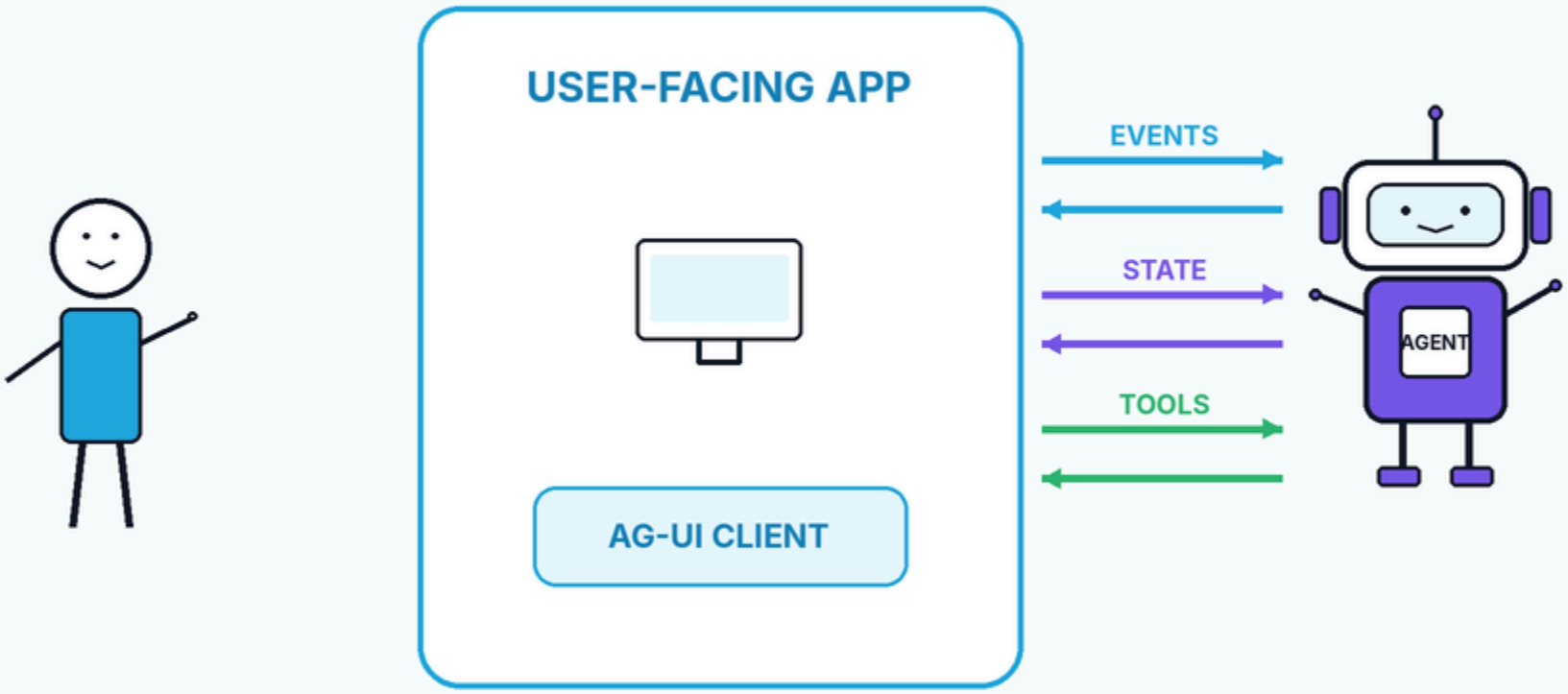
HOW DOES A FRONTEND KEEP UP WITH AN AGENT WHILE IT WORKS?

Meet AG-UI — the real-time bridge between user-facing apps and agent backends.



A2UI can describe an interface. AG-UI keeps the frontend and agent in a live conversation.

THE OLD WEB MODEL: ONE REQUEST → ONE RESPONSE

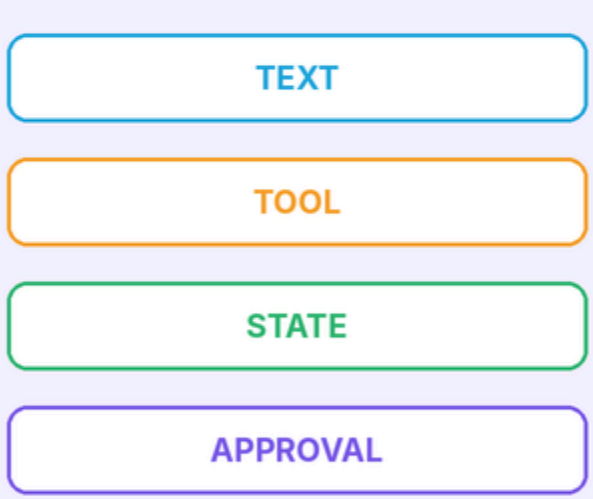
Agentic apps often need a continuous stream instead.

TRADITIONAL API

Request → server work → one final response.

