

Vibe Coding in Product Teams: Reconfiguring AI-Assisted Workflows, Prototyping, and Collaboration

Jie Li*

MIT Media Lab, Cake Researcher
Delft, Netherlands
jasminejue@gmail.com

Ruihao Zhu

Cornell University
New York, USA
ruihao.zhu@cornell.edu

Youyang Hou

Notion Labs
San Francisco, USA
youyang@makenotion.com

Hancheng Cao

Goizueta Business School, Emory
University
Atlanta, USA
hancheng.cao@emory.edu

Laura Lin

Collov AI
Redwood City, USA
lauralin.uxr@gmail.com

Abdallah El Ali*

Centrum Wiskunde & Informatica,
Utrecht University
Amsterdam, Netherlands
aea@cwi.nl

Abstract

Generative AI is reshaping product design practices through “vibe coding”, where product team members express intent in natural language and AI translates it into functional prototypes and code. Despite rapid adoption, little research has examined how vibe coding reconfigures product development workflows and collaboration. Drawing on interviews with 22 product team members across enterprises, startups, and academia, we show how vibe coding follows a four-stage workflow of ideation, generation, debugging, and review. This accelerates iteration, supports creativity, and lowers participation barriers. However, participants reported challenges of code unreliability, integration, and AI over-reliance. We find tensions between efficiency-driven prototyping (“intending the right design”) and reflection (“designing the right intention”), introducing new asymmetries in trust, responsibility, and social stigma within teams. Through a responsible human-AI collaboration lens for AI-Assisted product design and development, we contribute a deeper understanding of deskilling, ownership and disclosure, and creativity safeguarding in the age of vibe coding.

CCS Concepts

• Human-centered computing → Empirical studies in HCI.

Keywords

Vibe coding, generative AI, product design, product teams, UX design, human-AI collaboration, AI-assisted programming

ACM Reference Format:

Jie Li, Youyang Hou, Laura Lin, Ruihao Zhu, Hancheng Cao, and Abdallah El Ali. 2026. Vibe Coding in Product Teams: Reconfiguring AI-Assisted Workflows, Prototyping, and Collaboration. In *Proceedings of the 5th Annual Symposium on Human-Computer Interaction for Work (CHIWORK '26)*, June 22–25, 2026, Linz, Austria. ACM, New York, NY, USA, 16 pages. <https://doi.org/10.1145/3808045.3808062>

*Equal contributions.



This work is licensed under a Creative Commons Attribution 4.0 International License. *CHIWORK '26, Linz, Austria*

© 2026 Copyright held by the owner/author(s).

ACM ISBN 979-8-4007-2429-9/26/06

<https://doi.org/10.1145/3808045.3808062>

1 Introduction

The recent proliferation of Generative AI (GenAI) tools has transformed how people write [54] and design digital experiences [86], including how product team members prototype, develop, and deploy software. Built on large language and multimodal foundation models [6], systems such as ChatGPT¹, GitHub Copilot², Claude Code³, Cursor⁴, Lovable⁵, and Bolt⁶ translate natural-language prompts directly into working code, user interfaces, and interactive prototypes. Earlier AI-assisted tools, such as code autocompletion or small code snippet suggestions, were designed mainly to speed up single tasks without changing overall workflows [85]. By contrast, these new vibe coding systems act as continuous partners: they can generate entire components, integrate APIs, and adapt outputs to user context, collapsing once-separate stages of ideation, prototyping, and implementation into a single conversational workflow. This emerging style of working with AI, recently characterized as addictive [70], has been described as “vibe coding” [44], a term that entered popular discourse after Andrej Karpathy’s 2025 post on Twitter/X, describing it as “*There’s a new kind of coding I call “vibe coding”, where you fully give in to the vibes, embrace exponentials, and forget that the code even exists.*” [44].

In online communities, vibe coding often refers to a prompt-to-product workflow where high-level creative intent (“the vibe”) is translated into functional artifacts with minimal setup using generative agents, sometimes skipping human-led review or manual edits. Academic work has since expanded the concept: Horvat [40] frames vibe coding as a paradigm where developers act as high-level coordinators, guiding AI through iterative prompts rather than writing line-by-line code; Sarkar et al. [73] describe it as programming through dialogue; and Meske et al. [61] emphasize how it reconfigures intent mediation by redistributing epistemic labor between humans and machines. In this paper, we adopt a broad definition of vibe coding that reflects emerging professional practice: while workflows are conversational and intent-driven, they still involve human-led debugging, verification, and iterative refinement. This

¹<https://chat.openai.com>

²<https://github.com/features/copilot>

³<https://code.claude.com/>

⁴<https://www.cursor.com>

⁵<https://lovable.dev>

⁶<https://bolt.new>

broader definition distinguishes our work from narrower notions of “pure vibe coding” that are limited to quick, disposable prototypes. We frame vibe coding as relevant to product team workflows involving manual debugging, verification, and collaboration, even though many participants currently use it mainly for personal projects or early-stage prototyping rather than production. Our scope also accounts for the recent uptake of what is being called “agentic coding” or “agentic engineering”, terms which imply a higher degree of autonomous AI involvement than vibe coding [72]. Despite its rapid spread, little empirical research has examined vibe coding’s implications for product design and development workflows, cross-functional collaboration, and organizational adoption [83]. Open questions remain about how these tools fit with established pipelines, how they reshape human–human and human–AI collaboration, and how product-focused professional roles should be defined when the boundaries between UX designers, front-end engineers, and product managers or owners become increasingly blurred (cf., [52, 84, 86]). To this end, we ask:

- **RQ1:** How do product team members integrate vibe and AI-assisted coding into their design workflows, from ideation through prototyping to implementation?
- **RQ2:** What opportunities and risks do product teams envision for human–human and human–AI collaboration in the age of vibe coding?

To address these questions, we conducted in-depth interviews with 22 product team members working across sectors and company sizes. We use the term product team members to refer to professionals involved in product design and development whose daily work contributes to shaping user experience, including software engineers (particularly those in front-end and prototyping roles), UX/UI designers, product managers and product owners, and startup founders. This framing reflects the reality that vibe coding blurs traditional role distinctions, making product design and development a distributed responsibility across multiple functions.

We find that vibe coding has become a solid means for mediating between human goals and outputs, creating shifts in expertise distribution and new roles across product team practices (from prompt engineering to agentic AI oversight [72]), while simultaneously raising tensions between speed-driven prototyping (“intending the right design”) and deeper reflection (“designing the right intention”). Moreover, AI over-reliance in product design and development work not only erodes individual skill, but also introduces new asymmetries in trust, responsibility, and social stigma within product teams, paradoxically widening the gap between “AI-literate” and “AI-dependent” practitioners. Furthermore, while confidence in vibe coding outputs can erode critical thinking, it can also serve as an educational opportunity; however, only for those already skilled in coding. We also find that product team members emphasize resisting homogenization by advocating for critical thinking, creative restraint when vibe coding, and intentionally crafting learning moments that can result in variation and artistic exploration when designing through vibe code. Lastly, in vibe coding, we find that ownership is not tied to code execution but to the act of intention-setting, creating a new distinction between authorship of ideas and executional labor. Such a reconfiguration raises questions about how to best disclose AI usage, ownership

and authorship for code and design, and accountability in product teams. We discuss the implications of vibe coding, within the lens of ensuring responsible human-human and human-AI collaboration within AI-Assisted product design and development practices.

2 Background and Related Work

Below we review (1) GenAI tools for product design and development work; (2) how generative AI reconfigures workflows, roles and collaboration in cross-functional product teams; and (3) how vibe coding is emerging as an intent-driven authoring paradigm.

2.1 Generative AI-powered Tools for Product Design and Development Work

HCI research has long examined how to support product design and development through new tools [11, 32, 47]. As generative AI has entered practice, researchers have begun embedding it directly into design and development workflows, reshaping how product team members explore ideas, prototype solutions, and make decisions across roles [34, 43, 82, 86, 91]. Evaluations consistently show AI can augment human creativity by providing novel perspectives [15] and enabling two-way exchanges of ideas between practitioners and systems [51]. A wide range of systems illustrate these directions. Prompt-to-UI tools such as WireGen and Prompt-Infuser generate mid-fidelity wireframes or bind prompts to Figma elements, improving speed and iteration across early-stage product exploration and handoff [28, 52]. Co-ideation systems like Cobble, DesignWeaver, and UIDEC scaffold sketching, text-to-image exploration, and constraint-driven ideation, helping teams diversify ideas while staying within workflow constraints [10, 48, 55, 79, 87]. Exploration and search tools include VINS, which retrieves inspirations from sketches [10]. Studies of brainstorming systems like Xu et al.’s reflective design framework [95] and Wang et al.’s AIdeation platform [93] show that AI support yields more diverse concepts, higher satisfaction, and efficiency. Other work extends AI into integrated environments that support team-level decision-making. Zamfirescu-Pereira et al. [98] present Pail, an IDE with multiple LLM agents for exploring problem framings, decisions, and assumptions, enabling teams to reflect on alternatives and trade-offs rather than producing isolated outputs. Shin et al. [77] used LLMs to generate and simulate personas to aid empathy, while Duan et al. [23] developed a GPT-4-based system that provides feedback on UI mockups, which was useful for identifying early errors and textual issues, but offered less value as designs matured across iterations.

