The best user experiences tightly integrate the presentation layer with the underlying AI systems

UI Layer

Model and Tooling

Model:
Inference configs
Data prep
Fine-tuning
Distillation

Tooling:
RAG
Agents
Prompt management
Guardrails
Model routing

Design, Build, Test, Deploy, Maintain…

With rule-based software, designers could define and control for all the states an end-user may encounter

The best user experiences tightly integrate the presentation layer with the underlying AI systems

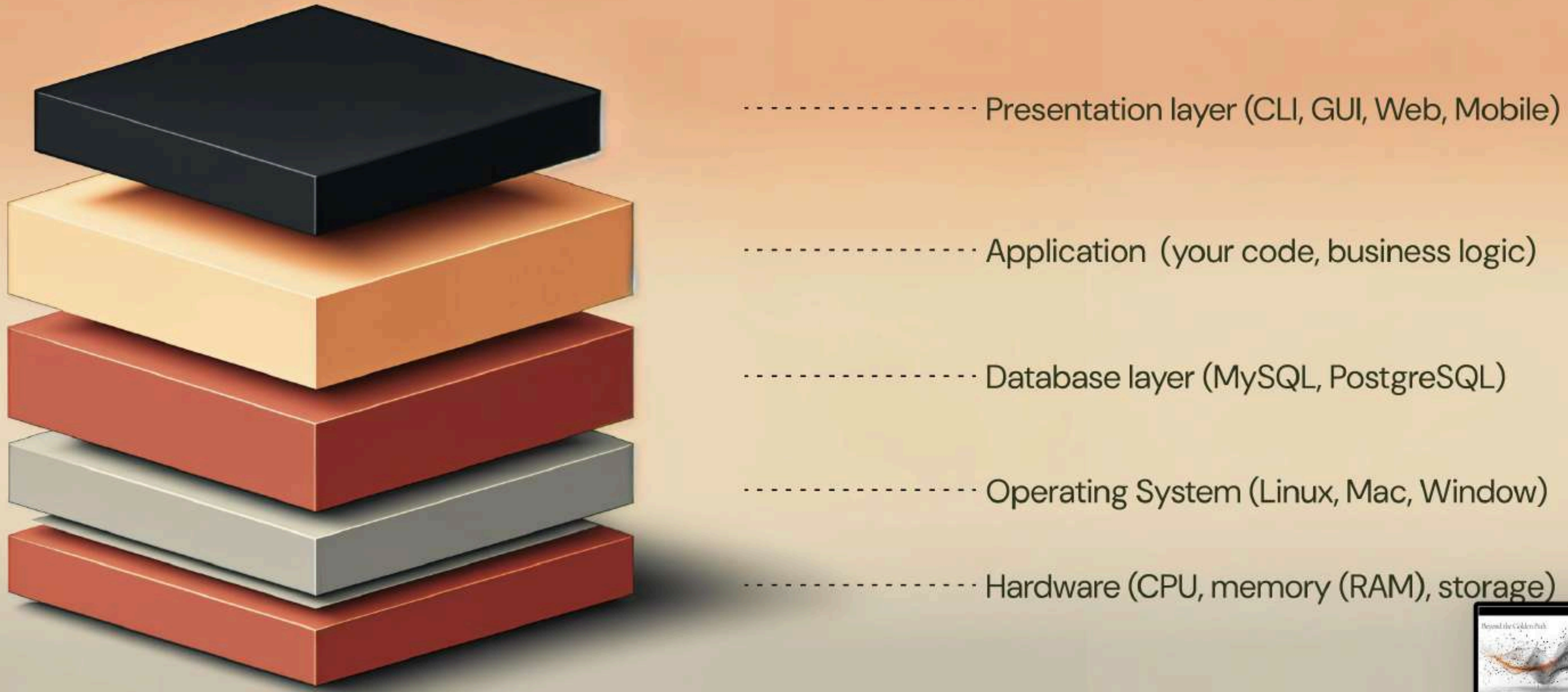
Beyond the Golden Path

AI Design shifts from a single golden path with occasional edge cases to a world where every experience is an edge case

Design, Build, Test, Deploy, Maintain...

[illegible]

Architectures shape experiences


Presentation layer (CLI, GUI, Web, Mobile)
Application (your code, business logic)
Database layer (MySQL, PostgreSQL)
Operating System (Linux, Mac, Window)
Hardware (CPU, memory (RAM), storage)

AI everywhere

Presentation layer: **Chat, Voice, Generative UI**

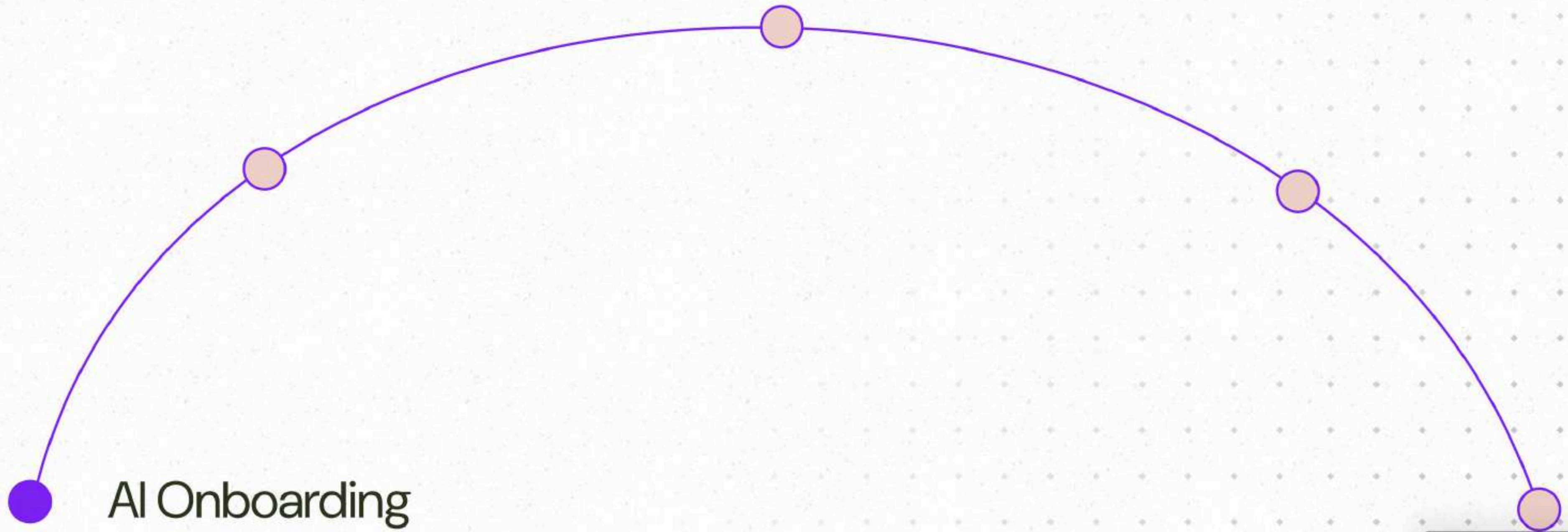
Application: **Model inference, Agents**