AGENTIC INTERACTION

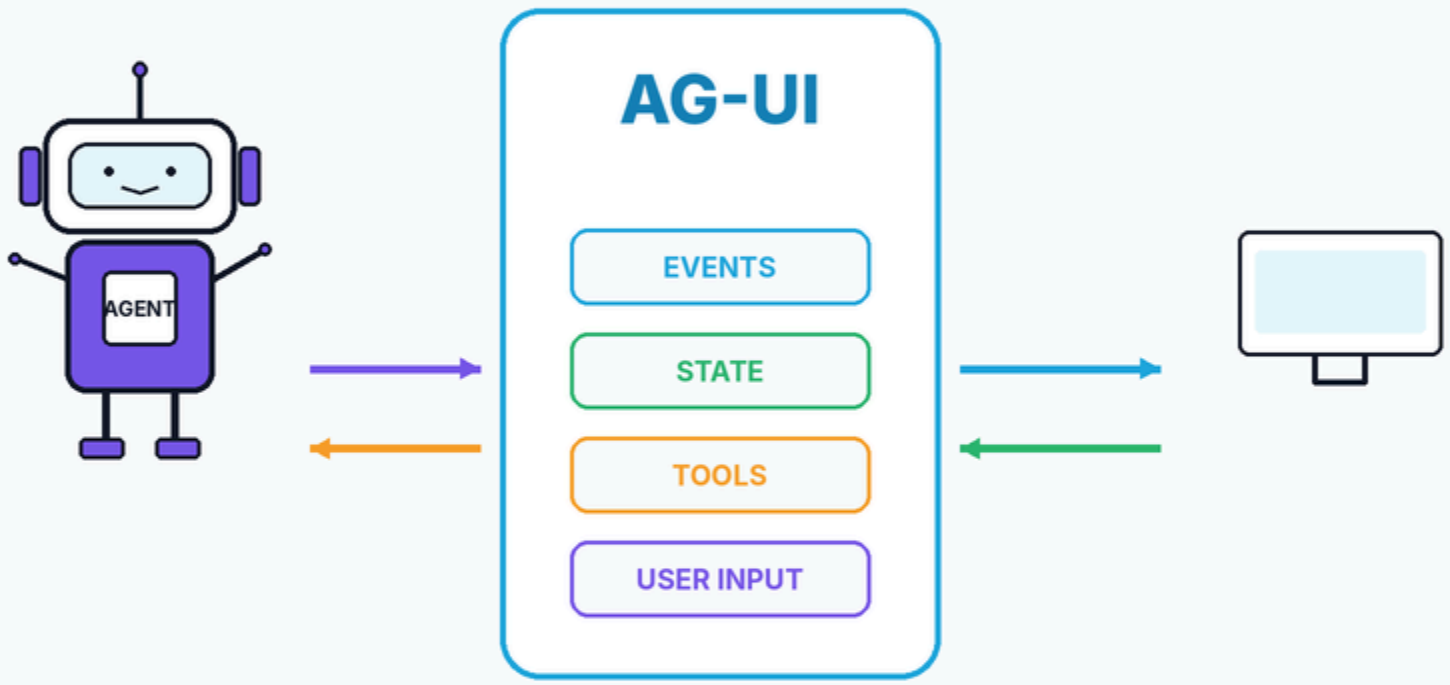
Text streams, tools run, state changes, users approve, and work continues.



Agents behave more like live processes than ordinary request/response endpoints.

AG-UI = AGENT–USER INTERACTION PROTOCOL

An open, lightweight, event-based protocol for connecting agents to user-facing applications.



AG-UI standardizes the interaction layer — not the agent's model, planner or business logic.

A2UI VS AG-UI: SIMILAR NAME, DIFFERENT JOB

They are complementary — and can be used together.

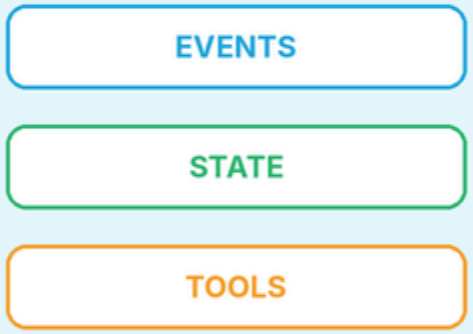
A2UI

A generative UI specification. It describes what components the agent wants the app to render.

WHAT UI TO SHOW

AG-UI

A bi-directional runtime protocol. It carries messages, events, state, tools and user interactions.

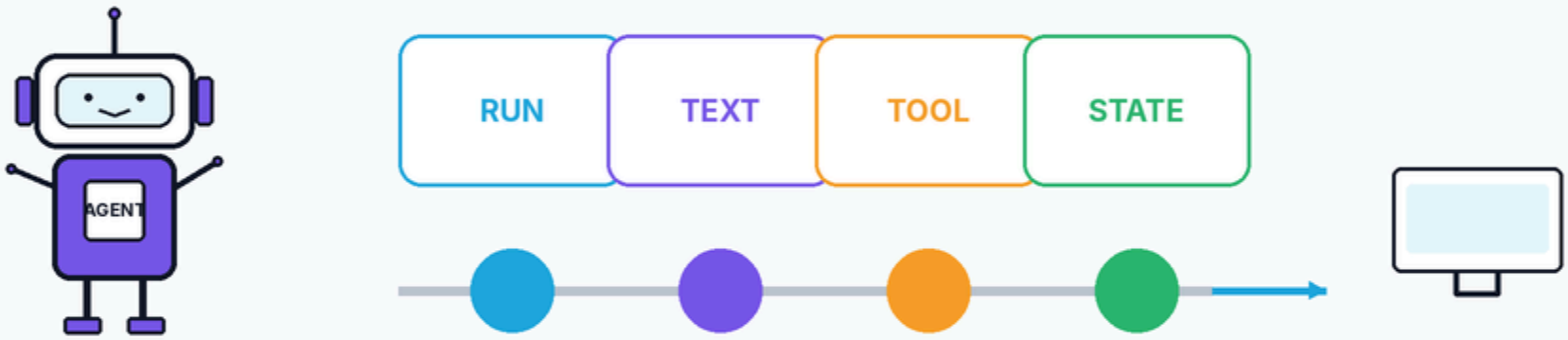


HOW APP ↔ AGENT TALK

A2UI can ride over AG-UI: one describes UI; the other carries the live interaction.