Prior works further map the design space of AI-powered creativity support [62], showing that they augment workflows but also introduce new responsibilities for product teams (e.g., debugging, error management, and bias mitigation [19, 52]). Scholars also however highlight barriers to adoption: Chat and Wong [13] show that product teams may avoid using generative AI because of collaboration demands, concerns about professional accountability, and organizational restrictions. Shin et al. [78] further show that LLM-assisted reframing can exacerbate disparities between novices and experts, raising concerns about agency and skill development within teams. As commercial tools rapidly advance and become embedded in everyday product work, our study builds on this foundation by examining how product team members adopt

and integrate vibe coding, through tools such as Copilot and Cursor, into their collaborative workflows.

2.2 Generative AI's Impact on Workflows, Roles, and Collaboration

Generative AI has demonstrated wide-ranging effects on creative and knowledge work. Empirical studies demonstrate measurable productivity effects: LLMs reduce task completion time in writing and communication [64, 67], improve call center performance especially for lower-skilled workers [9], and boost consultant performance in field settings [20, 21]. At the same time, impacts vary: enabling advanced AI tools can slow skilled developers [4], workers benefit differently depending on their experience profile [92], and long-term creativity may be diminished despite short-term gains [49]. Non-experts also struggle to design effective prompts, often treating LLMs as human interlocutors rather than computational tools [99]. Beyond productivity, generative AI raises concerns about bias, stereotyping, privacy, environmental cost, hidden labor [80], and risks of deskilling [26, 52]. Scholars emphasize the importance of keeping humans “in the loop” to maintain expertise, accountability, and meaningful engagement in AI-augmented work [2, 76].

These dynamics are becoming visible in product teams. Product teams increasingly use AI to generate alternatives, probe edge cases, and explore copy or layout suggestions, expanding the space of possible solutions while negotiating trade-offs across roles [88, 96]. Interviews with professional UI/UX designers find they value AI as a creative partner during divergent stages, while still emphasizing control and alignment with established practices [46]. Furthermore, designers now prototype through code, while engineers generate copy or layout ideas, blurring traditional divisions of labor [69, 97]. This flexibility raises ambiguity around authorship, ownership, and disclosure [16, 24]. Studies show that users often self-attribute AI-generated outputs [22], while others highlight tensions between disclosure and originality in collaborative work [38]. Teams adopt new governance strategies such as rotating “AI editor” roles, shared prompting conventions, and lightweight mechanisms to track model use [19, 57, 76], especially in light of disclosure requirements [24]. Lastly, research highlights collaboration challenges in cross-functional product teams developing AI systems. Subramonyam et al. [83] find prompt-guided prototyping broadens participation across roles, while Bruun et al. [8] document risks of compartmentalization, asymmetric power, and weak cross-disciplinarity. This suggests a tighter integration between UX, engineering, and AI expertise is needed. Together, the foregoing shows that while generative AI expands creative capacity and reconfigures collaboration within product teams, it also raises challenges for authorship, accountability, trust, and governance. Our study builds on this work by examining how vibe coding reshapes workflows and collaboration across product team members.

2.3 Vibe Coding as a New Form of AI-Assisted Authoring

Vibe coding captures a shift from manual artifact creation toward intent-driven collaboration with generative AI systems. Sarkar and Drosos [73] define vibe coding as a workflow of goal setting, prompting, evaluation, and iterative refinement, while Meske [61] frame it

as a collaborative flow in which human intent is mediated probabilistically rather than deterministically. Unlike earlier generative AI tools that focused on discrete tasks or outputs, vibe coding treats AI as a continuous authoring partner across the entire making process. Sapkota et al. [72] further distinguish vibe coding from agentic coding, which they posit enables autonomous software development, achieved with goal-driven agents with minimal human intervention. However the two nevertheless highly overlap, especially within product teams.

HCI and software engineering research has begun to document how these practices reshape programming workflows. Tools such as Codex, Claude Code, and Copilot accelerate production but re-allocate effort from authoring to validation and debugging, often characterized as a “generate–test–edit” cycle [45, 57]. Ross et al. [71] demonstrate how conversational assistants support not only code generation but also explanation and iterative exploration, extending interaction models beyond direct manipulation and search. Large-scale surveys reveal a complex balance of benefits and concerns: programmers value efficiency and clarity but report challenges with inaccuracies, limited contextual awareness, and over-reliance, alongside frustrations when outputs fail to meet functional requirements or resist steering [1, 53, 74]. Despite these insights, most prior work focuses narrowly on professional developers. As vibe coding practices diffuse into cross-functional product design and development teams, open questions remain about how AI-assisted authoring reshapes team workflows, role boundaries, and coordination practices. Our study addresses this gap by examining vibe coding in the context of product teams, specifically how it is being adopted and how it reconfigures collaboration, responsibility, and decision-making across roles in everyday product design and development work.

3 Method

We employed a semi-structured interview methodology for this study [58] to investigate how professionals across design, engineering, and product roles integrate AI coding agents into their UI design and frontend implementation workflows. Our aim was to capture their goals, workflows, challenges, and perceptions of these tools’ value in practice.

3.1 Participants and Interview Procedure

Our target group included professionals involved in product design and development, including UX designers, frontend and backend engineers, product managers, UX researchers, and hybrid designer–developer roles, reflecting the interdisciplinary nature of vibe coding practices.

Recruitment. We distributed a screening questionnaire across online communities and networks frequented by these professionals, including *LinkedIn*, *Twitter/X*, *Discord* servers for AI-assisted development, and mailing lists for AI design tool communities. The questionnaire captured professional role, years of experience, organizational context, and prior use of AI coding agents (e.g., *Cursor*, *Replit*, *Bolt*, *V0*, *Lovable*) for generating or refining UI artifacts. Eligibility required hands-on experience with at least one such tool for professional work or substantial side projects. We received 50 responses and selected 22 participants to ensure diversity across roles,

organizational sizes, regions, and use cases. This sample size aligns with prior HCI research on thematic saturation, which typically occurs around 15 participants ($M = 15$, $SD = 6$) [12, 39].

Interviews. Each participant completed a 60-minute online interview between May and June 2025. One researcher conducted the interview while another took notes. With participant consent, all sessions were audio-recorded and transcribed. The protocol included open-ended questions on: (1) background and current workflows; (2) experiences and challenges with vibe coding tools; (3) collaboration dynamics within teams; and (4) perceived value, limitations, and future expectations. Full interview guide is provided in [Supplementary Material A](#).

Participants. To ensure anonymity while maintaining traceability, participants were labeled P1–P22. The sample comprised 11 females and 11 males from North America ($n = 12$), Europe ($n = 8$), and Asia ($n = 2$). Professional experience ranged from less than 3 years ($n = 5$) to over 5 years ($n = 10$). Organizational sizes ranged from large enterprises ($>10,000$ employees; $n = 9$), to large organizations (1,000–10,000; $n = 4$), medium organizations (500–1,000; $n = 2$), small startups (<50 ; $n = 4$), and academic institutions ($n = 3$) (Table 2). All participants had direct experience with vibe coding tools, most commonly *Cursor* ($n = 16$), *Replit* ($n = 11$), *Lovable* ($n = 9$), *Bolt* ($n = 8$), *V0* ($n = 8$), *Figma Make* ($n = 2$). Many combined tools in multi-step workflows. Table 1 summarizes participant roles, example tasks, and typical tools.

3.2 Data Analysis

We used Miro⁷ for remote collaboration on data analysis. The audio recordings of the interviews were transcribed. We analyzed the data following a deductive/inductive hybrid thematic analysis approach [29, 94]. First, five researchers coded the transcripts deductively according to five categories in our code manual (see [Supplementary Material B](#)), developed from the research questions and interview guide. With participant consent, all coded data were turned into 617 digital statement cards on Miro, each containing a participant ID (P1–P22), a direct quote, and a one-sentence summary. Instead of calculating inter-rater reliability, we reached consensus through daily meetings and workshops [60].