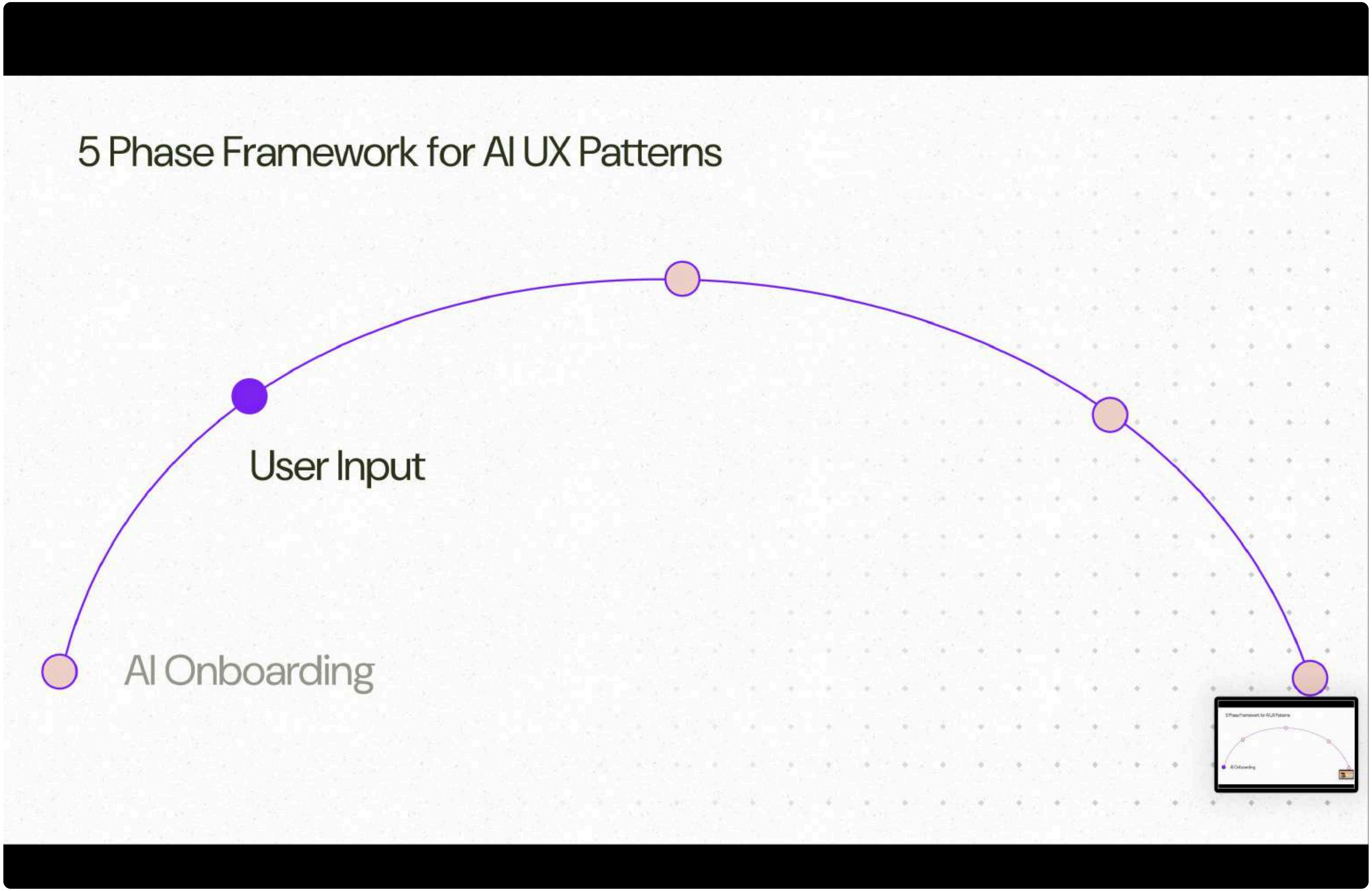
DB: **Vector databases (RAG)**

OS: **Apple Intelligence, Copilot+ PCs**

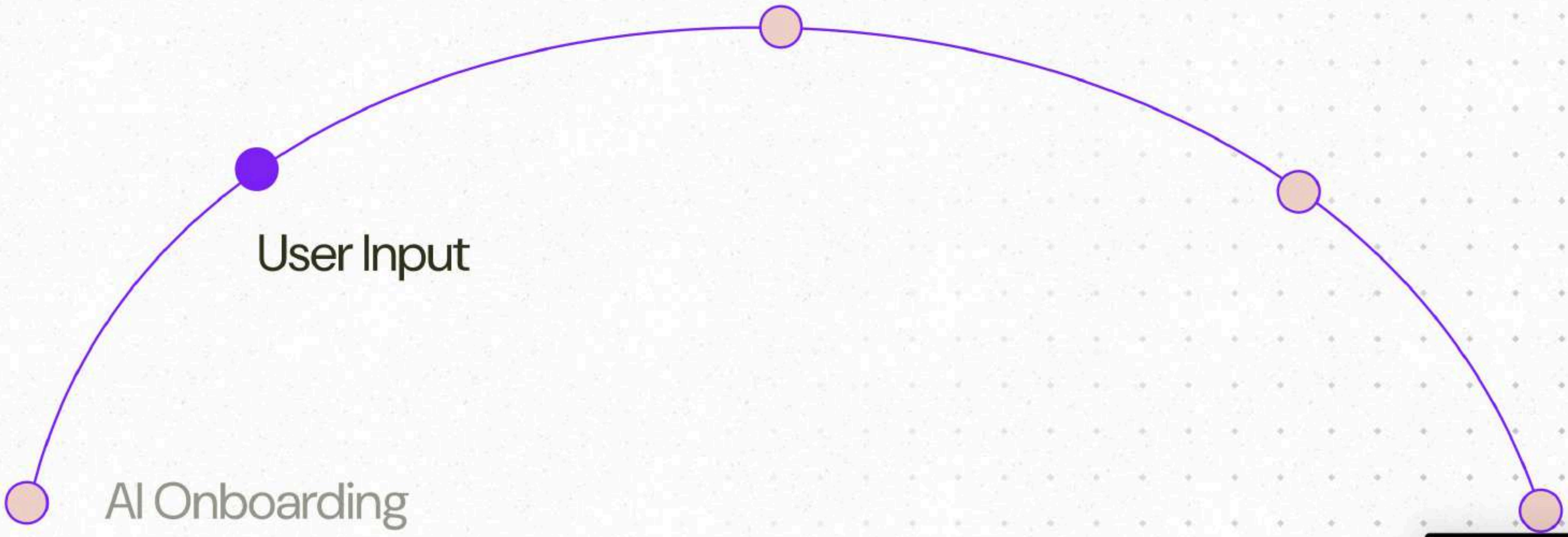
Hardware: **GPUs, HBM, SSDs**

5 Phase Framework for AI UX Patterns

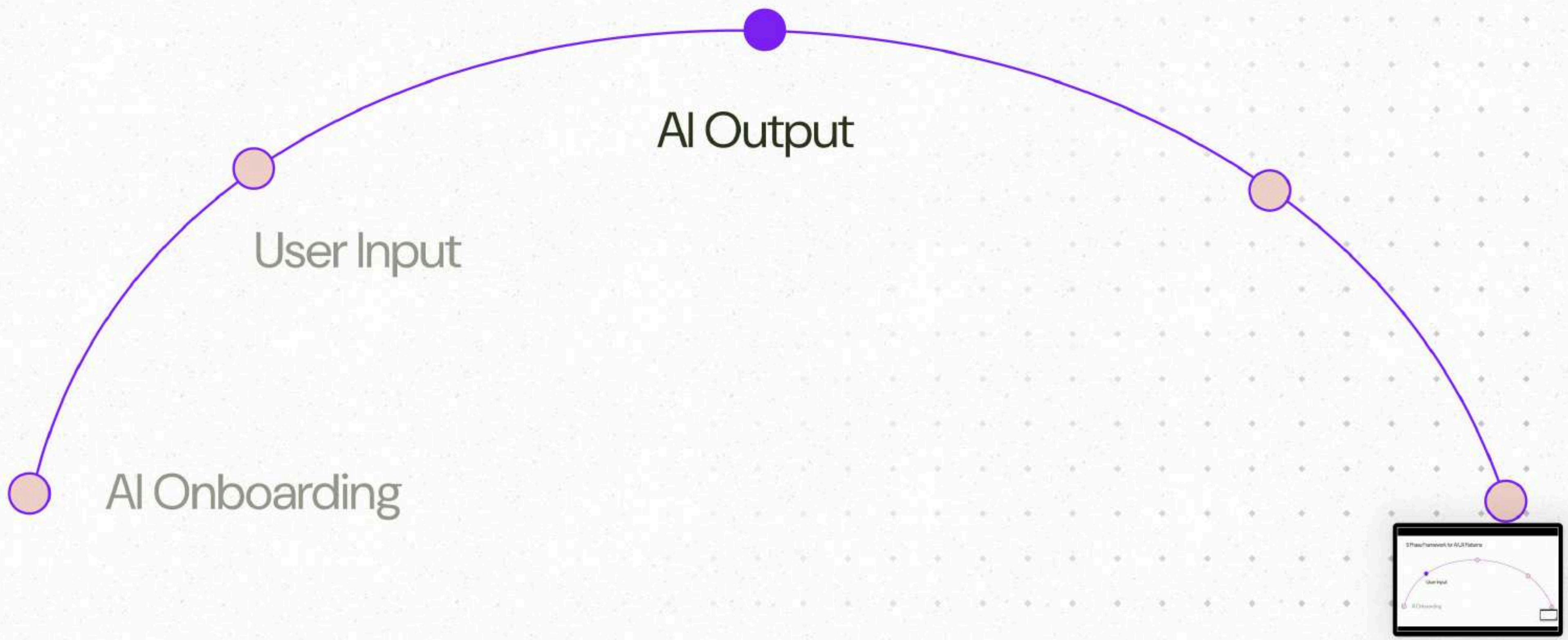




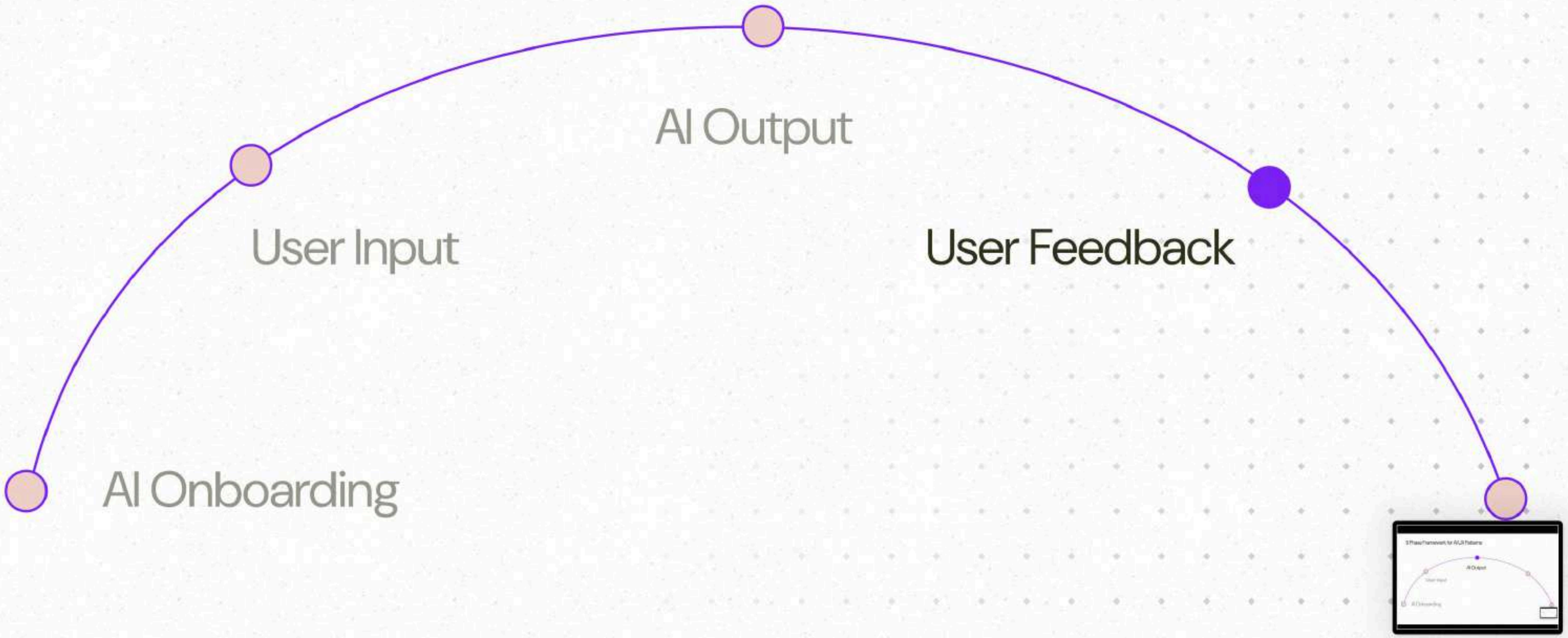
5 Phase Framework for AI UX Patterns



