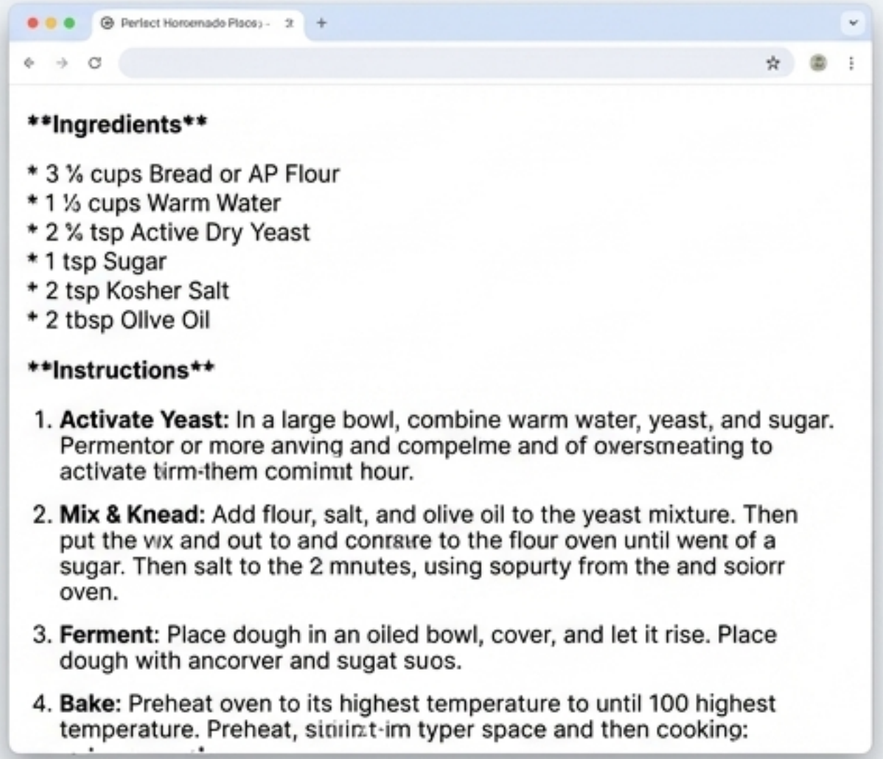
From a Wall of Text to a World of Interaction

Before: Standard LLM Output



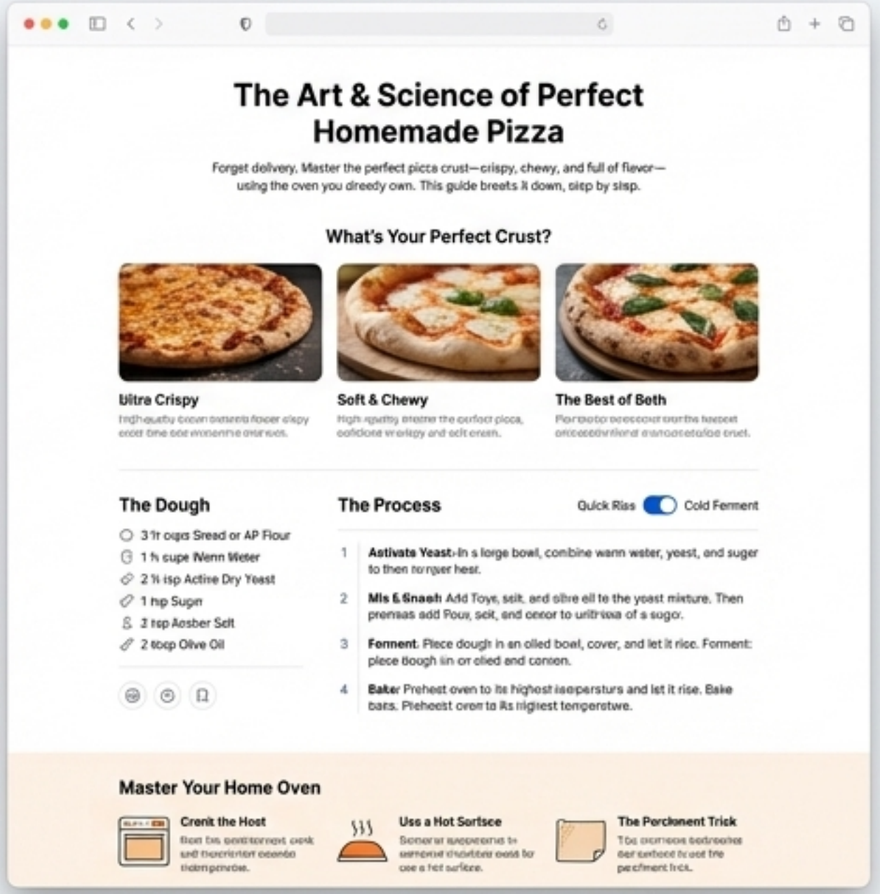
Ingredients

* 3 ⅞ cups Bread or AP Flour
* 1 ⅛ cups Warm Water
* 2 ¾ tsp Active Dry Yeast
* 1 tsp Sugar
* 2 tsp Kosher Salt
* 2 tbsp Olive Oil

