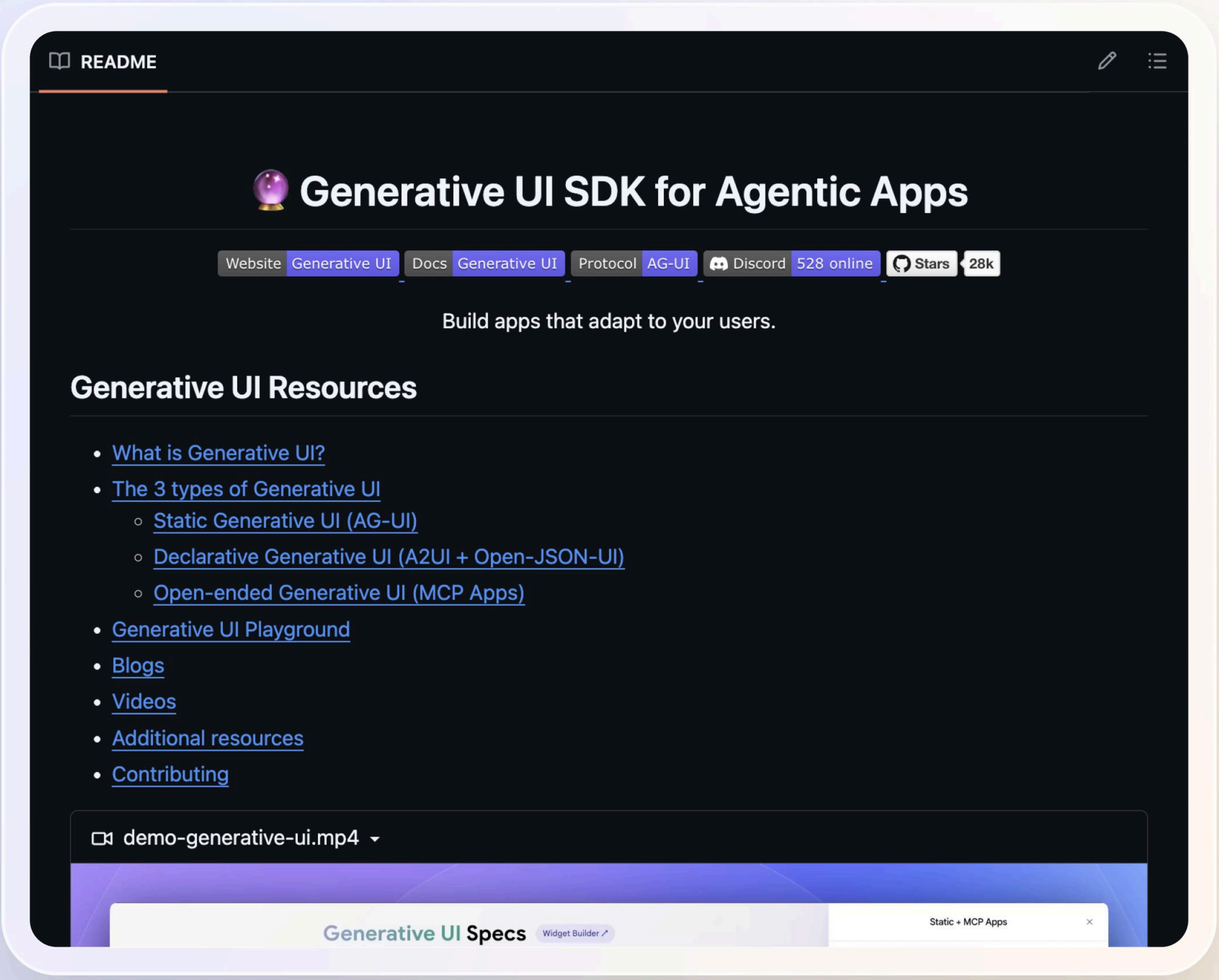
→ **CopilotKit** is the home for building agentic apps with ***generative UI*** that mix and match all these elements! Plug and play any agent framework, protocol, spec and more within minutes.


AG-UI
"You are absolutely right! AG-UI and A2UI work gre..."
67° Partly cloudy
CopilotKit
user interaction and collaboration?
You are absolutely right! AG-UI and A2UI work great together.
67° Partly cloudy
A2UI spec
MCP-UI spec
Open-JSON-UI spec
Your own custom specs

AG-UI acts as a universal runtime that works with A2UI, MCP-UI, Open-JSON-UI, and custom specs of any type.

How to Get Started

Check out this newly released educational repo on Generative UI, updated for 2026


README

🔮 Generative UI SDK for Agentic Apps

Website Generative UI | Docs Generative UI | Protocol AG-UI | Discord 528 online | Stars 28k

Build apps that adapt to your users.

Generative UI Resources

- What is Generative UI?
- The 3 types of Generative UI
  - Static Generative UI (AG-UI)
  - Declarative Generative UI (A2UI + Open-JSON-UI)
  - Open-ended Generative UI (MCP Apps)
- Generative UI Playground
- Blogs
- Videos
- Additional resources
- Contributing

demo-generative-ui.mp4

This repo is a great place to learn — including tons of different resources such as demos, tutorials, videos, code snippets, and more.

https://go.copilotkit.ai/generative-ui-repo

CopilotKit

Found this useful?

Repost It

to help one person in your network