THINK OF AG-UI AS A LIVE EVENT FEED

The agent emits typed events. The frontend reacts as each one arrives.



Frontend updates immediately
instead of waiting for “the final answer.”

Events turn one long agent run into small, understandable updates.

LIFECYCLE EVENTS: WHERE IS THE AGENT NOW?

The frontend can track the run without guessing.

1 RUN_STARTED

A run begins

2 STEP_STARTED

A unit of work begins

3 STEP_FINISHED

That unit completes

4 RUN_FINISHED

The run completes

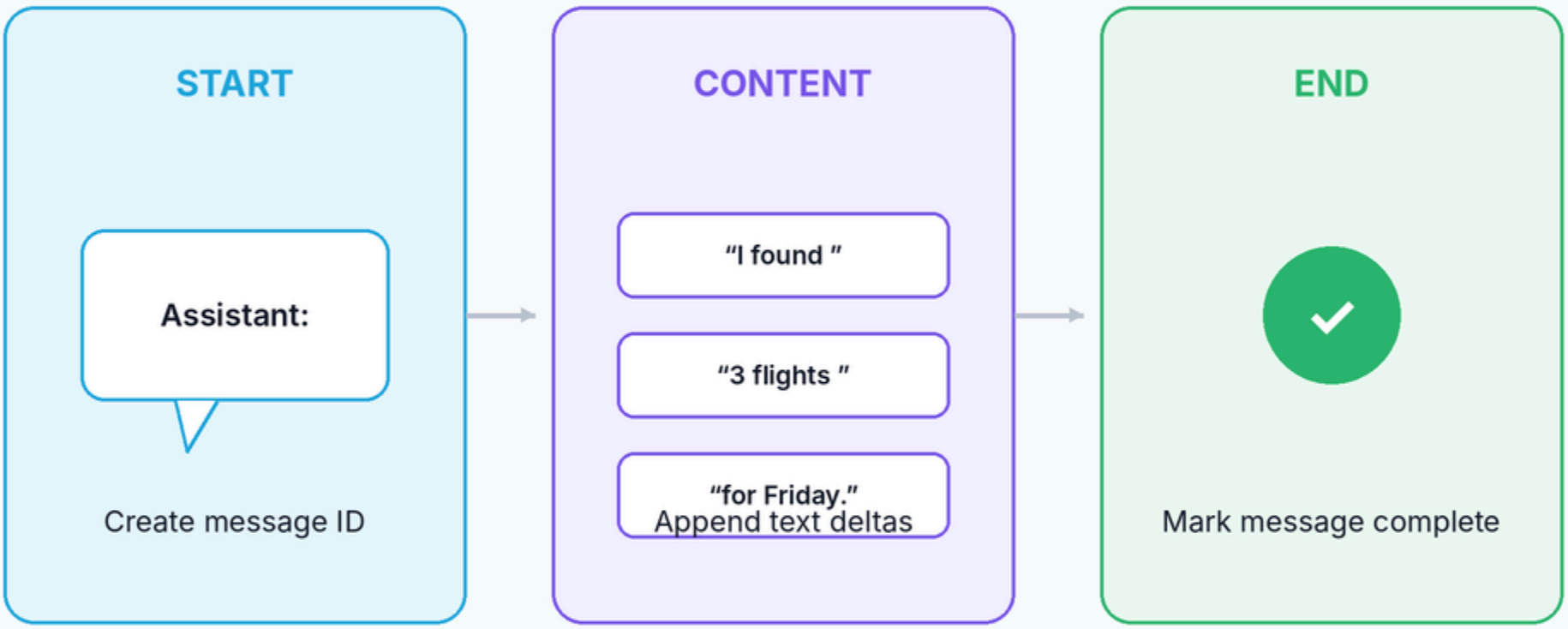
! RUN_ERROR

Something failed

Lifecycle events make progress, completion and failure visible to the app.

TEXT CAN STREAM TOKEN BY TOKEN

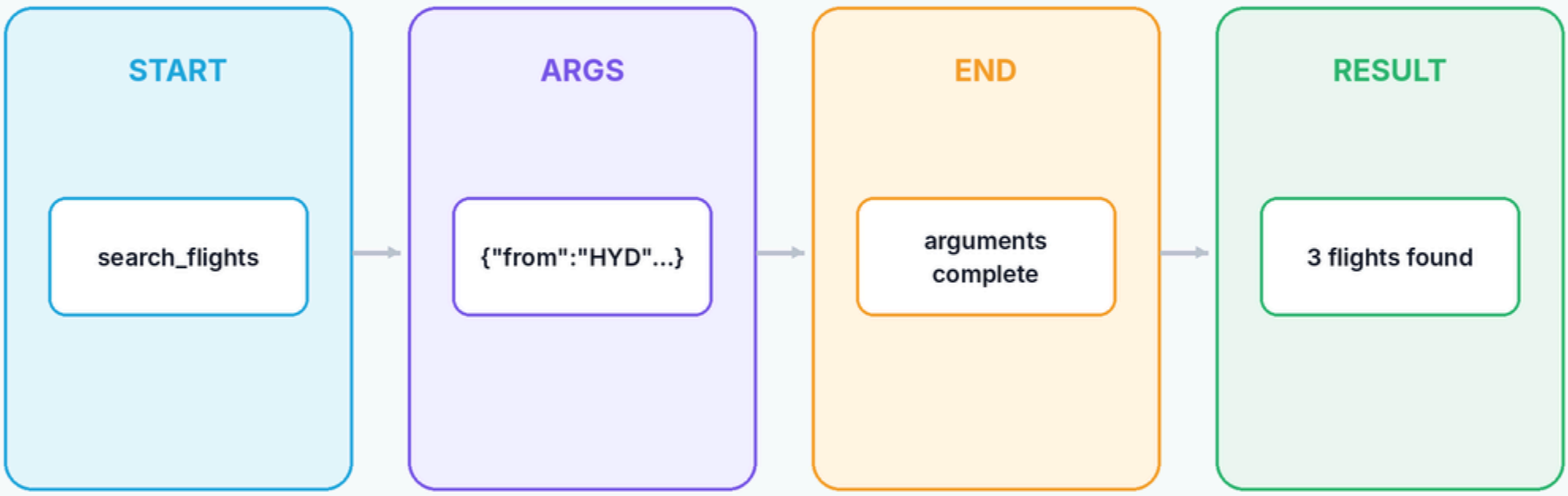
AG-UI uses a start → content → end pattern for messages.