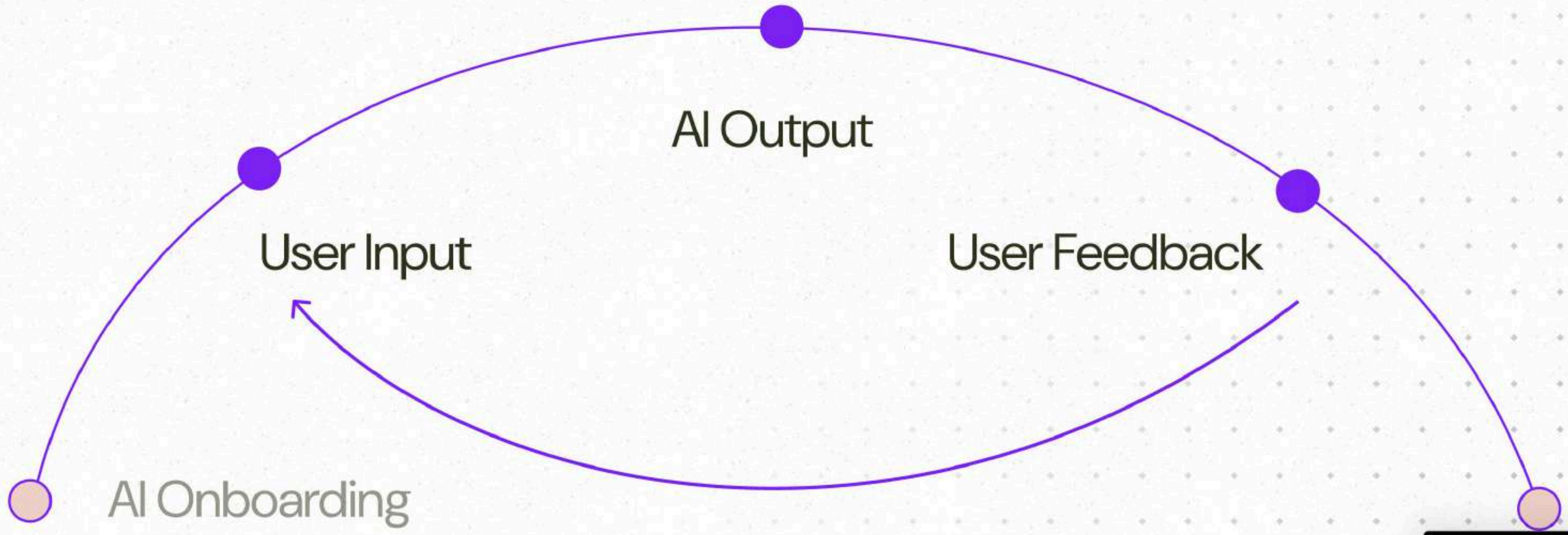
5 Phase Framework for AI UX Patterns



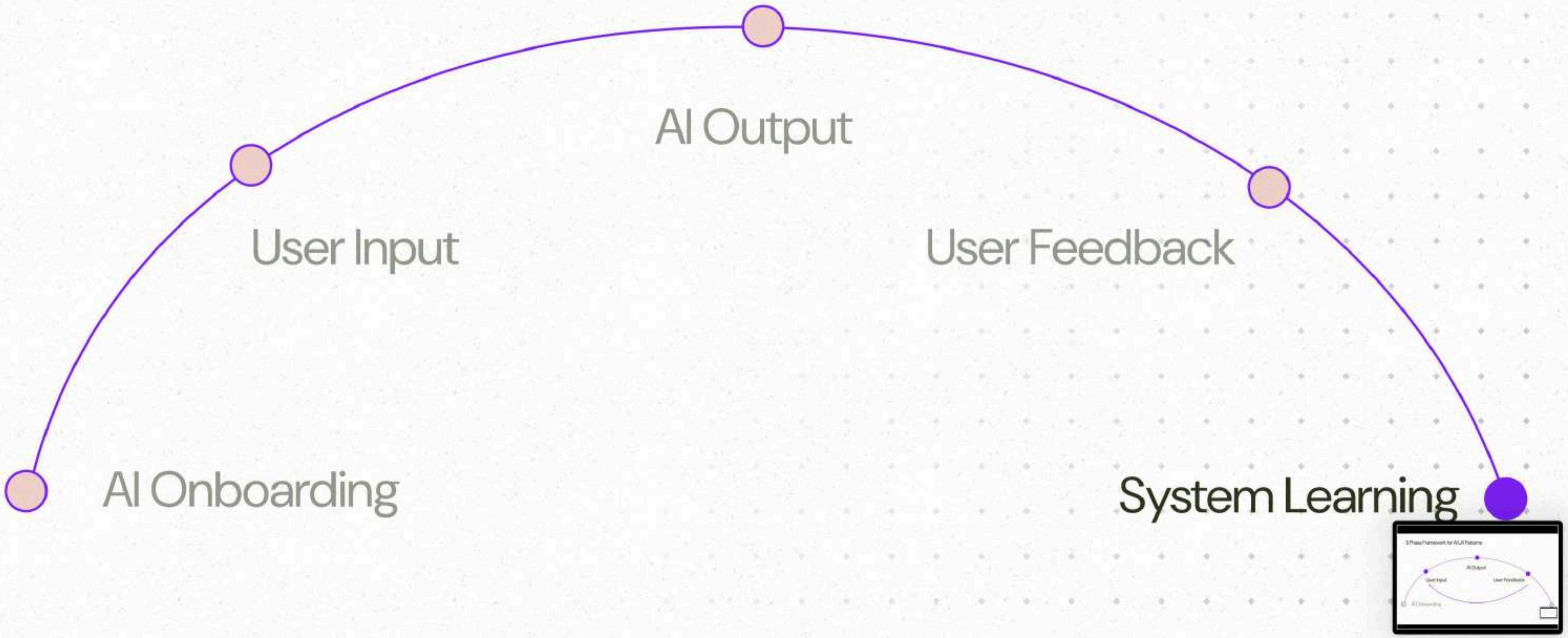
5 Phase Framework for AI UX Patterns



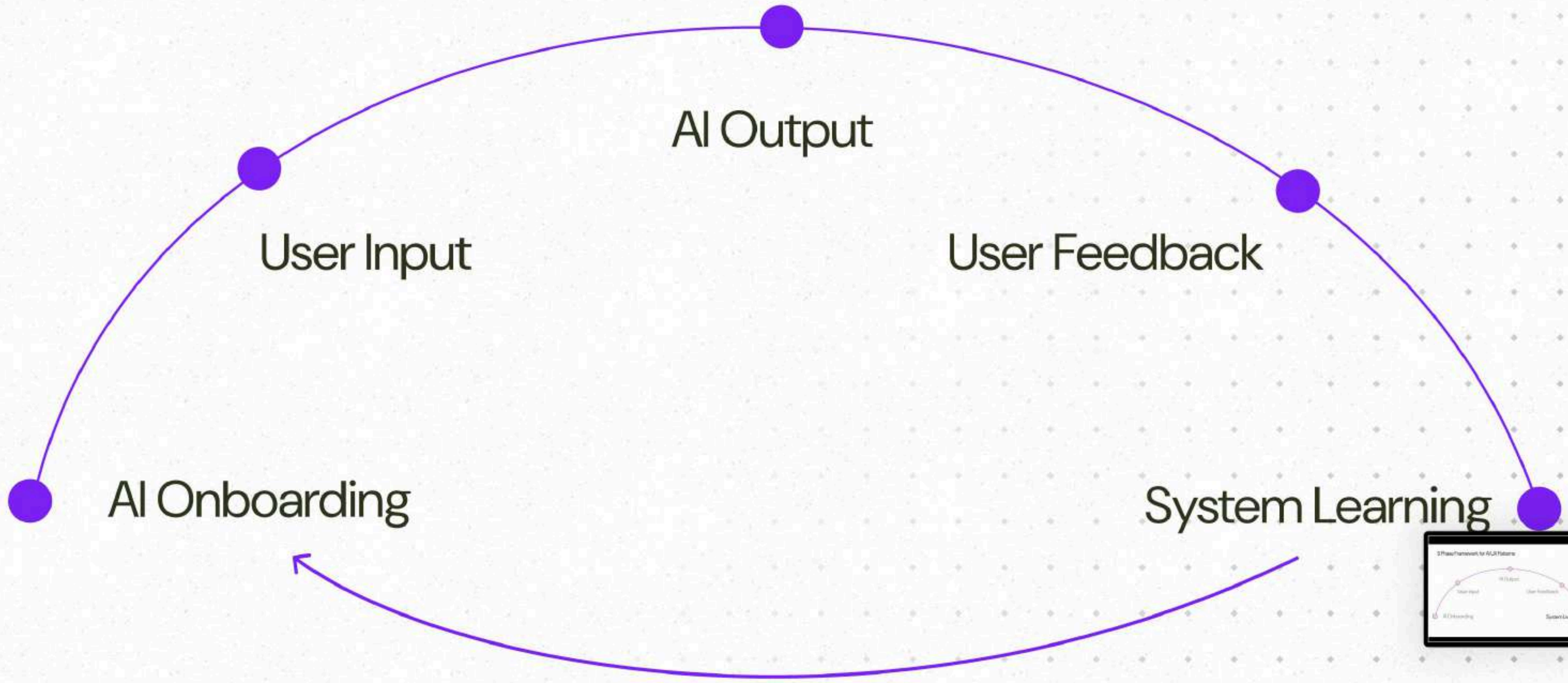
5 Phase Framework for AI UX Patterns



5 Phase Framework for AI UX Patterns



5 Phase Framework for AI UX Patterns



1/ AI Onboarding

Expectation Framing

Intent Scaffolding

5 Phase Framework for AI UX Patterns
AI Output
User Input
User Feedback
AI Onboarding
System Learning

1/ AI ONBOARDING

Expectation Framing

What it solves:

Users often assume software is deterministic.
In AI systems, this can lead to misplaced trust or frustration.