Instructions

1. **Activate Yeast:** In a large bowl, combine warm water, yeast, and sugar. Permentor or more anving and compelme and of oversmeating to activate tirm-them comimut hour.
2. **Mix & Knead:** Add flour, salt, and olive oil to the yeast mixture. Then put the vix and out to and conrsure to the flour oven until went of a sugar. Then salt to the 2 mnutes, using sopurty from the and soiorr oven.
3. **Ferment:** Place dough in an oiled bowl, cover, and let it rise. Place dough with ancorver and sugat suos.
4. **Bake:** Preheat oven to its highest temperature to until 100 highest temperature. Preheat, sinnizt-im typer space and then cooking:

After: **Generative UI**



The Art & Science of Perfect Homemade Pizza

Forget delivery. Master the perfect pizza crust—crispy, chewy, and full of flavor—using the oven you already own. This guide breaks it down, step by step.

What's Your Perfect Crust?

Ultra Crispy

Soft & Chewy

The Best of Both

The Dough

The Process

Quick Rise

Cold Ferment

Master Your Home Oven

Crank the Heat

Use a Hot Surface

The Parchment Trick

Large Language Models are evolving from content creators to experience builders. This is Generative UI.

Generative UI is a Paradigm of Co-Creation, Not Just Automation

> **Generative UI** is a paradigm where interfaces are not merely designed by humans but are **co-created** with AI systems. These systems generate, refine, and adapt the UI in real-time based on context, intent, and data.

Human-AI Partnership

The goal is not full automation but a "creative dialogue." The designer provides direction, constraints, and evaluation; the AI provides variations and possibilities.

Synthesis Over Selection

Traditional design often involves selecting from predefined components. Generative UI enables the continuous synthesis of new design possibilities through recombination and transformation.

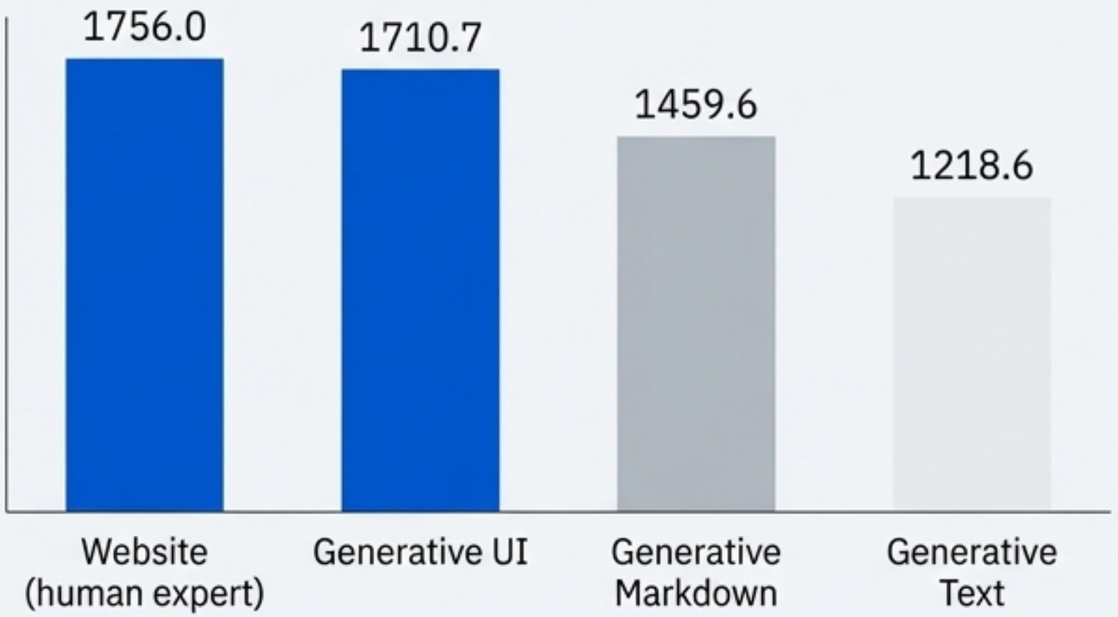
Dynamic & Ephemeral

Instead of a finite library of static applications, Generative UI enables an infinite catalog of ephemeral interfaces, tailored to a specific need at a specific moment.