Given broad initial categories, we applied an inductive approach to cluster the statement cards using affinity diagramming [35], consolidating them into themes and sub-themes. This resulted in five final themes. To ensure trustworthiness (cf., [63]), we describe our full five-step hybrid analysis process in [Supplementary Material B](#). All procedures adhered to data protection and privacy (incl. GDPR) requirements and ethical policy of X author’s institution.

4 Findings

Our analysis revealed five interrelated themes. **Theme 1: Vibe coding workflows** encompassed a four-stage process of ideation, AI generation, manual debugging, and testing, together with best practices and contrasts to Figma. **Theme 2: Benefits** included productivity, cognitive offloading, learning, and creativity. **Theme 3: Challenges** covered unreliability, difficulties in incremental refinement and debugging, pseudo productivity, over reliance, limited

context, and integration barriers. **Theme 4: Trust and accountability** addressed shifting AI trust, security and privacy concerns, and debates on authorship. **Theme 5: Collaboration** described AI driven role shifts, tensions between junior and senior roles, AI as a team member, and organizational barriers.

4.1 Theme 1: Four Stages in a Vibe Coding Workflow

Theme 1 outlines a four-stage workflow that blends creative planning, AI-supported generation, and human refinement to accelerate prototyping while maintaining quality. Although structured sequentially, the process is **highly iterative**: Testing and Review (Stage 4) often leads back to Debugging (Stage 3) or AI Generation (Stage 2), while debugging may expose missing context that returns work to Ideation (Stage 1). This back-and-forth continues until both functional reliability and design intent are achieved (Fig. 1).

4.1.1 Theme 1.1: Best Practices for Each Stage in the Vibe Coding Workflow. Stage 1: Context Setup and Ideation. Before generating anything, participants described spending time constructing the problem in a form that AI could act on. This often involved sketching architectures, defining goals, identifying pain points, and collecting visual or textual references such as screenshots, prior design versions, or Figma snippets. General-purpose LLMs like ChatGPT, Gemini, and Claude were used not primarily to generate final artifacts, but to support early reasoning through brainstorming, option comparison, and background research.

A recurring **best practice in Stage 1** was what participants effectively described as *conceptual decomposition*: translating vague intentions into smaller, operationally meaningful elements. One participant illustrated this with a telling metaphor: “If I just say ‘create me a glass of water,’ it might make a model of that shape which could be glass, or bricks, or wood, too generic to be useful. Instead, you need to break it down: first the water, then how much water, then the glass, then the shape, and finally put them together.” [P14] Participants noted that the process forced them to become more explicit about assumptions, constraints, and desired outcomes.

Participants also used LLMs to reason through the problem before committing to a direction. Some asked models to compare strategies, organize requirements, or provide structured alternatives. “I feed the whole problem into the model, and it comes back with a table of strategies ranked by intrusiveness.” [P6] Participants also emphasized the importance of retaining space for independent thinking before involving AI. “I always set aside time to brainstorm on my own.” [P22]

Stage 2: AI Generation and Refinement.

Once goals and references were established, participants moved into execution-oriented interaction with specialized vibe coding tools such as *Cursor*, *Replit*, *Bolt*, *V0*, *Lovable*, and *Figma Make*. These tools supported different kinds of generation, from inline code rewriting and rapid UI scaffolding to collaborative hosting and visual prototyping (Table 3). Participants described this stage as a back-and-forth process in which generated outputs were repeatedly adjusted through prompt changes, screenshots, highlighted code, and updated design references.

Best practices for Stage 2 was to avoid broad or under-specified requests. Participants emphasized controlling generation through

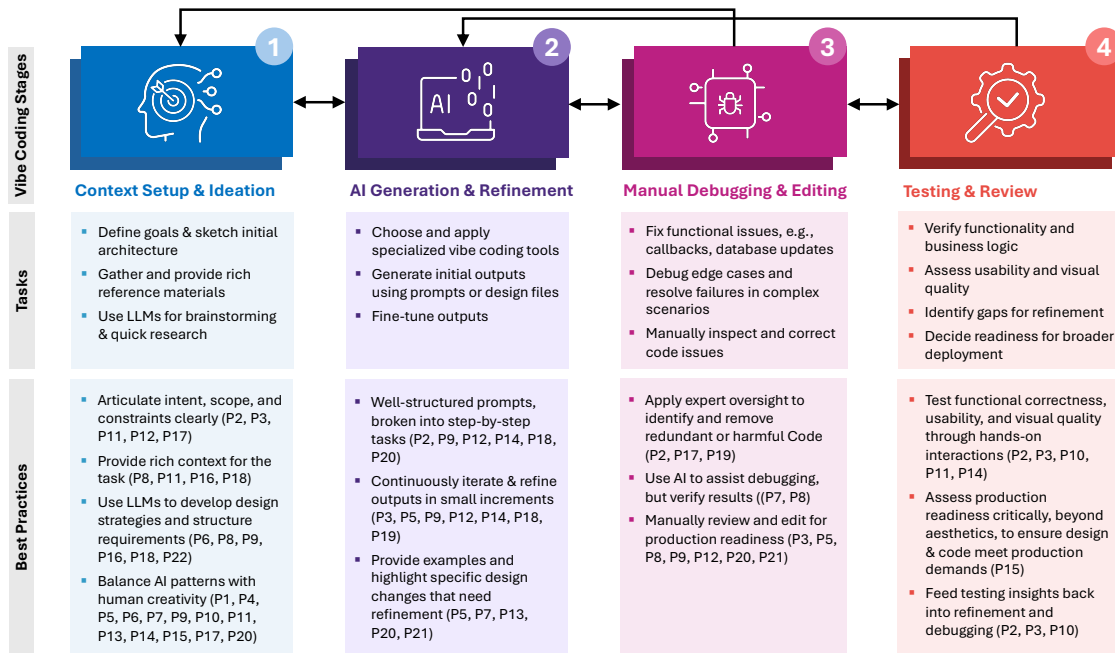
⁷Miro: <https://miro.com>

Table 1: Occupations, participants, and types of vibe coding tasks

Occupation / Role	Participants	Examples of Vibe-Coded Outputs	Typical Tools Used
UX/UI Designers	P4, P5, P6, P9, P10, P11, P12, P21, P22	Interactive prototypes, UI layouts, design refinements	Cursor, Lovable, V0, Bolt, Figma Make
Software Engineers	P2, P3, P18, P20	Frontend/backend code, API integration, debugging	Cursor, Replit, GitHub Copilot
Product Managers	P7, P15	Early prototypes for features and interaction flows	Replit, V0, Lovable
UX Researchers	P8, P16, P17	Testing early design variations, AI-generated scaffolds	Cursor, Lovable, Claude, Gemini
Founders / Hybrid Designer–Developers	P1, P13, P14, P19	Mobile app prototypes, end-to-end flows, MVPs	Cursor, Lovable, Bolt, Replit

Table 2: Work experience and organization size distribution among participants (P1–P22).

Characteristics	Participants (P1–P22)
Work experience	>5 years
	3–5 years
	<3 years
Organization size	Large enterprises (>10,000)
	Large organizations (1,000–10,000)
	Medium organizations (500–1,000)
	Small startups (<50)
	Academic institutions (universities)

**Figure 1: The four interactive stages of vibe coding and the best practices for each stage.**

structured, stepwise prompting and tightly scoped refinement. Rather than asking for sweeping revisions, they tended to make smaller

adjustments in order to preserve coherence and reduce unintended side effects. As one participant explained, “Generate a small code at

a time; if it fails, revert to the previous step. If I change it too much at once, it might hallucinate or just forget part of the request.”[P11] Here, effective prompting was less about writing a single powerful request than about pacing the interaction so that the generated output remained steerable. Participants also highlighted the importance of anchoring prompts to concrete artifacts, such as highlighted code segments, screenshots, or specific UI elements, to guide targeted modifications: “When I need to resize a button, I use targeted prompts by highlighting the code or sending a screenshot to show exactly what I want changed.”[P13]

Stage 3: Manual Debugging and Editing. Even after refinement, AI-assisted outputs required substantial human intervention. Participants frequently corrected callbacks, adjusted backend logic, and inspected code behavior to ensure correctness, particularly when AI-generated solutions failed in edge cases.