Streaming makes the app feel responsive even while the agent is still generating.

TOOL CALLS ARE EVENTS TOO

The UI can show what the agent is trying to do — and what came back.



The frontend can render “Searching flights...” before the tool has even finished.

Tool visibility makes agent actions easier to understand, inspect and approve.

SHARED STATE: SNAPSHOT + DELTA

The frontend and agent can stay synchronized without resending everything.

STATE_SNAPSHOT

A complete state object. Use it to establish or reset the known baseline.

Trip
city: Tokyo
budget: ₹1.5L
flight: none

STATE_DELTA

Only the changes. AG-UI uses JSON Patch operations for efficient updates.

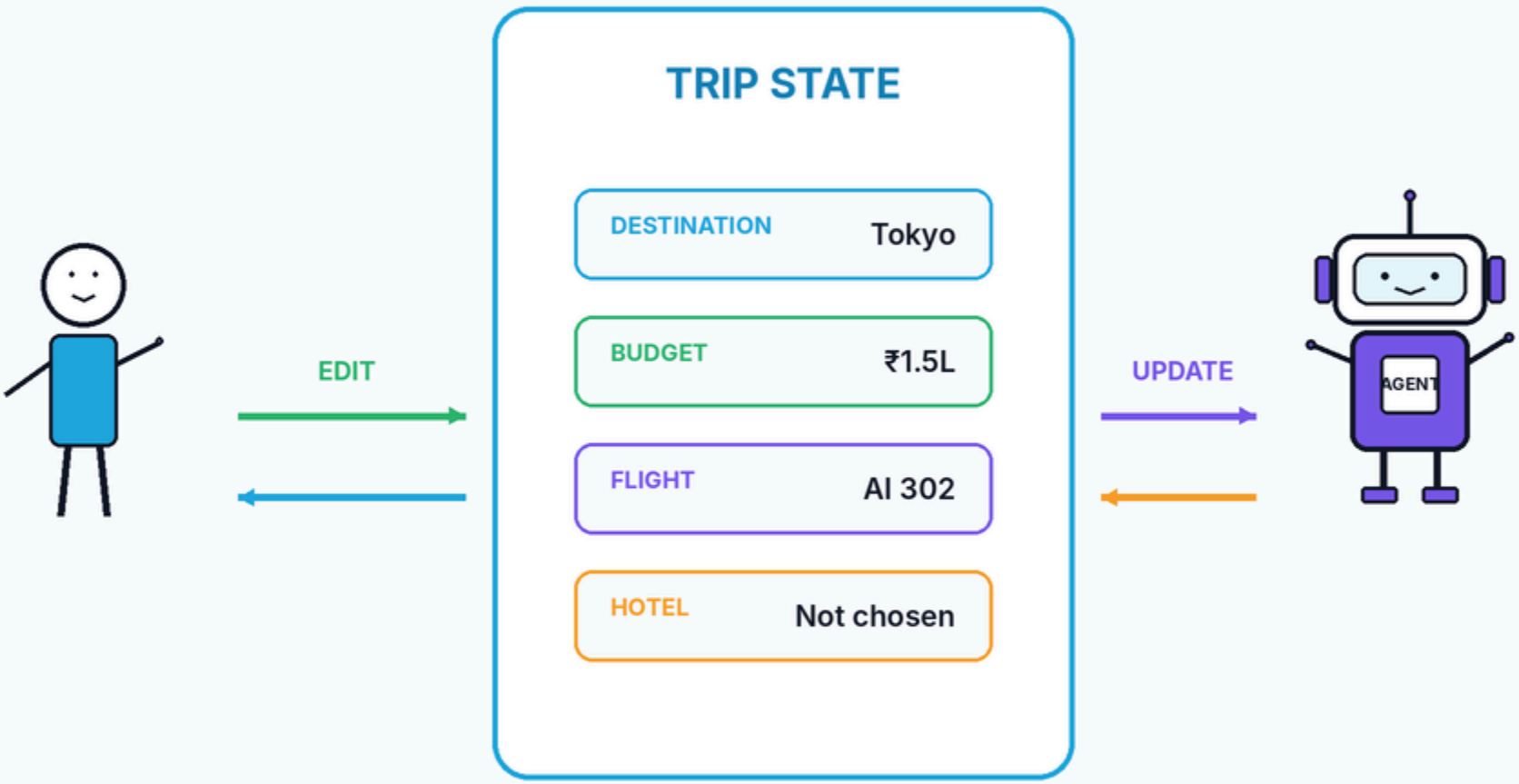
PATCH

replace /flight
→ “AI 302”

Snapshots give completeness. Deltas make frequent updates cheap.

WHY SHARED STATE MATTERS

The agent and the UI can collaborate on the same task state.