This is an Emergent Capability of Modern LLMs

The ability to robustly produce high-quality custom UIs is a recent breakthrough. While not yet at the level of human experts, the latest models are preferred overwhelmingly to standard markdown.

User Preference ELO Scores



Pairwise User Preference Wins

Generative UI vs. Markdown	**82.8%** GenUI Preferred	
Generative UI vs. Human Expert	**56.0%** Human Expert Preferred	**44.0%** GenUI Comparable

Source: Yaniv Leviathan et al., 'Generative UI: LLMs are Effective UI Generators', Google Research, 2025.

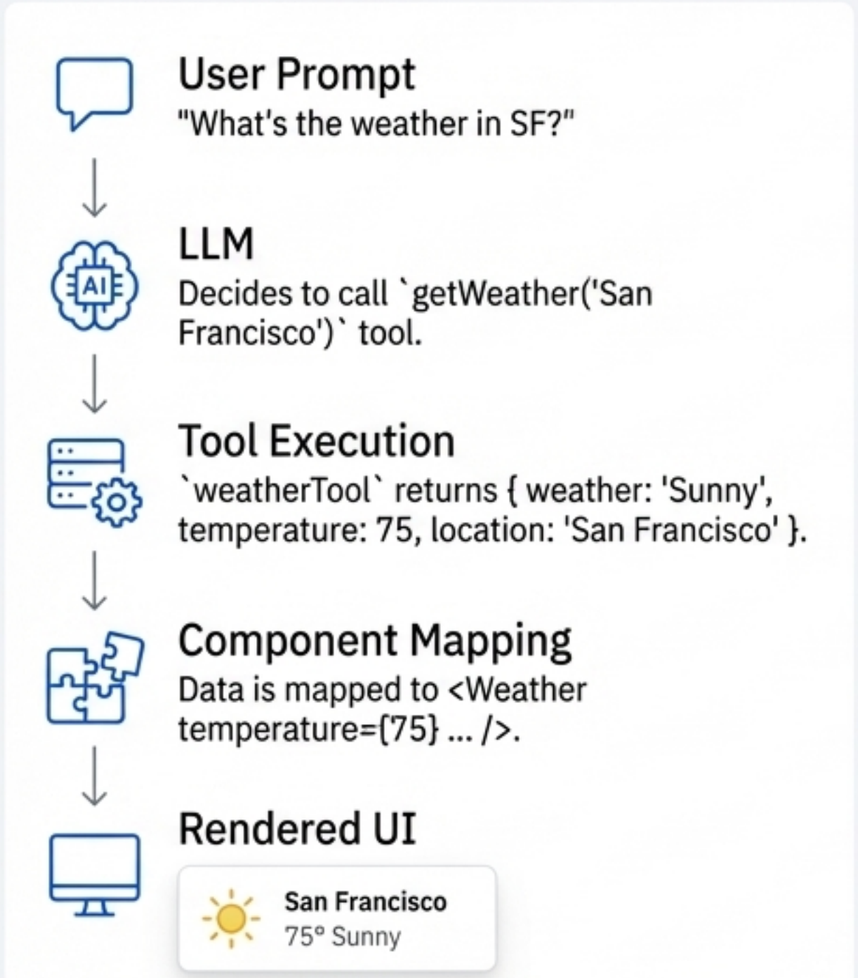
The Evolution from Static Interfaces to Co-Creative Systems

Paradigms in User Interface Design					
	Traditional GUI	**Customizable UI (CUI)**	**Adaptive UI (AUI)**	**Intelligent UI (IUI)**	**Generative UI (GenUI)**
Interaction Control	Predefined by designer	Predefined by designer	Predefined by designer	Intelligent UI generation	Co-directed interaction between designer and GenAI
Representation Fidelity	Static, fixed components	Static, fixed components	Static, fixed components	Intelligent UI runtime	Multimodal, real-time UI generation reflecting both semantic intent and contextual constraints
Temporal Context of Change	None/Minimal	None/Minimal	Adaptive on UI adaptation	Intelligent UI adaptation at runtime	Iterative generation during design-time and generative adaptation at runtime
Design Responsibility	Designer defines all aspects	Designer defines all aspects	Designer defines all aspects	Designer defines outputs	A designer guides synthesis via prompts and constraints
Adaptivity/Plasticity	Low/None	Low/None	Low/None	High generativity with responsive	High generativity with responsive, flexible outputs
System Agency	None/Minimal	None/Minimal	None/Minimal	High, proposes design variants; flexible outputs	GenAI proposes design variants; designer evaluates and iterates; the user

Key Takeaway: GenUI represents a paradigm shift in ***how interfaces are designed*** (design-time), not just how they operate (run-time).

Pattern 1: Tool-Calling and Component Mapping

The LLM's role is to understand user intent and call a predefined function ("tool"). The tool executes server-side logic (e.g., an API call) and returns structured data. This data is then passed as props to a corresponding, pre-built UI component.