Best practices for Stage 3 was to treat AI output as provisional. Participants repeatedly emphasized that code needed to be understood, not merely accepted, “Sometimes Cursor gives you lines of code that don’t do anything at all. That’s why you always need someone on the team who really knows what’s going on in the codebase, to catch these things.”[P17] Participants also described stepping in when AI outputs were brittle or failed under less common conditions: “If AI fails, I step in and understand the code before fixing it. This is especially true for edge cases where the output is brittle.”[P5] While some participants used AI tools to surface potential issues, they remain cautious: “Sometimes I just get lazy and type ‘error’ to let Cursor debug for me. It’s a quick way to see what Cursor thinks is wrong, but I never trust it blindly. I still go in and check the fix myself.”[P7]

Stage 4: Testing and Review. The final stage centered on evaluating whether AI-assisted outputs were functionally, visually, and operationally ready for broader use. Participants described testing prototypes by interacting with them directly, checking not only whether they looked plausible, but whether they behaved correctly across flows and scenarios. This stage often involved validating business logic, interaction behavior, responsiveness, and visual coherence together.

Best practices for Stage 4 was to resist being persuaded by surface quality, “I check whether the component behaves as expected, not just whether it looks right.”[P3] Others stressed that even visually convincing outputs often lacked safeguards required for deployment at scale: “I can see the UI is all right, but putting it out for a big audience? No way... it hasn’t been through any real security checks or code reviews.”[P15] Issues identified during testing frequently triggered returns to earlier stages, creating iterative loops between testing, refinement, and debugging: “When I find an issue during testing, I either adjust the prompt to fix it or switch to manual debugging before re-testing.”[P2]

4.1.2 Theme 1.2: Workflow Changes Compared to Non-AI Tools. Participants described vibe coding not simply as a faster version of existing workflows, but as a reordering of what happens first, what can be deferred, and who gets to participate. In contrast to more traditional product design and development workflows, which often begin with polished mockups and role-specific handoffs, vibe coding encouraged earlier movement into functional artifacts (Table 4). Several participants explained that they no longer needed to begin

with pixel-perfect mockups in tools like Figma, because they could move directly into interactive prototypes and test flows sooner (e.g., P1, P4, P5, P11, P12).

This changed what counted as progress. Rather than prioritizing visual polish at the outset, participants often treated functionality, flow, and interaction logic as the first things to validate: “...at this stage you just want to test the idea and see if the flow works. The exact visual details don’t matter yet. Aesthetics aren’t the priority. I just want to make sure the functionality is there.”[P8] Collaboration dynamics also changed. In conventional workflows, sketches or wireframes moved through multiple rounds of feedback, which participants described as slow and fragmented: “In [non-AI] Figma, you get overloaded with comments from designers, developers, everyone leaves notes.”[P15] In contrast, vibe coding enabled individuals to quickly produce and adjust working prototypes without heavy back-and-forth. Still, participants emphasized that Figma remains the tool of choice for large, production-level projects requiring pixel-perfect visuals and clear communication across stakeholders.

4.2 Theme 2: Vibe Coding Benefits

Theme 2 captures how participants experienced productivity and creativity benefits in vibe coding workflows. These benefits observed in our study, including faster prototyping, reduced cognitive load, and support for ideation, are well documented in prior work on AI-assisted programming and design (e.g., [3, 90]). Our findings extend this line of work by showing how these benefits emerge specifically through the integration of ideation, generation, and prototyping into a continuous workflow.

Participants described faster transitions from abstract ideas to working prototypes, enabling early testing without producing pixel-perfect mockups. “From ideas in my head to ideas I can see. It’s like having a thought partner to explore, learn, and debug with and I can get feedback almost instantly.”[P9] Rather than focusing only on speed, participants framed these benefits as a shift in how work is distributed between human and AI. AI tools were used to track details across evolving prototypes and support iterative refinement, reducing the need to manually manage intermediate steps. “Our brains can’t remember every detail of the code... But AI can keep track of all the details and recall them instantly.”[P18] This redistribution of effort also enabled more direct engagement with prototyping, particularly for designers and less-technical team members, who could independently generate functional artifacts without relying on separate implementation stages.

Participants also described how vibe coding supports creative exploration by lowering the barrier to starting and iterating on ideas. Rather than working from a blank screen, they could quickly generate and refine multiple directions. “I can easily overcome the whiteboard fear.”[P13] Importantly, creativity was not attributed solely to AI-generated ideas, but to the ability to explore, compare, and iteratively refine outputs within the same workflow. “AI follows first principles and generates ideas through probabilistic distributions... Compared to human creativity, which is more about convergence thinking, AI works differently.”[P6] Some participants also noted that the pacing of interaction with AI introduced brief pauses that supported reflection and creative reset during otherwise fast-paced workflows. “When I am waiting for vibe coding tools to

Table 3: Participant-reported experiences with vibe coding tools.

Tool	Description	Illustrative Quote
Cursor	Valued for inline debugging and context-aware rewrites, enabling developers to refine code without leaving the editor.	P12: “Cursor is great because it allows me to debug and tweak code inside the editor without switching context.”
Replit	Supported collaborative experimentation and fast testing with built-in hosting.	P9: “Replit is better when I need to work collaboratively or test something quickly online.”
Bolt	Provided near-immediate visual previews from natural language or Figma designs, supporting rapid UI iteration.	P5: “I use Bolt when I want more visual feedback and test different design ideas. It gives me instant UI previews. It feels like working with a visual playground.”
V0	Generated full web page layouts quickly and integrated with Figma, though limited for complex applications.	P11: “V0 is good for landing pages and quick UI drafts. It’s super fast but not great for more complex apps.”
Lovable	Offered text-to-app flows with backend logic and supported iterative design with prompt-based aesthetic adjustments.	P10: “Lovable helps me iterate design ideas with prompts and lets me adjust the aesthetics pretty well.”
Figma Make	Enabled rapid generation of high-fidelity UI mockups directly within Figma, supporting early concept communication and smooth transition into manual refinement.	P21: “With Figma Make, I can quickly generate a relatively high-fidelity prototype to demonstrate my concept, and then export it to Figma to manually tweak and refine the design.”

Table 4: Comparison between Vibe Coding and Non-AI Tools for Product Design and Development (as reported by participants).

Aspects	Vibe Coding Tools	Non-AI Tools	Participants
Primary Focus	Functionality-oriented; rapid prototyping and iteration; working prototypes over polished visuals	Visual-first; pixel-perfect mockups and high-fidelity UI design	P1, P3, P8, P11, P21, P22
Workflow Speed	Very fast; skips manual mockups and jumps to interactive prototypes	Slower; requires manual layout creation and refinement	P1, P3, P6, P10, P21
Collaboration	Best suited for personal or small-team projects; reduced coordination overhead	Optimized for team collaboration with multiple stakeholders	P7, P12, P15, P21
Use Case Stage	Primarily early-stage ideation, concept testing, experimentation	Primarily mid-to-late stage design; refining and finalizing visuals for production	P8, P14, P19
Tool Strengths	AI-assisted code generation; rapid interactive prototyping	Strong visual design tools; clear communication of design intent	P1, P5, P7, P9, P13
Considerations	Less focus on refined aesthetics; may need additional design polishing	Slower iteration speed within small teams; can be overloaded with comments/feedback	P3, P7, P8, P12, P16, P18, P21

generate the results, that pause actually gives me a moment to breathe and reset. That space helps me be creative.” [P14]

4.3 Theme 3: Vibe Coding Challenges

Rather than reiterating well-known limitations of AI-assisted programming (e.g., hallucinations or unreliable outputs) [3, 90], our findings highlight how these issues manifest as workflow-level breakdowns in vibe coding, particularly in integration, coordination, and scaling contexts. Participants described that while vibe coding works effectively for early-stage prototyping, breakdowns emerge as projects move toward production, where reliability, integration, and coordination become critical.

4.3.1 Theme 3.1: Integration and Scaling Breakdowns. Participants emphasized that vibe coding was rarely used for production-level tasks, particularly when projects involved complex back-end systems, security-sensitive operations, or large-scale deployments. While AI tools performed well for rapid front-end prototyping, they could slow down iteration after reaching the Minimum Viable Product (MVP) stage, especially for experienced practitioners seeking fine-grained control and customization. *“It works fine for quick UI ideas, but backend work or anything complex just doesn’t hold up. Developers are still needed for security, optimization, and scaling.” [P8]*

In addition to generation errors (e.g., difficulties meeting exact UI specifications), participants described integration as the

primary bottleneck. Backend connections, APIs, and deployment pipelines were frequently unreliable, even when inputs were correct. *“I couldn’t get it to connect to the OpenAI API, even with the right key.”*[P11]

System-level integration further introduced new coordination challenges. Participants reported conflicts across tools, where outputs generated by different AI systems interfered with each other, requiring manual reconciliation. *“Plugging AI code into my system is tricky. Databases are the worst. The UI might look fine, but the database connection almost always breaks with wrong queries. I end up fixing it manually. And when I use different tools like Claude in Cursor and GitHub Copilot, they don’t talk to each other; one undoes or misreads what the other did, and I’m stuck deciding which to trust.”*[P1] These breakdowns were amplified during deployment. While local prototypes often functioned as expected, cloud deployment and scaling introduced instability and failure points. *“It works locally, but cloud deployment just keeps breaking.”*[P9]

Participants also noted that AI-generated codebases tend to grow in unstructured ways over time, accumulating redundant or unnecessary code that degrades maintainability and obscures system logic, particularly in larger projects. *“It can balloon into something unmanageable. The AI just keeps adding stuff, and before you know it, the code’s bloated and a pain to work with.”*[P13] Finally, compared to traditional design tools like non-AI Figma, AI-generated layouts lacked pixel-level fidelity and adaptability across screen sizes, requiring manual correction.