The UI can change the state. The agent can react to the same updated state in the next step.

THE FRONTEND CAN EXPOSE TOOLS

AG-UI can pass client-defined tools to the agent — including human confirmation.

FRONTEND TOOLS

CONFIRM BOOKING

Ask the person before purchase

GET SELECTION

Read what the user picked

OPEN REVIEW

Show a review panel

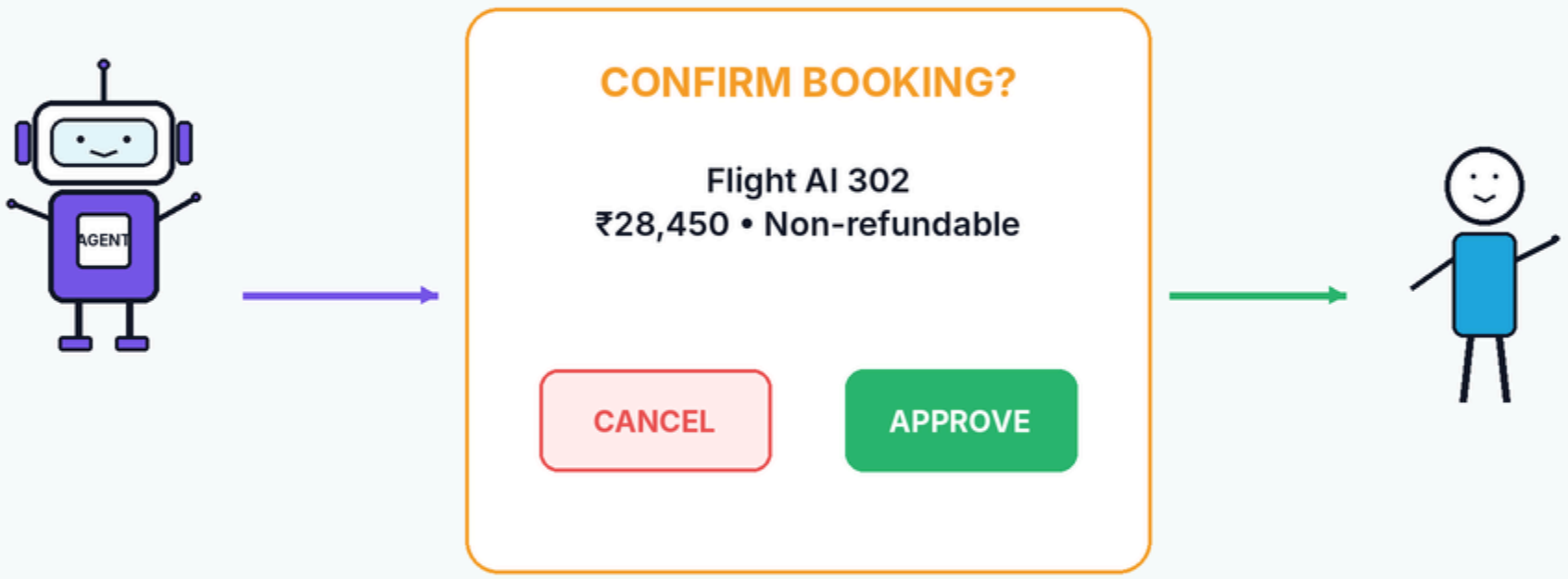
REQUEST INPUT

Collect missing information

Some “tools” belong in the user’s app — because the person is part of the workflow.

HUMAN-IN-THE-LOOP: PAUSE BEFORE THE CONSEQUENCE

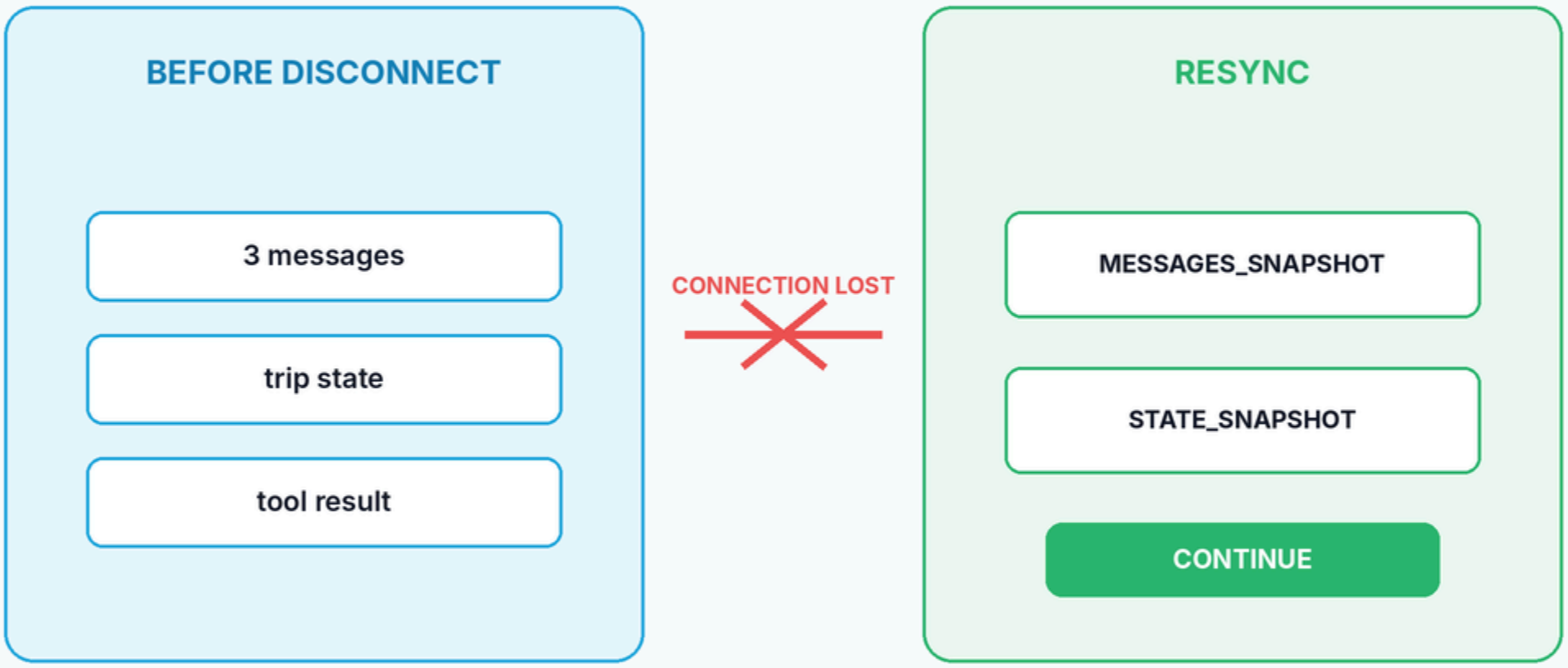
A frontend tool can turn a risky action into an explicit user decision.