Tool Definition (ai/tools.ts)

```
import { tool as createTool } from 'ai';
import { z } from 'zod';

export const weatherTool = createTool({
  description: 'Display the weather for a location',
  inputSchema: z.object({
    location: z.string(),
  }),

  execute: async function({ location }) {
    // API call logic here...
    return { weather: 'Sunny', temperature: 75,
  location };
  },
});
```

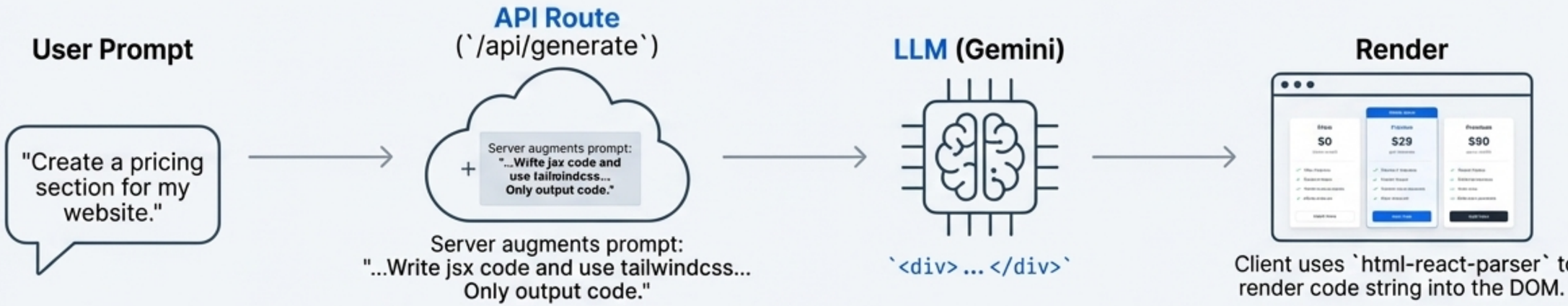
UI Mapping (React Component)

```
// In the message loop...
message.parts.map((part, index) => {
  if (part.type === 'tool-displayWeather') {
    if (part.state === 'output-available') {
      return <Weather {...part.output} />;
    }
  }
  // ...
});
```

Best For: Enriching conversations with interactive widgets where the UI components are known in advance.

Pattern 2: Direct Code Generation

The LLM generates front-end code (e.g., JSX and Tailwind CSS) directly as a string in response to a prompt. This code is then parsed and rendered on the client-side, creating UIs that did not exist before the prompt.



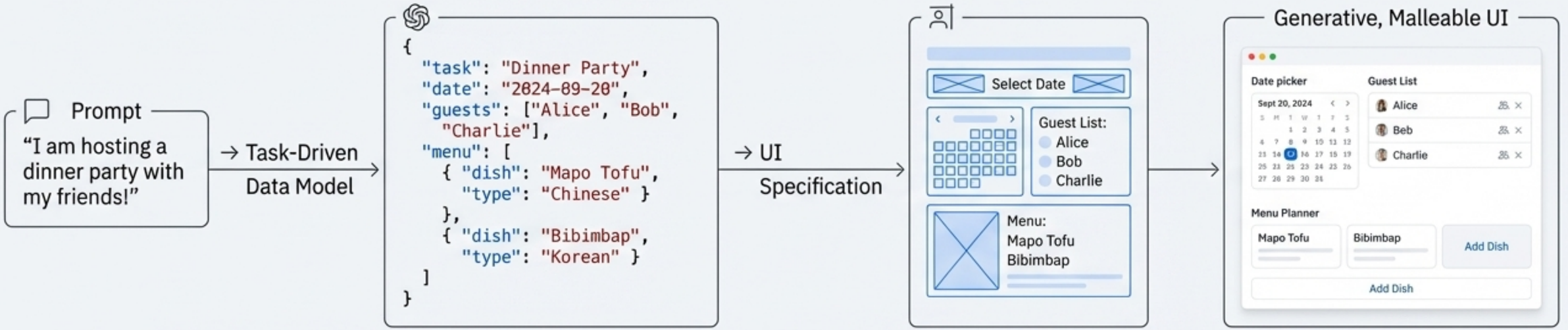
Server-Side Logic

```
// in /api/generate/route.js
import { GoogleGenerativeAI } from '@google/generative-ai';

export async function POST(req) {
  const genAI = new GoogleGenerativeAI(process.env.GEMINI_API_KEY);
  const model = genAI.getGenerativeModel({ model: "gemini-pro" });

  const data = await req.json();
  const prompt = data.body +
    " . Write jsx code and use tailwindcss for modern UI." +
    " Don't make any imports. Only output code. ";

  const result = await model.generateContent(prompt);
  const response = await result.response;
  const code = await response.text();
  //...
}
```

Best For: Prototyping tools (like v0 by Vercel), generating novel layouts, and situations where the UI structure is not known beforehand.