4.3.2 Theme 3.2: Breakdowns in Incremental Refinement and Traceability. Participants described difficulties in making small, precise changes to AI-generated code. Even minor UI adjustments could cause cascading breakages, requiring repeated adjustments to bring the output back to what was intended. *“It struggles with exact positioning or maintaining responsiveness across devices.”*[P12] Debugging complex behaviors, such as front-end interactions, asynchronous logic, or database queries, were also time-consuming, with AI agents sometimes entering repetitive correction loops, without resolving the underlying issue. *“It forgets what the outcome was supposed to be and my earlier instructions, so I have to keep pulling it back on track.”*[P15]

Participants further emphasized limited traceability in AI-assisted workflows. Debugging lacked clear attribution, making it difficult to identify where issues stemmed from. *“Debugging is difficult and opaque. You can’t always tell if the problem came from my prompt, the AI’s own logic, or some earlier change it made to the code.”*[P10] A further barrier was the lack of robust version control. While some tools like GitHub and Lovable had versioning features, others required manual workarounds such as copying AI-generated code into temporary documents for iteration. Participants also criticized the absence of transparent, human-readable histories of AI actions, making it difficult to trace errors or understand why certain decisions were made. *“We need detailed histories of AI actions to improve reliability and responsibility.”*[P17] Finally, participants noted that AI outputs often over-generated content, introducing unintended components or features that required additional cleanup and coordination effort. *“Sometimes it just spits out way too much, giving random components I never asked for, and then I have to clean it all up.”*[P17]

4.3.3 Theme 3.3 Pseudo-Productivity, Shallow Creativity, and Over-Reliance. Some participants warned of a “productivity paradox”, where speed gains encouraged taking on more work, increasing pressure without necessarily improving outcomes. *“I get more projects, they try to beat me more, although it gives me pleasure to see my potential.”*[P19] Others noted that AI’s convenience could lead to shallow creativity, with teams settling for first-draft outputs rather than exploring deeper, more original solutions. *“It’s easy to stop at the easy answer instead of pushing for depth. You get something decent from the AI, and it feels good enough to move on. But that can make you skip the harder, messier exploration where the really creative breakthroughs often happen.”*[P21]

Participants also described compulsive iteration, almost feel like addiction, where the ease and immediacy of vibe coding made it difficult to disengage, even when progress felt unsatisfying. *“It’s easy to spend a whole day just vibe coding, trying small changes over and over, but still not feel happy with the result at the end.”*[P7]

Concerns about over-reliance were also widespread. While vibe coding reduced dependence on developers for basic tasks, it risked replacing this with dependency on AI, potentially eroding foundational coding and design skills, particularly for new practitioners. Participants stressed the need for intentional pauses, human review, and diverse design inputs to avoid homogenization. *“I’m definitely more independent from developers now, but I do rely on AI a lot. My worry is that for newer people, they might skip learning the basics and just trust whatever the AI gives them. If we’re not careful, everything will start looking and working the same. I think it’s important to step back sometimes, get human feedback, and keep different design voices alive.”*[P14]

4.3.4 Theme 3.4 Breakdown of Contextual Alignment at Scale. Participants emphasized difficulties in maintaining alignment with project-specific context, particularly as projects grew in size and complexity. AI systems struggled to remain consistent with prior decisions, design systems, and domain-specific constraints over time. *“It forgets things really quickly...once the project gets big, it loses track of the context and slows down. It also doesn’t really know our product-specific rules.”*[P6] These breakdowns were especially visible when working with structured inputs such as design systems or component libraries. While AI could generate visually similar outputs from screenshots, it often failed to respect underlying design logic, requiring manual correction to maintain consistency. *“It’s fine if I just send a screenshot, but once I try to go beyond that, like using design libraries, it doesn’t really follow them. And as the project gets bigger, it starts forgetting context and losing alignment.”*[P22]

4.4 Theme 4: AI Trust, Ownership, and Accountability

Theme 4 examines how trust in AI tools within product teams is situational rather than absolute, shaped by task complexity, project maturity, and domain-specific stakes. Participants described shifting notions of ownership, authorship, and accountability as AI became integrated into design and development workflows.

4.4.1 Theme 4.1 AI Trust Boundaries and Fit for Use. Participants reported that AI tools were most trusted for ideation and early-stage prototyping, where architectural soundness was less critical.

These scenarios typically involved low-stakes or personal projects and work that followed well-established patterns, such as common UI layouts or standard coding structures. *“Simple UI tasks, generating early-stage prototypes. That’s where it shines.”* [P10] Trust declined sharply in production contexts, especially for backend integration (e.g., setting up database connections, authentication systems, or API gateways), performance-critical systems (e.g., real-time analytics dashboards or high-traffic e-commerce platforms), and complex business logic (e.g., financial transaction validation, medical image processing, or multi-step workflow automation). Multi-stakeholder workflows and backend functionalities such as databases or APIs were also seen as poor fits, particularly when coordination across designers, developers, and Quality Assurance (QA) teams was required. *“I just don’t trust AI when it comes to integrating with production. It’s fine for prototypes or quick UI work, but once you’re dealing with live systems, real data, large audience, it can break in ways you don’t see coming.”* [P1]

4.4.2 Theme 4.2 Evolving and Calibrated Trust. Trust in vibe coding tools evolved through use. Participants described learning to calibrate when and how to rely on AI outputs. Confidence increased in bounded scenarios, such as scaffolding code, repetitive tasks, or well-understood patterns, while skepticism remained in complex or high-stakes situations. *“I trust AI more for small personal tasks now. I’ve seen hallucination issues improve, and I like how Cursor sometimes flags potential mistakes in its output.”* [P20] This calibration was also shaped by how AI systems communicated uncertainty. Participants reported greater willingness to rely on AI when its limitations were more visible or interpretable, enabling them to position it as a collaborator rather than an authority. *“It’s not that I think AI is perfect now. It’s not, but it feels more upfront about what it knows and doesn’t know. That makes me more willing to trust it with starting points and drafts, even if I’ll still do a pass myself.”* [P14]

4.4.3 Theme 4.3 Workflow Constraints from Security and Compliance. Security and compliance concerns shaped how and where vibe coding could be applied, particularly in collaborative and enterprise contexts. These concerns directly constrained usage, limiting AI involvement in tasks that required handling sensitive, proprietary, or regulated data. *“Due to compliance issues, I avoid uploading company data into generative AI tools.”* [P4] Concerns spanned both technical and procedural dimensions. Technically, participants mentioned risks such as prompt data leakage, insecure cloud-based workflows, and phishing attacks targeting teams unfamiliar with secure AI practices. Poor security hygiene, such as sharing credentials in prompts, was repeatedly flagged. *“You wouldn’t believe how many people paste API keys or production database credentials into a prompt. That’s basically handing them to whoever runs the model.”* [P7] Procedurally, enterprise policies often restricted AI tool use unless vendors provided explicit guarantees regarding data storage, retention, and non-training clauses. *“Our legal team won’t let us use certain AI tools unless there’s a signed agreement about where the data goes, how long it’s stored, and a guarantee it’s not used for training.”* [P12]

4.4.4 Theme 4.4 Ownership and Accountability. While participants valued AI for accelerating execution and surfacing alternatives, final responsibility consistently remained human. *“I’m the navigator; AI*

is the intern.” [P11] Many participants tied ownership to problem framing rather than execution: those who defined goals, constraints, and decisions retained authorship, even when AI handled much of the implementation. *“The work is still mine. I provide the ideas, guide AI step by step, and AI realizes them.”* [P7] At the same time, some reported feeling less connected to outputs when AI generated large portions of them, especially in tasks involving minimal manual refinement. *“When AI gives me a near-finished draft and I’ve barely done anything beyond approving it, it feels like someone else’s work. I don’t feel as connected to it as I do when I’ve been hands-on from the start.”* [P21] This raised concerns about the erosion of personal creative signature and the ability to stand behind the work with full confidence.