Framing expectations early — before the user even interacts — helps establish a healthy model of the system's capabilities and limitations.

Why it matters:

Users bring mental models from traditional software into AI-driven interfaces.

Without upfront messaging that frames the system as collaborative, probabilistic, and imperfect, users may expect precision and get frustrated by unexpected or "wrong" behavior.

Design Considerations:

- How do we signal that this is a collaborative system, not an oracle?
- Can we use friendly, human-centered language to build trust without overpromising?
- Are we clear about the kinds of things the AI is good and bad at?

Examples:

Product	Example Phrase	Context Location
GitHub Copilot	"I can help, but I make mistakes."	Welcome tooltip/copy
LinkedIn	"People you may know"	Section he
ChatGPT	"May occasionally produce incorrect info"	System m

1/ AI ONBOARDING

Intent Scaffolding

What it solves:

Users often face a blank slate or don’t know what the AI system is capable of. This pattern helps reduce that friction by revealing affordances or nudging users toward successful prompts or actions.

These affordances also provide a signal to the user about how "freeform" or "structured" the AI system expects interaction to be.

Why it matters:

AI systems can support an extremely broad range of use cases, and this ambiguity can lead to user anxiety, hesitation, or ineffective prompts.

By scaffolding input with examples or interactive chips, the system shows its capabilities without restricting user freedom. This is especially important for new users or infrequent users who lack prompt fluency.

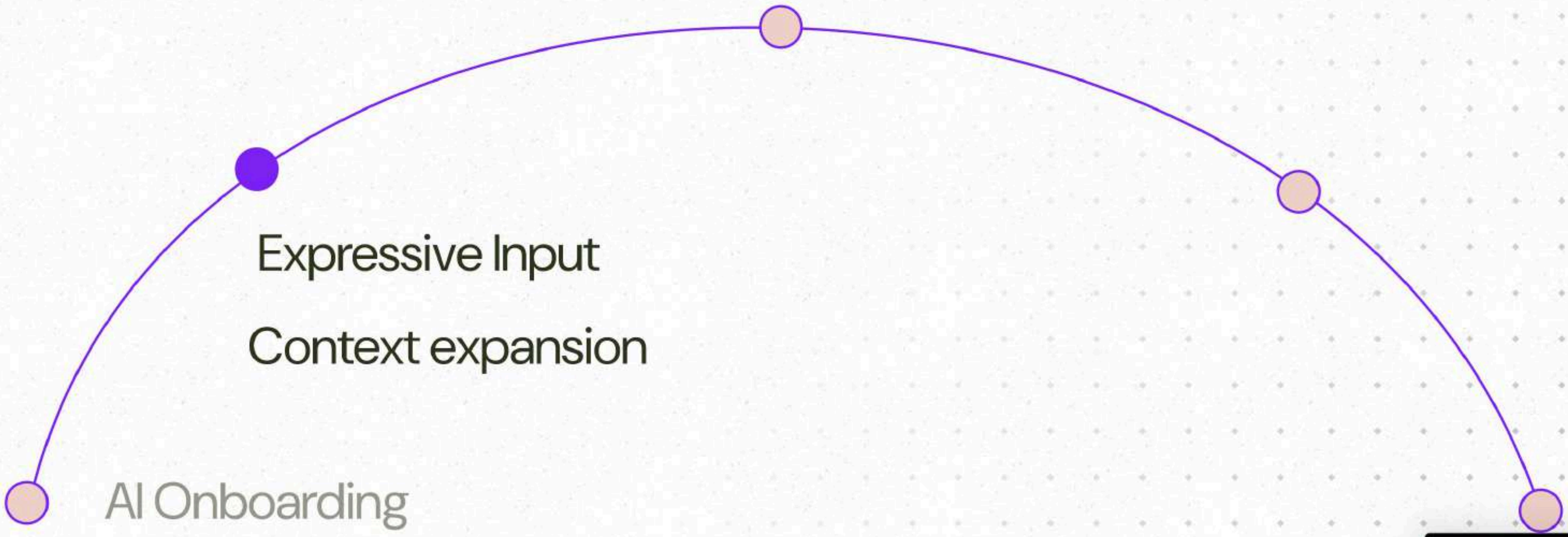
Design Considerations:

- What’s the most common thing users want to do here?
- Can we visually or contextually suggest that action?
- Are our examples diverse enough to represent different goals or styles?

Examples:

Product	Example	UI Mechanims
ChatGPT	Suggested prompts in home screen	Static buttons
Google Maps	Filter chips (“Open now,” “4+ Stars”)	Inline chip filter
Notion AI	Examples of tasks under text input	Prompt

2/ User Input



2/ USER INPUT

Expressive Input

What it solves:

AI outputs require the right input. Yet, users may struggle to express nuance, tone, or creative intent through traditional natural-language prompts alone.

Expressive input patterns provide lightweight, visual, or emotionally rich ways to shape AI behavior — enhancing intent clarity or creative exploration without requiring detailed language.

Why it matters:

These patterns make AI feel more approachable, fun, and intuitive. They invite experimentation, support faster iteration, and reduce prompt anxiety — especially for users who don't know what to type.

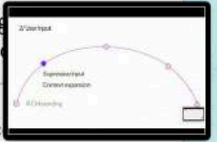
Expressive inputs can also unlock more personalized, emotionally resonant experiences without adding cognitive load. These pattern are especially helpful in creative, multimodal, or mobile-first environments.

Design Considerations:

- Are expressive inputs intuitive, or do they need onboarding?
- Do users feel in control or surprised by how the AI interprets them?
- Can these inputs be combined with traditional prompts?
- Are expressiveness tools accessible across modalities (text, voice, visuals)?
- Is it clear how the system is interpreting visual/emotional cues?

Examples:

Product	Input type	Context
Midjourney	Emoji-as-prompt	Visual generation input
Luma AI	Madlibs-style prompt builder	Encourages structured creativity
Figma	2x2 grid	Modifying range of
Canva Magic Write	Icon-based prompt builder	Guided c



2/ USER INPUT

Context Expansion

What it solves:

AI systems often struggle when they don't have access to the right information. This pattern helps users supply the content or context the AI needs — by uploading relevant documents, selecting the right data scope, or connecting to trusted sources.