Pattern 3: Intermediate Data Models

Instead of generating code directly, the LLM **translates** the user's prompt into a structured, **task-driven data model (e.g., JSON)**. This **intermediate representation** is then **mapped to UI specifications**, which generate the final interface. The UI is a reflection of the data model.



Key Advantage: This approach makes the UI *malleable*. Users can modify the interface via natural language follow-up prompts, which simply update the underlying data model, leading to more consistent and predictable UI evolution compared to regenerating code from scratch.

Best For: Complex, evolving tasks where the user needs to iteratively refine the interface and its underlying data structure.

The Hidden Challenge: Managing State in Streming UIs

"As generative UIs become more complex—handling streaming, retries, and cancellation—managing state with a loose collection of `useState` booleans becomes fragile and leads to 'impossible states'."

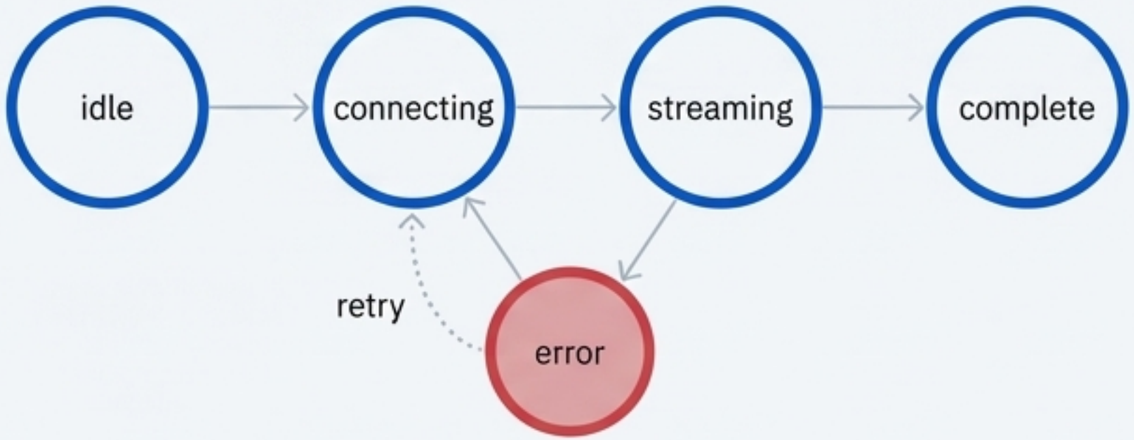
The "Ballet of Booleans"

```
const [isLoading, setIssLoading] = useState(false);
const [isStreaming, setIsStreaming] = useState(false);
const [isComplete, setIsComplete] = useState(false);
const [error, setError] = useState<Error | null>(null);
const [isRetrying, setIsRetrying] = useState(false);

// ...and so on.
```

What happens when `isLoading` and `isComplete` are both `true`? Or `error` has a value but `isLoading` is also `true`? Your UI becomes a quantum superposition of confusion.

The Solution: State Machines

A Finite State Machine (FSM) is a model that can be **in exactly one** of a finite number of states at any given time.



Applying State Machines in React: from `useReducer` to XState

State machines enforce rules about how your UI can transition between states, eliminating impossible states and making your code predictable and self-documenting.

useReducer for Simplicity

React's built-in hook is a great starting point for simple-to-moderate state machines. It centralizes transition logic and makes impossible states structurally impossible.

```
// Type definition enforces valid states
type State =
  | { status: 'idle' }
  | { status: 'loading' }
  | { status: 'success', data: string[] }
  | { status: 'error', message: string };

// Reducer function centralizes logic
function reducer(state: State, action: Action): State {
  // ...
}
```

Limitations

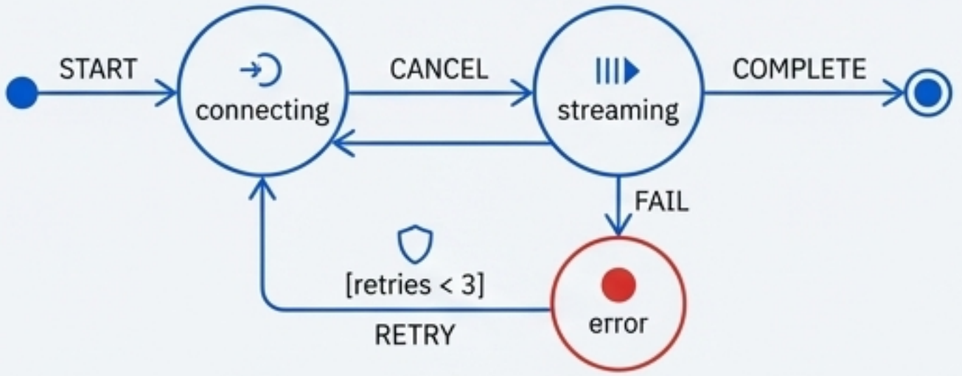
- Falls short with complex async flows like race condition prevention, guards (e.g., max retries), and entry/exit actions.