Participants agreed that clear attribution and traceability were essential, both to acknowledge human expertise and to identify which parts of the output could be trusted or needed review. *“You’ve got to be clear about which parts came from you and which parts came from the AI, and keep track of who signed off on what. Otherwise, it’s a nightmare to figure out who’s actually responsible or who deserves the credit.”* [P17] Copyright and intellectual property concerns were also prominent. Participants expressed unease about using AI tools trained on copyrighted works without permission. Several participants proposed adopting AI crediting guidelines similar to citation practices in academic citation norms, including prompt documentation, edit histories, and explicit disclosure of AI involvement. *“We should treat AI like a co-author...record the prompts, note where AI stepped in, and document the edits...anyone looking at the work later can see exactly what was human and what was AI.”* [P17]

4.5 Theme 5: Human-Human and Human-AI Collaboration

Theme 5 examines how vibe coding tools reshape collaboration within product teams by accelerating handoffs, reducing coordination bottlenecks, and enabling professionals to work beyond traditional role boundaries. Participants described shifts not only in how work is produced, but also in how responsibilities, authority, and collaboration are negotiated across roles and organizational contexts.

4.5.1 Theme 5.1 AI-Driven Role Shifts and Collaboration Dynamics. Participants reported that AI reduced early-stage coordination overhead by enabling individuals to explore ideas independently before engaging others. This allowed designers, engineers, and product managers to iterate on prototypes without relying on sequential handoffs. *“It speeds up handoff and iteration across different roles. What used to take weeks can now be done in just a few days.”* [P3] This shift also blurred traditional role boundaries. Designers increasingly engaged in code generation and prototyping, while engineers shifted toward reviewing, refining, and integrating AI-generated outputs instead of building everything from scratch. *“Live, interactive prototypes inspire engineers more and make it easier to get buy-in, and prototyping is now accessible to everyone, not just designers or engineers.”* [P5] Collaboration became more output-centered, with teams jointly evaluating AI-generated artifacts rather than dividing work strictly by role. This reduced reliance on technical specialists for early-stage exploration and enabled more fluid, hybrid forms of work. *“We [designers] don’t need to wait for working prototypes*

to be coded. AI acts as a copilot designer or an associate PM [product manager], making communication with engineers much easier.” [P11]

AI also enabled professionals to work beyond traditional silos. Designers conduct research and prototyped independently. Developers reduced manual scaffolding and shifted toward product-led thinking and oversight. Participants anticipated more hybrid roles (e.g., UX engineers) and leaner teams. These shifts were accompanied by evolving skill expectations, including coding fluency for designers and product managers, as well as meta-prompting and evaluation skills. Existing practices, such as creating overly detailed manual mockups, may become less central. “Designers will learn more about code, and developers will start contributing to design.” [P4] AI also increased individual autonomy. Participants reported greater flexibility and a stronger sense of ownership when they could independently explore and test ideas, reducing the need for early-stage coordination and interpersonal negotiation. “I like the flexibility and ownership of doing both design and code. I avoid a lot of arguments.” [P1] However, these changes introduced new tensions around ownership. In some teams, particularly larger organizations, AI-generated outputs were treated as shared drafts that anyone could modify, diluting individual ownership. “With these AI drafts, anyone can jump in and tweak things. It’s not really ‘my design’ anymore, just something we all poke at.” [P7] Senior developers described yet another pattern, where foundational expertise strengthened ownership: “Our knowledge [in coding] helped the team spot AI mistakes and steer design decisions.” [P18]

4.5.2 Theme 5.2 Junior versus Senior Roles: Job Security and Cultural Tensions. Automation of entry-level tasks raised concerns about the erosion of junior roles. Participants worried that juniors could be relegated to prompt crafting and tool operation rather than developing foundational skills. “Junior designers may end up as tool users or prompt crafters rather than concept creators.” [P4] At the same time, senior developers increasingly assumed oversight roles, relying on AI for execution while guiding overall direction. “The demand for entry-level positions is decreasing, while senior roles are shifting toward decision-making.” [P7]

These shifts also introduced social tensions. Some participants reported feeling judged when using AI tools, particularly in environments where AI-generated outputs were perceived as lower quality or less authentic. “I feel a bit ashamed using AI in front of skeptical senior engineers.” [P20] Participants also pointed to mismatches between AI-generated outputs and established engineering standards. “Senior developers value clean, human-readable code. AI-generated code is often verbose and looks ‘junior’.” [P20]

4.5.3 Theme 5.3 Evolving Roles of AI in Human Teams. Participants framed AI not as a replacement for human collaboration, but as a reconfigurer of roles and responsibilities. Many treated AI as analogous to a “junior developer” [P3], delegating standardized and well-defined tasks, such as generating boilerplate code or prototypes. “This frees engineers and designers to focus on higher-level judgment and decision-making, while workflows evolve to integrate AI seamlessly.” [P3] AI was also described as an “amplifier of human intent” [P20], a thinking partner that could detect when users were stuck, teach prompt vocabulary, and explain code syntax on request. For some, this meant developing new practices of meta-coding: learning how to instruct an AI coder rather than writing

every line themselves. “I stay the conceptual lead and let automation handle execution.” [P20] Some noted AI’s educational potential, particularly for newcomers. Tools like Cursor not only generated functional code but also explained its logic, helping participants learn syntax, debug issues, and gradually acquire coding fluency. This was especially valuable for “onboarding into unfamiliar or legacy codebases” [P20].

Participants also reflected on their relational stance toward AI. While many treated it pragmatically as a tool, some admitted to playfully anthropomorphizing it. For example, naming agents for easier attribution “We name our AI agents so it’s easier to know which one produced what output.” [P17], or debating whether politeness improved results “Some people say if you’re polite to AI, it gives you better results.” [P7] Overall, participants envisioned AI evolving from an assistive tool into a more autonomous teammate, one that aligns with user intent, explains its reasoning, and integrates smoothly into workflows. Yet, participants cautioned that greater autonomy could risk overriding instead of supporting human decision-making. “Right now it’s an assistant, but I can see it becoming more like an autonomous teammate...It should also help me think better, not think for me.” [P10]

4.5.4 Theme 5.4 Organizational Adoption and Barriers. While participants valued the collaborative potential of vibe coding tools, adoption within organizations was shaped by existing structures and constraints. In larger organizations, established workflows and role boundaries often limited how AI tools could be integrated, particularly when responsibilities became less clearly defined. “In large organizations, which typically have stricter policies and more rigid team structures, vibe coding is difficult to embed, can blur responsibilities, and may conflict with performance expectations.” [P9] Change management and breaking long-standing professional routines were cited as critical to adoption. “The tech is there, but the challenge is changing habits.” [P8] Some pointed out that current tools lacked safeguards, such as automatic verification layers, to prevent flawed AI-generated code from reaching production. “There’s no firewall, nothing to stop bad AI code before it hits the system.” [P17]

To make these role-specific patterns explicit, Table 5 synthesizes how participants across roles differed in their use of vibe coding tools, perceived benefits, experienced challenges, and trust calibration.

5 Discussion

5.1 Emerging Intention-to-UX Ecosystems across Organizational Practices

Vibe coding extends prior work on how AI reshapes coding and product design and development workflows [45, 52, 57]. Whereas prior research showed that AI tools shift effort from writing code to validation and debugging, our findings suggest that vibe coding tools also reconfigure the broader workflow (RQ1), shifting it from pixel-first to function/problem-first, and from ideation toward validation (Sec.4.1.2). Consistent with earlier work [28, 52, 55, 56], participants highlighted how vibe coding supports rapid iteration and conversational exchanges that accelerate idea generation. To that end, the most valuable aspects of product design work shifts away from execution toward higher-level instruction and decision

Table 5: Role-specific differences in the use, benefits, and challenges of vibe coding tools.