The protocol carries the interaction; your product still defines which actions require approval.

RECONNECT WITHOUT LOSING THE THREAD

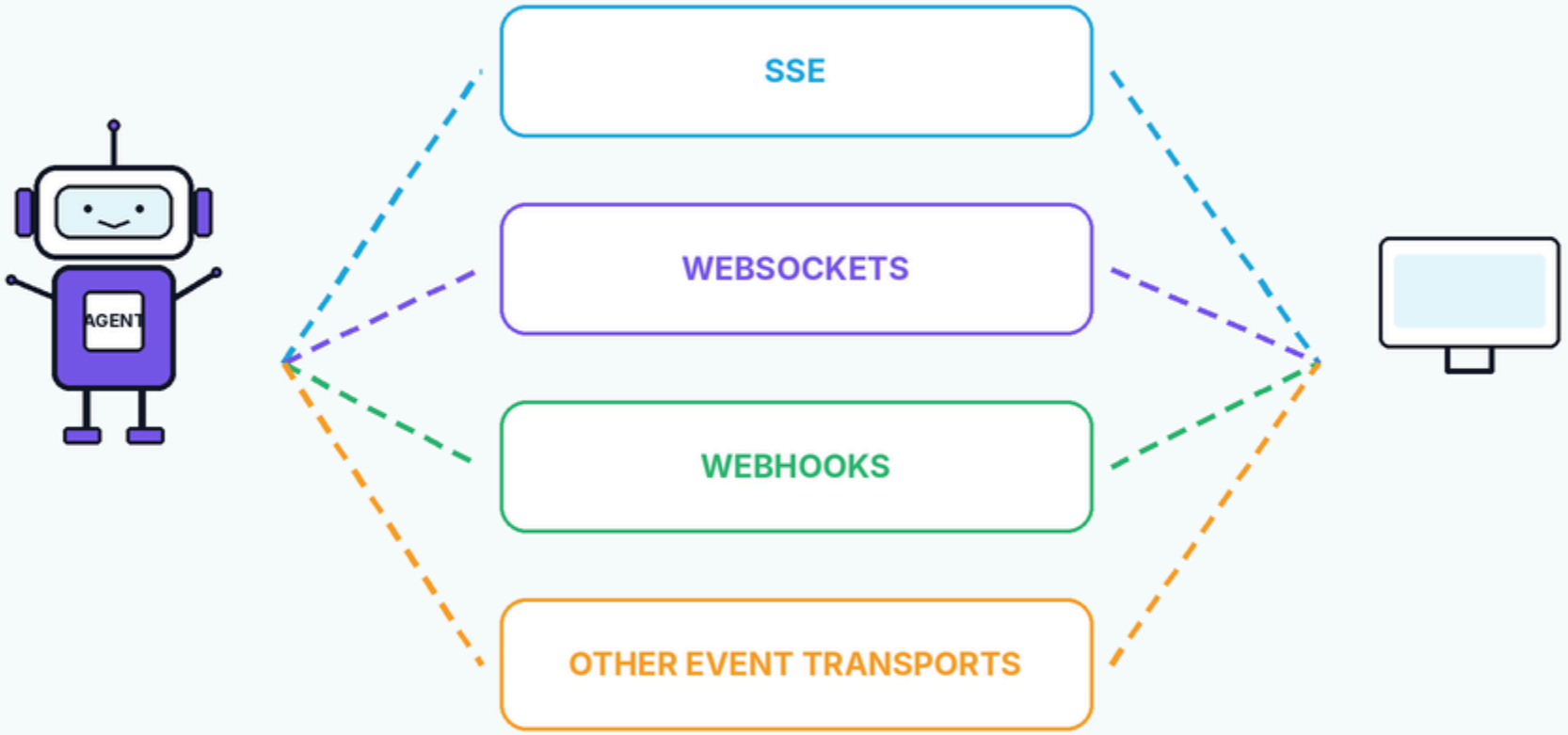
Snapshots let the client rebuild what it needs after an interruption.



A fresh snapshot gives the frontend a known baseline before streaming new deltas again.