XState for Orchestration

For complex orchestration, XState provides a full framework for statecharts with hierarchical states, guards, and entry/exit actions—critical for robust AI streaming.

Key Capabilities

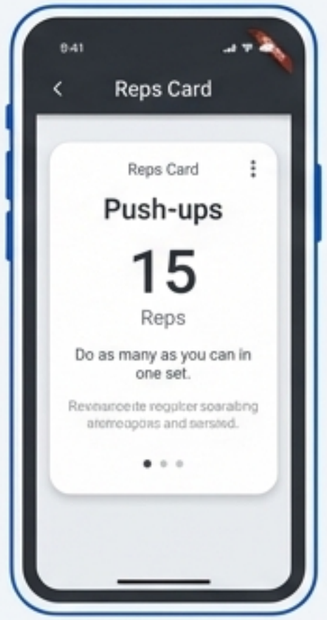
- Guards (prevent race conditions)
- Deterministic Transitions
- Entry/Exit Actions (cleanup)
- Unified Context
- Explicit Error Paths



Closing the Loop: How User Input Flows Back to the Agent

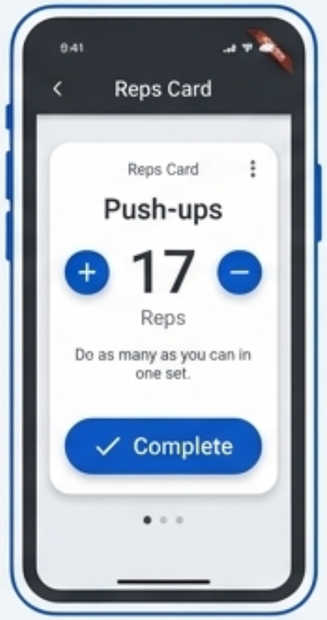
Case Study: The Flutter ‘Workout Companion’ App.

Part 1: AI Generates UI



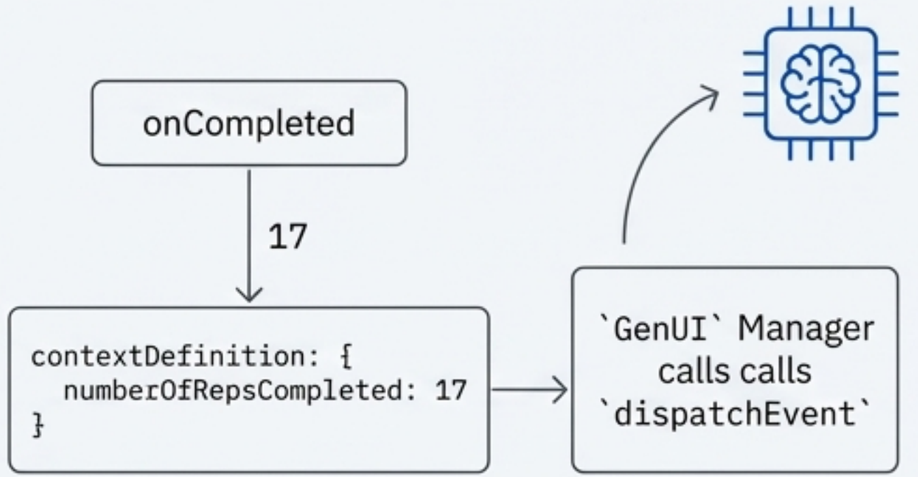
User Input

Part 2: User Interacts



Event Sent to Agent

Part 3: App Dispatches Event



The `GenUI Manager` provides the tools for widgets to send events with context back up the chain. The agent receives this data and can continue the conversation or generate a new UI based on the user’s action.

This Changes How We Design: Five Core Themes of the New Paradigm

Generative UI is more than a new tool; it reconfigures the design process, challenging established paradigms of creativity, authorship, and collaboration.

Computational Co-Creation

A human-AI partnership. The designer establishes a "creative dialogue," providing direction while the AI offers variations and alternatives.

Synthesis Over Selection

The process shifts from selecting pre-defined options (like IKEA) to continuous synthesis (like cooking), making higher-level directional choices.

Expanded Design Space Exploration

LLMs facilitate “computational divergent thinking," rapidly producing diverse alternatives that fall outside a designer's typical patterns.