Role	Primary Use of AI	Key Benefits	Key Challenges	Trust & Stakes
UX/UI Designers	UI generation, prototyping, early ideation, design refinement	Increased autonomy; reduced reliance on developers; faster transition from ideas to interactive prototypes; supports creativity and exploration	Limited precision control; difficulty with responsive layouts; weaker adherence to design systems; risk of shallow design exploration	High trust in early-stage design; low trust in production; moderate risk (design quality, UX coherence)
Software Engineers	Code generation, debugging, back-end/frontend implementation, integration	Reduced manual scaffolding; faster iteration; ability to focus on higher-level architecture and decision-making	Integration failures (APIs, databases); debugging opacity; code bloat; lack of traceability; unreliable outputs at scale	Low trust in production-critical systems; high responsibility for correctness, security, and scalability
Product Managers	Feature prototyping, interaction flows, idea validation, coordination	Faster validation of ideas; improved communication with designers and engineers; reduced dependency on technical teams for early exploration	Limited technical depth; difficulty assessing correctness; reliance on others for validation; ambiguity in ownership	Moderate trust in early-stage outputs; high risk in misinformed decisions if outputs are incorrect
UX Researchers	Testing design variations, generating scenarios, exploratory prototyping	Rapid generation of alternatives; supports hypothesis testing; lowers barrier to experimentation	Limited contextual understanding; inconsistency with research protocols; difficulty validating AI-generated outputs	Moderate trust for exploration; low trust for rigorous evaluation or data-sensitive contexts
Founders/Hybrid Designer-Developers	End-to-end prototyping, MVP building, product iteration	High autonomy; ability to execute full product cycles independently; reduced team dependency; faster time-to-market	Scalability limitations; technical debt; over-reliance; integration breakdowns in complex systems	High trust in early-stage building; cautious trust in scaling; high personal stake in outcomes

making, largely centering on articulating intent and orchestrating AI-mediated processes, which aligns with findings on AI non-use by designers [13]. Yet, our study highlights bottlenecks in debugging and editing, where extensive human oversight remains critical, especially for non-programmers on product teams. At the organizational level, these shifts raise questions about how teams should redistribute expertise and what new roles may emerge for prompt engineering, UX design, and AI oversight [37, 52]. Participants anticipated leaner product teams where boundaries blur across designers, developers, and product managers and owners, alongside new roles such as prompt librarians” and prompt QA (Quality Assurance)” (Sec. 4.5.1).

Our findings point to the emergence of an “intention-to-UX ecosystem” (Sec. 4.1.1 & Sec. 4.4.4), where AI mediates between human goals and executable outputs [61]. Rather than a linear workflow, this ecosystem consists of interconnected components including human actors (e.g., designers, developers, product managers), prompts as representations of intent, AI systems that generate code and interface elements, generated artifacts such as prototypes and code, and evaluation and debugging processes that feed back into subsequent prompts and design decisions. These components interact through iterative feedback loops in which outputs reshape prompts, design decisions, and even initial project goals, making development an ongoing process of intent translation and refinement rather than a sequence of predefined steps. The four-stage workflow we identify (Fig. 1) can be understood as the operational structure of this ecosystem, with each stage representing a different configuration of interactions between human intent, AI generation, debugging, and evaluation, and with transitions between stages forming iterative feedback loops that refine both the product and

the underlying intent. Participants described AI as an amplifier of human intent, shifting focus from pixel-perfection to functionality, orchestration, and strategic judgment. While the ease of prompting creates opportunities for code and app democratization [84], it also creates challenges since prompt engineering still remains difficult for non-experts [14, 83, 99]. Within this ecosystem, prompting functions as a critical mediation layer between intent and output, where difficulties in prompting represent bottlenecks that slow iteration and increase reliance on expert users. To that end, we uncover practical prompting strategies that can aid onboarding and adoption, and we encourage tool creators to integrate such guidance directly into their systems (Sec. 4.1.1). At the same time, the acceleration of prototype creation risks privileging speed, what we see as intending the right design”, over reflective exploration of values and alternatives through “designing the right intention” (cf., [89]). These dynamics can be understood as emergent properties of the intention-to-UX ecosystem, emerging from rapid iteration cycles, AI-generated suggestions, and organizational pressures toward efficiency. Without this layer, product and UX design cycles risk premature convergence and homogenization (Sec. 4.3.3), a recurring risk when AI tools are blindly adopted for product design and development work (cf., GenAI perceptions in UX design [52]).

5.2 AI Over-reliance, Trust, and Collaboration in Product Team Practices

It is clear that there is a global shift toward vibe coding and related agentic engineering practices (cf., [72]) taking place, carrying both risks and opportunities (RQ2). Not surprisingly, a major risk this carries is job replacement and worker deskilling from over-reliance on GenAI output [75]. For the first, despite that we find that fears

of job replacement, especially among junior product designers (cf., [52]), are reduced, at least current United States data suggests otherwise. However, rather than focusing on macro-level employment outcomes, our findings primarily highlight how these concerns manifest within day-to-day collaboration, trust, and responsibility dynamics in product teams. Second, even if junior professionals are shielded from unemployment risks, vibe coding introduces new barriers to collaboration. This is especially pronounced when asymmetry between skill sets is coupled with general (perceived) unreliability of AI generated code. This can distort social perceptions of the users and abusers of AI, which can include reputational harms (cf., [37]) and junior professionals having to navigate such social stigma in such work cultures (Sec 4.5.2).

In particular, our findings suggest new asymmetries where senior developers are more likely to distrust AI-generated contributions, while junior practitioners rely on them more but may face reputational risks for doing so. While there may be clear perceived benefits, the excess usage of AI tools lends itself to proneness of over-delivery, cognitive overload (cf., visually impaired users [30]), and even so-called ‘agent psychosis’ [70] potentially exacerbated by addictive patterns associated with vibe coding (cf., Sec 4.3.3). More broadly, participants described how heavy reliance on AI-generated code could result in unintended features, lower code quality, and difficulties collaborating on shared codebases, which unsurprisingly leads users for example to decline AI generated merge requests [81]. This leads to a worrisome development: on the one hand, experienced product team members are less likely to trust AI-generated code (see Sec 4.5.2), where recent studies show accepting <44% of AI-generated code [4], and in parallel, we are observing a shift where stakeholders are shifting their over-reliance on expert developers toward over-relying on AI systems instead (see Sec 4.3.3). This creates a paradoxical shift in responsibility, where trust moves away from individual developers toward AI systems, even while team members remain skeptical of AI-generated outputs.

While one can argue that this shift ultimately serves to increase access and democratizes coding, this risks technical deskilling, responsibility gaps, and lowering code quality [40, 61, 75]. This is especially so given difficulties in controlling the AI tools to generate the right specifications and desired output [53]. This raises a tangential yet critical point: not only can AI generated code create conflicts among technically-gearred members in teams (see Sec 4.5.1), but between AI models themselves when they offer conflicting suggestions. Such situations further complicate collaboration and trust in AI-assisted design and development. This can create confusion and uncertainty, especially among more junior professionals. This naturally leads to users exhibiting distrust in AI-generated code, and showing preferences for independent learning [1], a critical point within AI literacy to which we turn to next.

5.3 Revisiting Creativity and Agency in Vibe Coding for Product Teams

A recurrent finding from product team members was the so-called “productivity paradox” (Sec 4.3.3), where the speed gains from using AI resulted in individuals taking in more work. Given the increased workload, an obvious caveat highlighted was that this may lead to shallow creativity, with teams more quickly settling for first-draft

or ‘good enough’ outputs, risking speed-quality trade-offs [27]. This enhanced ability to work faster, in an agile manner (cf., [17]), easily lends itself to the promise of substantial productivity gains from GenAI coding [18]. So what then remains of human effort, agency and creativity in this future of product design and development work? Earlier work by Li et al. [52] and Cha and Wong [13] found for example that human agency is a recurring theme among UX practitioners, where humans bring contextual understanding, ethical judgment, and accountability across design tasks, and prioritize maintaining control over their processes. With such qualities, they should be the ones to lead complex design-related decision-making. At the same time, there is a risk of creativity exhaustion [52] and generally cognitive overload, which can result not only in homogenized design within product teams, but such speed gains can exert a high amount of pressure on human creators to review such output.

Nevertheless, it is worth asking to what extent this enhanced ability for GenAI divergent exploration of concepts (Sec 4.3.1) has merit. Bellemare-Pepin et al. [5] found that LLMs can surpass average human performance on the Divergent Association Task [65], a psychological test designed to measure divergent thinking. Moreover, LLMs are approaching human-level creative writing abilities, though still behind in performance compared with highly creative people. Even if GenAI surpasses people in such divergent thinking, product team members emphasized that humans still maintained ‘strategic oversight and conceptual leadership’ in these emerging role shifts (Sec 4.5.1), echoing earlier findings that humans are and will remain the final arbiters of human-AI alignment [52], within and beyond knowledge and creative work. While GenAI can certainly enhance creative output during assisted UX-related tasks and may lead to increased independence and human autonomy when working in product teams (Sec 4.2 & Sec 4.5.1), prolonged reliance on such tools may risk diminishing individuals’ creative performance [49] – this is a cause for concern over the long-term impact on the flourishing and nurture of human creativity (RQ2). It then comes as no surprise that professionals advocate for resisting homogenization through deep exploration (Sec 4.3.3) and stressing the importance of critical thinking and creative restraint through intentionally crafting learning moments that may foster artistic variation in product design work.