Why it matters:

Giving users a way to provide context — like uploading documents, connecting tools, or selecting scopes — makes AI responses more accurate, relevant, and useful. It builds trust by showing what the system is referencing, and moves the experience beyond generic output toward something truly personalized and meaningful.

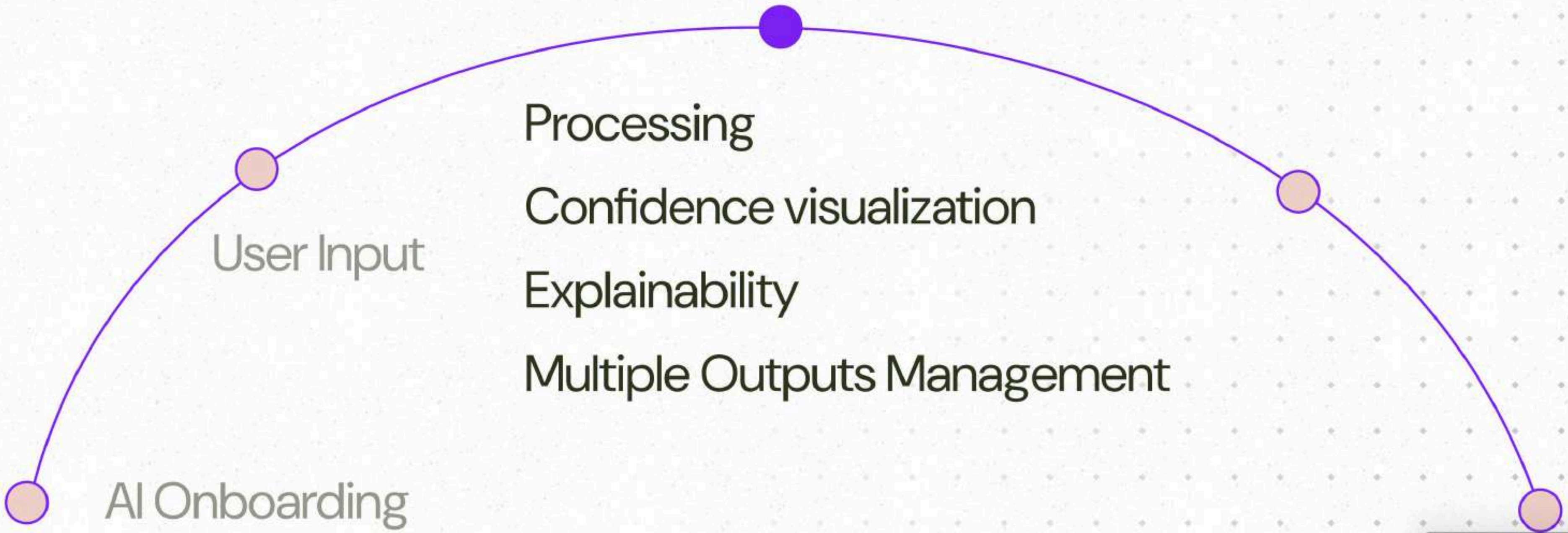
Design Considerations:

- Is it clear what the AI has access to?
- Can users see or manage the source content?
- Is the system surfacing or referencing the added context?
- Are privacy and data boundaries respected?
- Can users change or reset the scope during interaction?
- Are privacy and data boundaries respected?

Examples:

Product	Input type	Context
Perplexity	Upload document	Grounded document Q&A
Zapier AI	Add integrations to inform assistant	Workflow automation system i
Writer AI	Picks knowledge base from dropdown	Domain-

3 / AI Output



Processing

Confidence visualization

Explainability

Multiple Outputs Management

3/ AI OUTPUT

Processing

What it solves:

AI responses often involve invisible, multi-step operations — from calling APIs or MCPs, to generating drafts or running retrievals. Without clear signals that work is happening, users may feel confused, impatient, or assume the system is broken.

Processing cues reassure users that the system is active and build trust by revealing how work is being done, not just that something is coming.

Why it matters:

Good processing design shapes user expectations around time, effort, and trust. It manages impatience, prevents premature abandonment, and builds transparency — especially in agentic or tool-using systems where multiple steps are hidden. It also allows users to calibrate mental models: "Is this AI just thinking, or is it stuck?"

Design Considerations:

- Is it clear that the system is working — not stalled — through animations or visible cues?
- Is the wait time contextualized appropriately, with progress indicators or previews for longer tasks?
- Is transparency about processing balanced with cognitive load, depending on task complexity?

Examples:

Product	Processing mechanism	Context	Type
ChatGPT	Typing dots animation (...) while streaming response	Chat generation — shows active thinking	Streaming
Claude (Artifacts)	Typing animation while drafting, live updating doc preview	Full document creation with inline artifacts	Streaming + partial steps
Illicit	Show retrieval, search, and reasoning stages visibly before synthesis	Research agent surfacing work-in-progress	Processing Steps
Attio AI	Status updates like "Analyzing meeting notes..." → "Drafting summary..."	CRM workflow with multi-stage processing	[illegible]
Perplexity	Source search previews before composing full response	Retrieval-augmented Q&A system	Steps

Confidence Visualization

1 of 2

What it solves:

AI systems often produce responses that sound polished and confident — even when they're speculative, incomplete, or wrong. This pattern helps users calibrate how much to trust an output by surfacing how confident the system is, through visual, textual, or structural signals.

Why it matters:

When users can't see how confident the system is, they're more likely to assume it's right — even if the underlying model is uncertain. Confidence Visualization gives users the tools to interpret not just what the AI said, but how sure it was about it, allowing them to make better decisions: trust, verify, ignore, or dig deeper.

This pattern is essential for:

- Preventing overtrust in risky domains (e.g., health, finance, legal)
- Encouraging light-touch verification in productivity tools
- Supporting power users and developers in debugging or evaluating output

3 types of Confidence Signals

- **Text** explicitly tells the user about model certainty through labels like "Low confidence," "Experimental," "May be incorrect" or a metric like precision, recall, F1 score, confidence, or accuracy.
- **Visuals** can subtly signal the strength or weakness of the output through UI elements like opacity, grayed text, dashed outlines, iconography, and color gradients.
- **Structural** elements organize or prioritize outputs by reliability through result ordering, disclosure toggles, "Best match" vs. "Other options," and fallback results.

*For transparency int [illegible]
appeared, see the Explainability pattern.

3/ AI OUTPUT

Confidence Visualization 2 of 2

Design Considerations:

- Are we signaling how sure the system is — not just what it returned?
- Is it clear how confidence impacts the user's next action (e.g., verify, trust, ignore)?
- Is it clear how confidence impacts the user's next action (e.g., verify, trust, ignore)?
- Are the confidence signals accessible (color + text + structure)?
- Are we err-ing on the side of clarity, or confusing users with vague signals?

Examples:

Product	Confidence Signal	Modality	Display Context	Design notes
GitHub Copilot	Dimmed gray code suggestions	Visual	Inline in code editor	Suggests tentative contribution that solidifies when accepted
Salesforce Einstein	Color-coded "Likely to convert" labels	Visual	Analytics dashboard	Uses color and label to signal confidence in predicted outcomes
Perplexity	Citation count implying confidence	Structural	Below generated answers	Encourages trust through quantity and quality of supporting links
Notion AI	"Verify this content" callout	Textual	In block editor	Textual "verify this content"

*For transparency int

appeared, see the Explainability pattern.