AG-UI DOES NOT FORCE ONE TRANSPORT

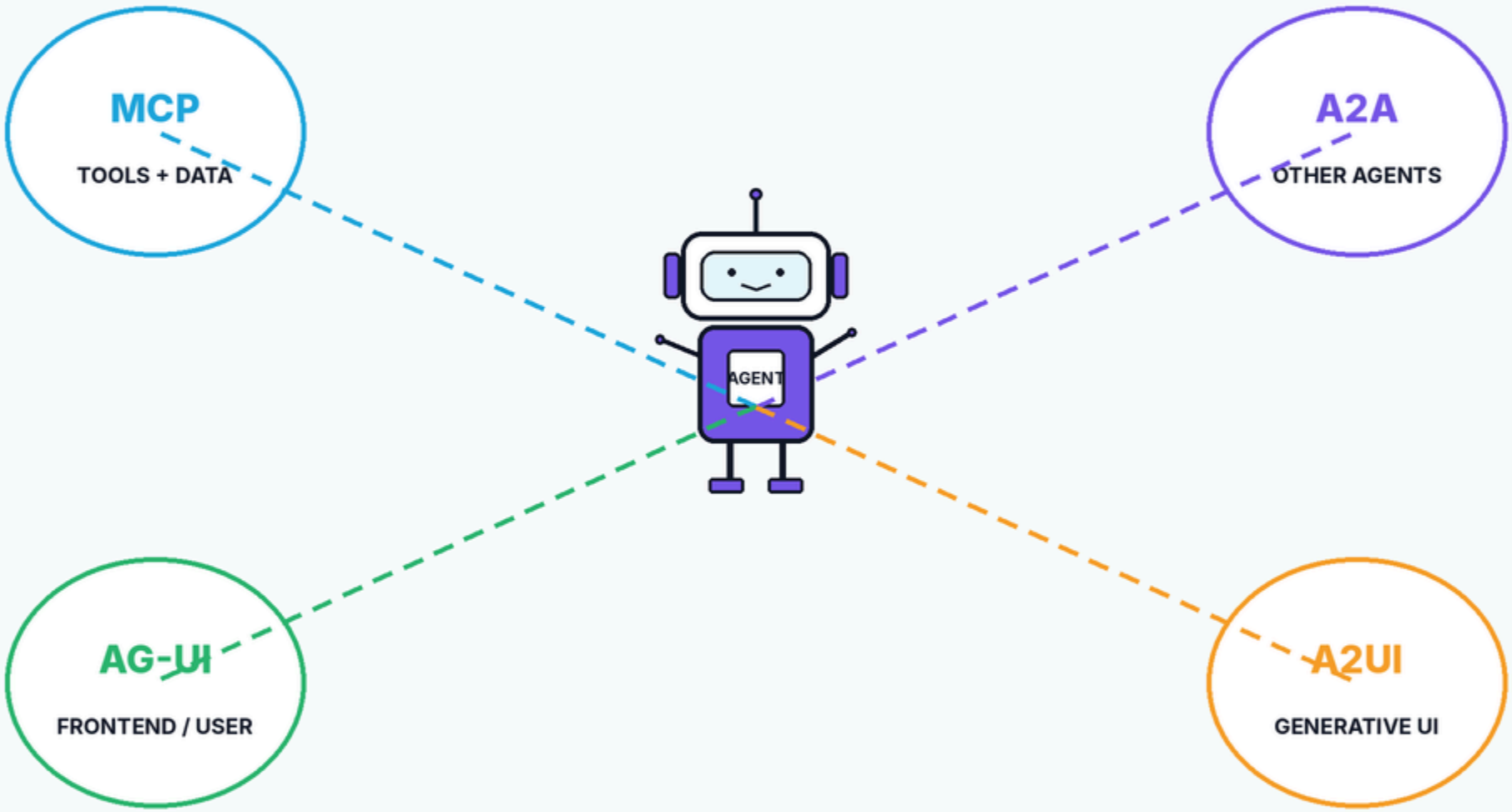
The protocol is event-based and can work over different delivery mechanisms.



AG-UI defines the interaction semantics; your architecture chooses how the events travel.

THE AGENTIC PROTOCOL STACK

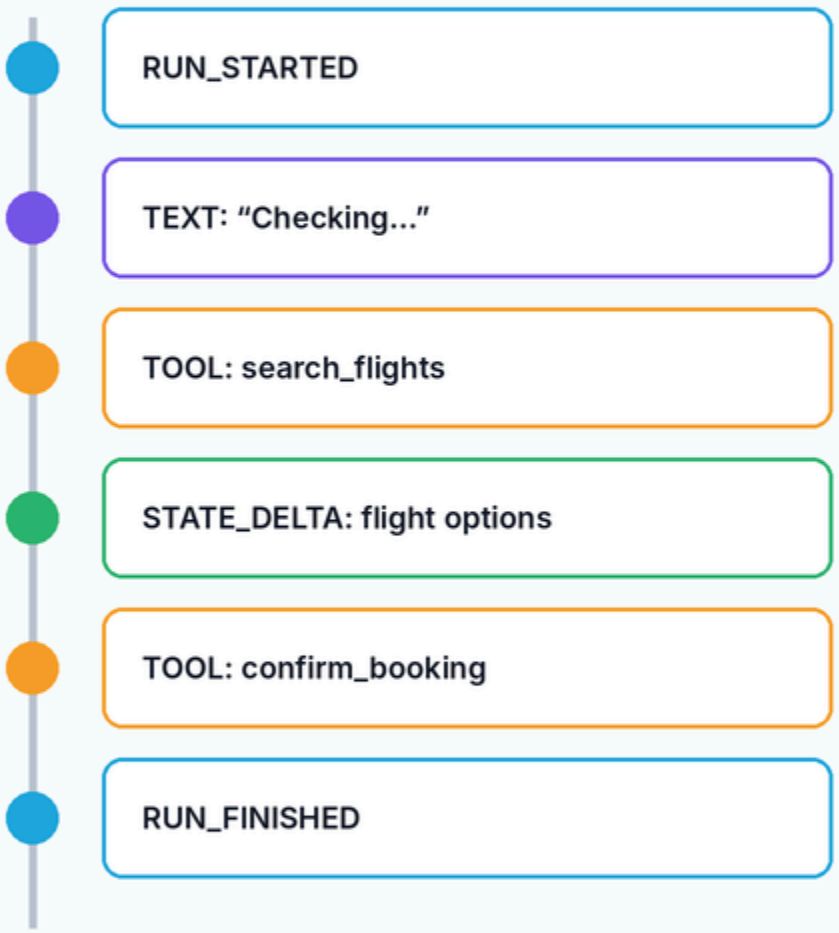
Different protocols solve different connections — and can be combined.