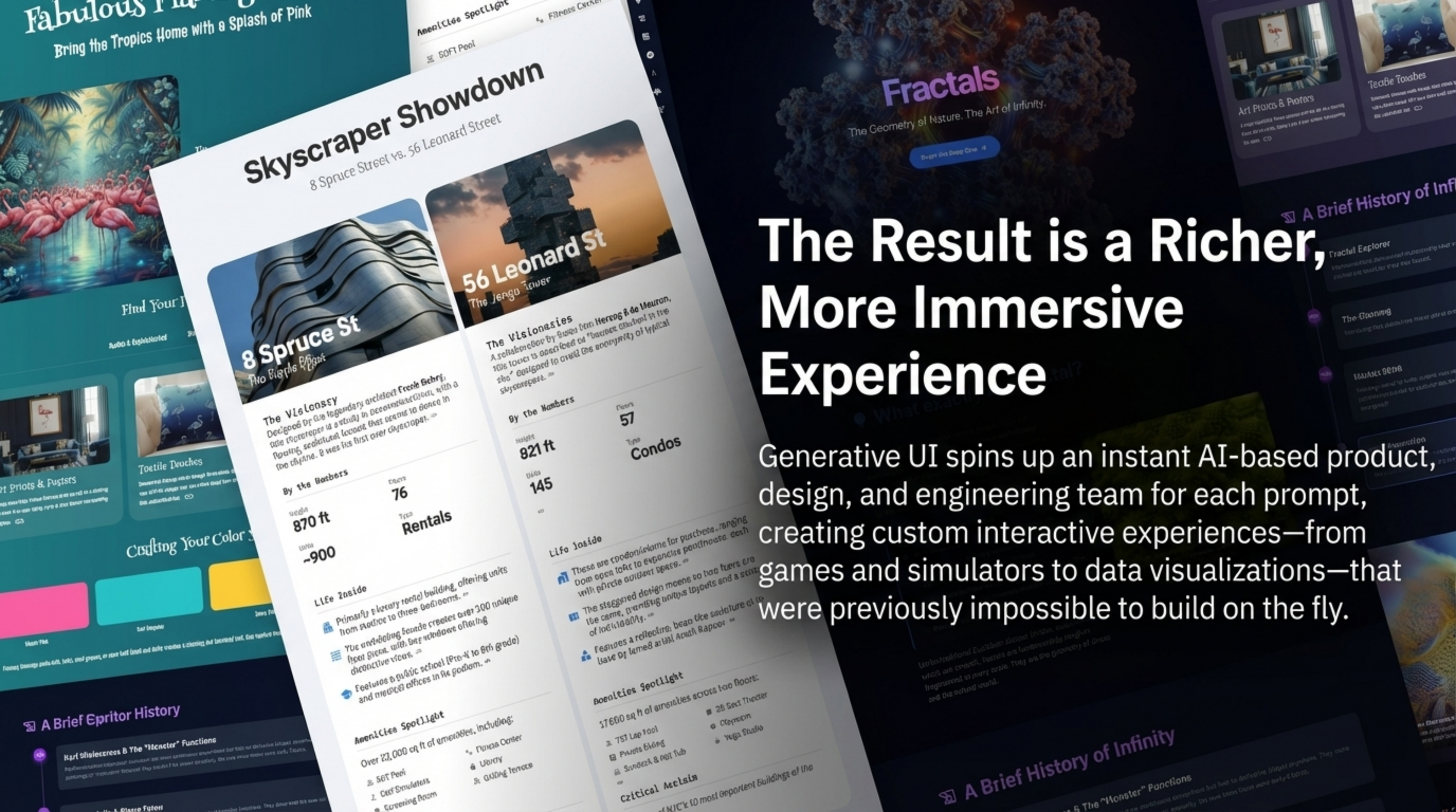
Representation Fluidity

Seamless, bidirectional translation between sketches, wireframes, visual designs, and functional code, accelerating iteration from days to minutes.

Contextual Adaptation

Systems demonstrate “implicit design knowledge,” adapting to brand guidelines, accessibility standards, and user preferences beyond explicit rules.

All themes and quotes adapted from Kyungho Lee, "Towards a Working Definition of Designing Generative User Interfaces."



The Result is a Richer, More Immersive Experience

Generative UI spins up an instant AI-based product, design, and engineering team for each prompt, creating custom interactive experiences—from games and simulators to data visualizations—that were previously impossible to build on the fly.