3/ AI OUTPUT

Explainability

What it solves:

When AI generates results — especially recommendations or selections — users often want to know *why* they were shown what they see.

This pattern provides transparency into the system's reasoning by surfacing underlying logic, inputs, or past user behavior that influenced the output.

Why it matters:

In recommendation and retrieval systems, opaque results can feel random or manipulative. Explainability builds trust and supports better decisions. It also enhances the feeling of personalization by showing that the system is learning and adapting. Users are more likely to engage with outputs they understand.

Design Considerations:

- Can the user tell why this result appeared?
- Is the explanation understandable to non-technical users?
- Does it increase trust, relevance, or curiosity?

Examples:

Product	Example Explanation	Display Context
Netflix	"Because you watched X"	Below title
Amazon	"Deals related to items you've saved"	Grid header
Spotify	"Based on your recent listening"	Playlist

*For patterns that help users evaluate the reliability or strength of AI output, see Confidence Visualization.

3/ AI OUTPUT

Multiple Outputs Management

What it solves:

Sometimes there are multiple valid outputs — and the system can't pick just one. This pattern helps users compare, select, or merge options in a way that's clear and empowering, instead of overwhelming.

Why it matters:

Ambiguity is a strength of generative systems — but it's also a usability risk. If users can't tell what changed between versions, they might get frustrated or abandon the task. Done well, this pattern turns ambiguity into creative control.

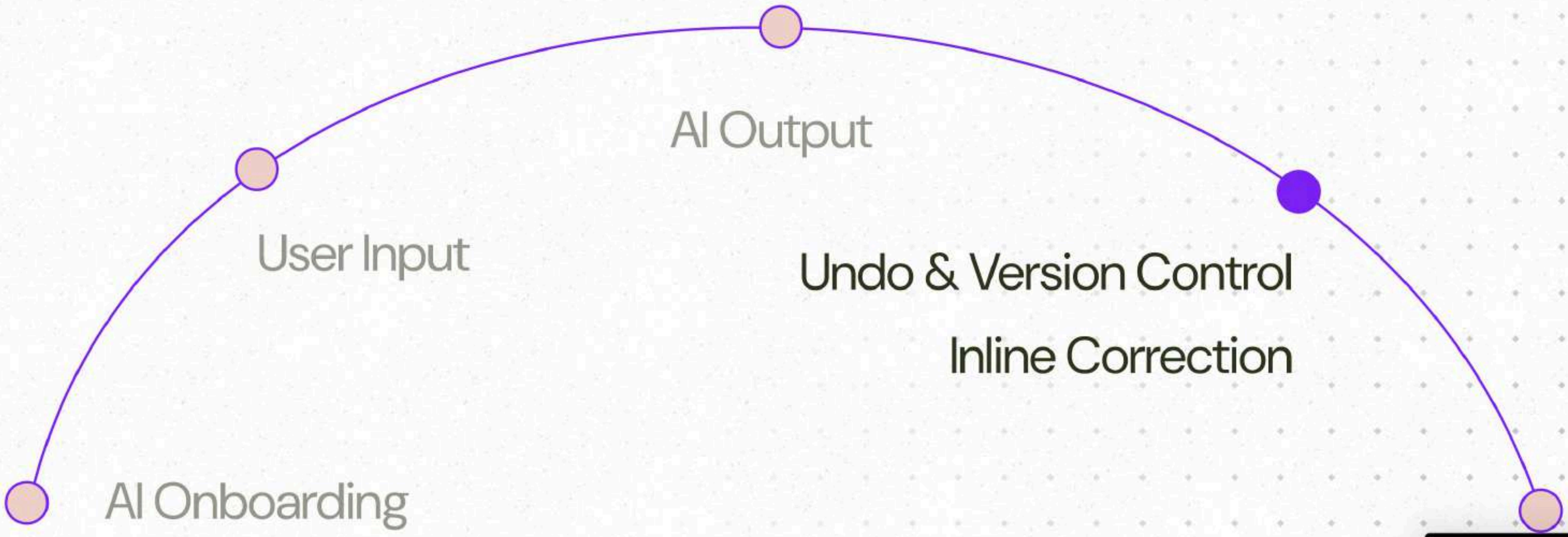
Design Considerations:

- Do users understand how options differ?
- Can they easily preview and select (or combine) results?
- Is there a fallback if all options miss the mark?

Examples:

Product	Output Options UI	Interaction type
ChatGPT	"Regenerate" and "Compare drafts	Tabbed or side-by-side compare
Midjourney	Grid of 4–8 variations	Image thumbnails with actions
Gmail	"Other drafts" preview	Click-to- draft

4/ User Feedback



4/ USER FEEDBACK

Undo & Version Control

What it solves:

In traditional UX, users know what an action will do (e.g., delete a file, apply bold). But AI outputs are generative — users may not fully expect what just changed. Undo becomes a way to understand the impact, not just reverse it. AI actions often restructure content unpredictably, making it hard to trace what happened. This pattern provides a safety net: a way to step back, compare changes, and regain control in a probabilistic system.

Why it matters:

Undo and version control in AI UX are about more than recovery — they're about exploration with confidence. These patterns invite users to try bold actions, knowing they can easily review or go back. They also make generative systems more transparent and trustworthy by surfacing what the AI changed, when, and how. In doing so, they help bridge toward memory, system learning, and long-term user trust.

Design Considerations:

- Can users clearly see what changed — and easily go back to a previous version?
- Are undo controls easy to find and placed near the relevant content?
- Can users move between versions without risk?
- If version history is retained, is it clearly communicated and easy to navigate?

Examples:

Product	Input type	Context
Notion	"Undo" after AI rewrite	Writing and document editing
Google Docs	"See version history"	Document collaboration
Cursor	"Show Diff" for AI-generated code edits	Developer IDE with inline AI edits
ChatGPT (Canvas)	Visual diff view of AI-generated changes	Structured d

4/ USER FEEDBACK

Inline Correction

What it solves:

AI output is often "almost right" — requiring small tweaks, clarifications, or rewrites. Users may want to correct tone, length, accuracy, or clarity without restarting the entire interaction. This pattern enables quick, focused corrections directly within the output, without breaking flow or jumping back into a prompt box.

Why it matters:

When users can shape AI output directly, they're more likely to adopt the system into real-world workflows like writing, productivity, and customer service. Inline correction encourages *collaborative iteration*, builds trust by showing the AI is *flexible and fixable*, and reduces fatigue by avoiding full re-prompts. It supports everyday needs like refining drafts, clarifying tone, and fixing errors — making AI feel like a true partner, not a black box.