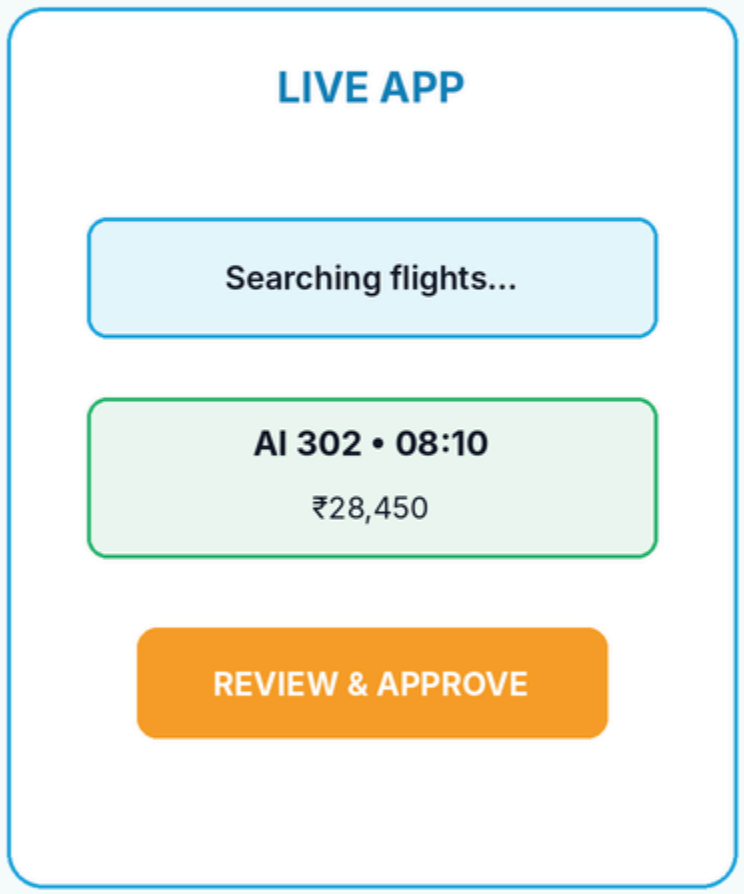


MCP gives capabilities. A2A coordinates agents. AG-UI connects the app. A2UI can describe UI.

REAL EXAMPLE: PLAN MY CONFERENCE TRIP

One AG-UI run can produce many kinds of updates.





The frontend does not receive one blob. It receives a sequence it can render as work happens.

AG-UI HAS A GROWING FRAMEWORK ECOSYSTEM

The protocol already connects to many popular agent runtimes and platforms.

LANGGRAPH

MS AGENT FRAMEWORK

GOOGLE ADK

AWS STRANDS

MASTRA

PYDANTIC AI

AGNO

LLAMAINDEX

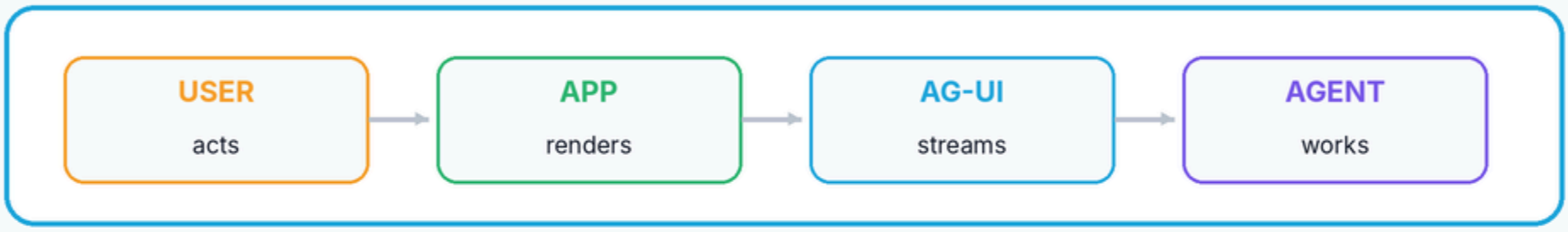
CREWAI / AG2

PLUS: A2A MIDDLEWARE • BEDROCK AGENTCORE • A2UI / MCP APPS

The goal is interoperability: one user-facing app can connect to different agent backends.

DAY 19 TAKEAWAY

AG-UI is the live communication layer between an agent and the application people use.



EVENTS

Stream progress

STATE

Stay synchronized

TOOLS

Expose actions

HUMAN

Approve / edit

AG-UI IS NOT THE AGENT BRAIN — IT IS THE LIVE BRIDGE TO THE PRODUCT.

NEXT: DAY 20 — Agent Harness: the runtime around the model that keeps the whole agent loop working.

Events make agents observable. Harnesses make them operable.