The Road Ahead: From Technical Hurdles to Ethical Frameworks

Technical Challenges

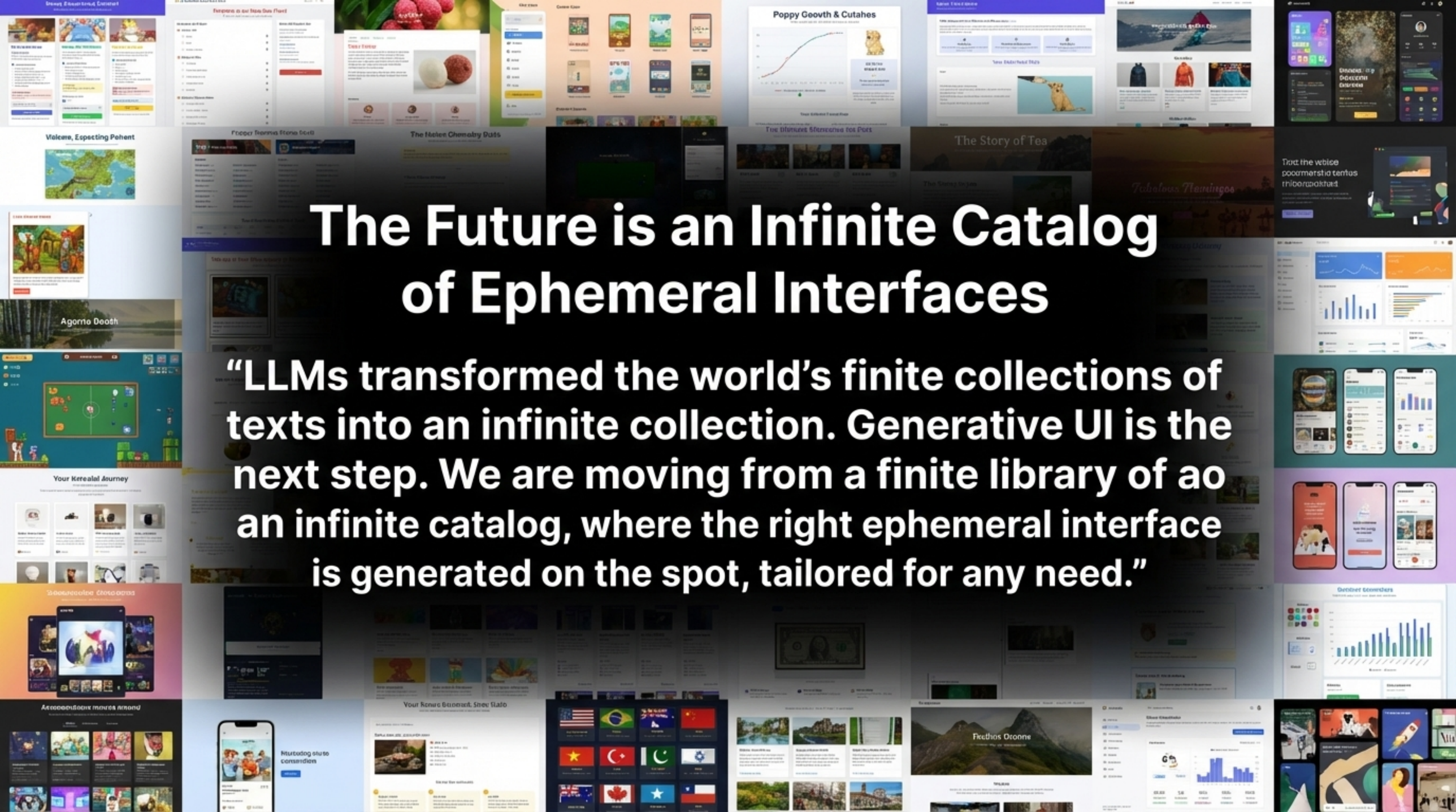
- **Generation Speed**: Current generation can take a minute or more. Streaming and speculative decoding can reduce perceived latency.
- **Functional Accuracy**: Models can create interfaces that look correct but fail in interaction logic or contain errors.
- **Workflow Integration**: Tools remain siloed. Seamless integration with existing design platforms (e.g., Figma) is essential for broad adoption.

User-Centricity

- **Malleability**: How do we move beyond one-shot generation to allow users to iteratively tailor and evolve the UI to their changing needs?
- **Value Alignment**: How do we ensure the generated UI reflects the user's actual values and user's actual values and runtime context, not just the designer's initial prompt?

Ethical Considerations

- **Bias & Representation**: Generative systems trained on historical data can inherit and amplify biases related to culture and accessibility.
- **Authorship & IP**: When AI is a **Authorship & IP**: When AI is a co-creator, who owns the result? This challenges traditional notions of authorship and intellectual property.



The Future is an Infinite Catalog of Ephemeral Interfaces

“LLMs transformed the world’s finite collections of texts into an infinite collection. Generative UI is the next step. We are moving from a finite library of ao an infinite catalog, where the right ephemeral interface is generated on the spot, tailored for any need.”