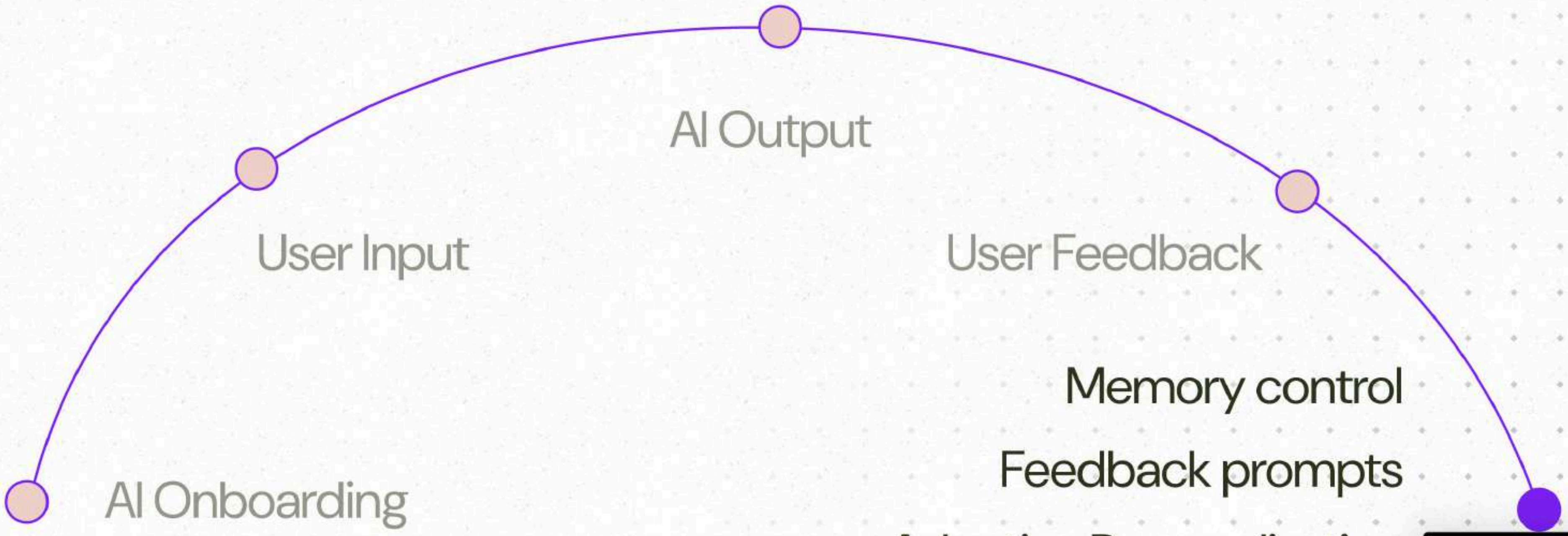
Design Considerations:

- Affordance: Is it clear which parts of the output are interactive or editable?
- Feedback loop: Is the correction immediate, understandable, and reversible?
- Efficiency: Does this reduce prompt fatigue or overall user effort?
- Tone: Do corrections feel empowering and collaborative, not overly corrective?
- Granularity: Can the user control the right level (word, phrase, sentence)?

Examples:

Product	UI variant	Description	Best for
Notion, Jasper	Hover Actions	Appears on hover as a small menu (e.g. rewrite, shorten, rephrase)	Dense editors, keeps UI clean
Notion, Jasper	Side panel controls	Edits open controls in a sidebar (e.g., sliders for tone/length)	Structured authoring tools
Sana, Wordtune	Tap-to-cycle	Click regenerates or cycles through alternates	Conversational flows, low-friction revision
Grammarly, Notion AI	Editable inline text	Output becomes directly editable, often with AI-assist cues	Conte
Jasper, Grammarly	Dropdowns with options	Predefined rewrite options (e.g., tone, structure, format)	Guide forma

5/ System Learning



Memory control

Feedback prompts

Adaptive Personalization

Inline Correction

5/ SYSTEM LEARNING

Memory Control

What it solves:

As AI systems span multiple interactions or sessions, users need clarity on what the system remembers — and control over it. Without visibility, AI memory can feel invasive, confusing, or even creepy. Memory Control solves this by giving users the ability to view, edit, or erase what the system stores about them, building transparency and safeguarding trust over time.

Why it matters:

Persistent memory can make AI systems feel smarter and more personalized — but it can also introduce risk if users feel spied on or misunderstood. Memory Control shifts the balance back toward user agency: it lets people shape what is remembered, correct what's wrong, and maintain consent as systems learn. In a world where AI feels increasingly ambient, giving users control over memory becomes essential for ethical, trusted AI UX.

Design Considerations:

- Is it clear what the system remembers — and what it doesn't?
- Can users easily view, edit, or delete stored memories?
- Is memory management accessible mid-flow — not buried deep in settings?
- Is the language around memory clear, consent-based, and non-creepy?
- Is memory opt-in or opt-out? Why?

Examples:

Product	Memory control mechanism	Context	Design variant type
Chat GPT	Toggle memory on/off, editable memory entries	Conversation history, personalization	Global Memory Toggle + Editable Memory List
Replika	Shows what it has "learned" about the user	Conversational AI / relationship bot	Editable Memory List
Alexa	Auto-expiring memories or user-triggered deletes	Voice interface	Sess Mem
Notion	Uses doc/workspace scope as "memory"	Workspace-based productivity AI	Mem Pick

5/ SYSTEM LEARNING

Feedback prompts

What it solves:

Users often want to express whether an AI result was helpful, off-track, or worth improving — without writing a full correction. This pattern introduces lightweight, structured ways for users to give input, such as thumbs up/down, emojis, or star ratings. These signals help train the system or surface future improvements, all without interrupting flow.

Why it matters:

Feedback prompts make it effortless for users to influence how the AI evolves over time. They allow systems to detect satisfaction, drift, or edge cases without burdening the user with heavy input. Especially in fast, casual, or mobile-first experiences, lightweight signals like thumbs up, star ratings, or emojis often act as the only feedback loop available. Capturing these signals thoughtfully is critical for long-term system learning, improvement, and trust.

Design Considerations:

- Is it clear whether feedback affects this interaction, future personalization, or system learning?
- Do users get a signal that their feedback was received and matters, even if results aren't immediate?
- Is giving feedback quick and effortless, without disrupting the user's flow?
- Can users optionally add more context to explain their feedback if they want to (e.g., explain a thumbs-down)?

Examples:

Product	Prompt type	Example Interaction	Context/ Best for
ChatGPT	Thumbs up/down	👍/👎 below AI responses	Conversational chat UIs, completions, summaries
Duolingo	Star rating	1-5 stars after a lesson	Learning reinforcement and satisfaction
Google Docs	Yes/No question	"Was this suggestion helpful?"	Inline feedb or tone sug

5/ SYSTEM LEARNING

Personalization

What it solves:

Personalization solves the gap between generic AI outputs and user-specific needs, styles, or intentions. It allows systems to adapt to individual users — whether by learning preferences over time (implicit) or letting users guide behavior in the moment (explicit).

Without personalization, AI outputs often feel one-size-fits-all, leading to frustration, inefficiency, or loss of trust.

Why it matters:

Personalization makes AI experiences feel more human, useful, and aligned with users' goals. It builds trust by making the system responsive to individual needs, efficiency by reducing repetitive corrections, and engagement by empowering users to steer the interaction.

In a world of increasingly powerful AI, personalization helps preserve a sense of agency, ownership, and partnership between user and system — critical for long-term adoption and satisfaction.

Design Considerations:

- How do we respect privacy while still offering meaningful personalization? (Allow opt-in/opt-out, give visibility into what's being remembered.)
- How visible should personalization be to the user?
- When should the user be invited to personalize versus when should the system adapt silently?

Examples:

Product	Personalization mechanism	Context / Purpose
Spotify AI DJ	Curated commentary based on user taste	Personalized playlist narration + dynamic music selection
Duolingo	Adjusts difficulty based on past mistakes	Personalized reinforcement learning and pacing
Midjourney	/prefer settings (like style weights)	Influences model outputs toward user's aesthetic
Perplexity	Tracks preferred sources/ topics	Personalizes toward user reading habits

Feedback prompts