5.4 Perceived Ownership, AI Disclosures, and Responsibility in Vibe Coding Software Products

A recurrent finding concerns perceptions of ownership and accountability in vibe coding workflows (Sec. 4.4.4). Here, responsibility was consistently seen as a human duty, where ownership was tied to ideation rather than execution. Essentially, the person who intended the design was viewed as the true “owner”, even if AI executed much of the work. This suggests a shift in how ownership is defined in collaborative product development workflows, moving from implementation and code authorship toward intention-setting, decision-making, and oversight. This is a shift from the early debates in AI art and its (harmful) impact on artists, where prompting a model does not mean the prompter is the artist (cf., [42]). In vibe coding contexts, however, participants did attribute ownership to those who defined the intent and guided the process, indicating a

reconfiguration of authorship and responsibility in collaborative human–AI development workflows. We find that while earlier concerns about authorship erosion and homogenization [41, 42, 52] still hold, they now manifest differently: instead of product designs converging on similar aesthetics, homogenization occurs in the kinds of solutions AI tends to suggest (e.g., repeated interaction flows). Furthermore, ownership concerns also shifted: the risk is less about losing credit for ideas and more about losing control over execution, since AI often generated large parts of the work. This reinforces the idea that control over intent and direction, rather than execution itself, becomes the central locus of ownership and responsibility in vibe coding workflows. At the same time, AI disclosure practices (Sec. 4.4.4) remain inadequate (even if mandated by legislation; cf., [24]). Labels such as “AI-assisted” can easily obscure the multi-step nature of collaboration (cf., [50]); especially with agentic AI tools executing hundreds of actions (in parallel), where only some are accepted by team members. When considering the speed-driven nature of collaborative product design and development environments, it is not unexpected that vibe coding and role fluidity will produce uneven recognition of contributions. Here, hidden labor like expert prompting or debugging model output may get undervalued while those who merged the final version receive most of the credit. This further complicates traditional notions of authorship, as contribution becomes distributed across intention-setting, prompting, evaluation, debugging, and final integration rather than concentrated in code writing alone. Such uneven visibility of contributions may also create tensions within and across teams, for example when team members frame similar problems or pursue similar solutions but only some use AI tools to execute tasks faster. As such, it is clear that attribution mechanisms such as design logs or attribution frameworks [38] would be necessary to help surface this hidden work transparently, and ultimately ensure fairness in process and output.

Furthermore, disclosures and ownership become even more complex when considering the evolving roles of humans and AI (Sec. 4.5.3). When models are playfully anthropomorphized, such as thanking the AI or giving it nicknames, it humanizes interaction and risks implying the system was acting with real intent. This makes it crucial that disclosures highlight human oversight and specify roles clearly so that agency is not blurred or overstated [51]. This essentially ties to the responsibility gap [59], where responsibility for decision-making is distributed among multiple participants, including AI systems. This becomes critical when considering organization security and privacy (Sec. 4.4.3). For example, consider unsafe practices such as pasting sensitive data into prompts, which can result in incidents like the *Replit* case where AI-generated changes reached production with minimal review [31]. Taken together (RQ2), our findings caution that with all the vibe coding exuberance and its purported benefits come real risks of (organizational) responsibility and accountability (cf., [68]), which only increases as AI models are increasingly treated as part and parcel of product teams.

5.5 Study Limitations and Future Work

We note that our study provides analysis and interpretation of only a snapshot in time of vibe coding practices, during a time with rapid

AI development. AI tools and workflows emerge, fall out, or adapt quickly. While this may benefit from more longitudinal research, we believe that our findings on product team members’ practices for individuals and organizations are nevertheless relevant in such snapshots, if only to help us navigate uncertain futures marked by ever-more capable AI models. Second, our investigation opted for a breadth-first approach, mapping a wide range of perspectives across different organizational contexts. This necessarily came at the expense of digging deeper into the nuances of vibe coding workflows. As such, future work would benefit from building on this foundation with more in-depth studies of particular tools, workflows, or team practices.

Finally, our participant sample broadly encompasses product design and development, which reflects the increasingly fluid boundaries of such product teams and their practices. This spans multiple roles, including UX designers, front-end engineers, and product managers and owners. While on the one hand such diversity enriched our data, it also complicates how a given team member’s precise role is now defined (e.g., hybrid designer/developer), and whether an individual does or continues to self-identify with such as label. We see this ambiguity not just as a challenge but as a signal of shifting professional identities in the era of widespread LLM use and (anticipated and attempted) skill democratization (cf., [75]). Similarly, discourse on vibe coding recently distinguishes this from agentic coding/engineering [72]), however we find this is itself unclear given the high overlap within and across workflows. In this regard, we believe this provides a promising avenue for future work: to closely examine how responsibilities for intent-setting, execution, and evaluation are redistributed across and within product-focused roles, how the role of largely autonomous agents will embed in workflows when they run 24/7, and how these changes reshape collaboration in blended human–AI teams (cf., [16]).

6 Conclusion

Our study examined how “vibe coding” is reshaping product design and development work by collapsing ideation, prototyping, and implementation into a conversational, intent driven workflow. Drawing on interviews with 22 product team members across enterprises, startups, and academia, we mapped a four-stage vibe coding workflow comprising ideation, AI generation, debugging, and review. We find that while vibe coding accelerates iteration, supports creativity, and lowers barriers to participation, professionals also reported challenges of code unreliability, integration, and AI over-reliance. Furthermore, We find that tensions emerge between efficiency-driven prototyping (“intending the right design”) and careful reflection (“designing the right intention”), which introduces new asymmetries in trust, responsibility, and potentially negative social perceptions within product teams. Through the lens of responsible human-human and human-AI collaboration for AI-Assisted product design and development, we reflect on and contribute to a deeper understanding of worker deskilling, ownership and AI disclosure, and the implications toward creativity safeguarding in this age of vibe coding.

Positionality and AI Usage Disclosure Statement

We recognize author positionality shaped perspectives in this paper [33, 66]. Our team comprises six researchers located in the USA and the Netherlands, with backgrounds in cognitive science, HCI, UX, machine learning, and data science. These diverse perspectives, spanning academic and industry research and design, informed our research framing and analysis. Data collection and analysis were conducted manually. Microsoft Teams was used to record and transcribe interviews and Google NotebookLM to organize the coding manual. Thematic analysis and affinity diagramming were carried out collaboratively by five researchers, ensuring reflexive, iterative engagement [7, 36]. We used LLMs (GPT-5.2 and Claude Sonnet 4) to suggest writing structure and readability enhancements. All analytical decisions, codes, and interpretive claims including our discussion remain human-only. We disclose this AI usage to ensure transparency in AI-mediated scholarship in HCI [25].

References

- [1] Mehmet Akhoro and Caglar Yildirim. 2025. Conversational AI as a Coding Assistant: Understanding Programmers' Interactions with and Expectations from Large Language Models for Coding. arXiv:2503.16508 [cs.HC] <https://arxiv.org/abs/2503.16508>
- [2] Salema Amershi, Dan Weld, Mihaela Vorvoreanu, Adam Fourney, Besmira Nushi, Penny Collisson, Jina Suh, Shamsi Iqbal, Paul N. Bennett, Kori Inkpen, Jaime Teevan, Ruth Kikin-Gil, and Eric Horvitz. 2019. Guidelines for Human-AI Interaction. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland UK) (CHI '19). Association for Computing Machinery, New York, NY, USA, 1–13. <https://doi.org/10.1145/3290605.3300233>
- [3] Shradha Barke, Michael B. James, and Nadia Polikarpova. 2023. Grounded Copilot: How Programmers Interact with Code-Generating Models. *Proc. ACM Program. Lang.* 7, OOPSLA1, Article 78 (April 2023), 27 pages. <https://doi.org/10.1145/3586030>
- [4] Joel Becker, Nate Rush, Elizabeth Barnes, and David Rein. 2025. Measuring the Impact of Early-2025 AI on Experienced Open-Source Developer Productivity.
- [5] Antoine Bellemare-Pepin, François Lespinasse, Philipp Thölke, Yann Harel, Kory Mathewson, Jay A. Olson, Yoshua Bengio, and Karim Jerbi. 2025. Divergent Creativity in Humans and Large Language Models. arXiv:2405.13012 [cs.CL] <https://arxiv.org/abs/2405.13012>
- [6] Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, et al. 2021. On the Opportunities and Risks of Foundation Models. <https://crfm.stanford.edu/assets/report.pdf>
- [7] Virginia Braun and Victoria Clarke. 2006. Using thematic analysis in psychology. *Qualitative Research in Psychology* 3, 2 (2006), 77–101. <https://doi.org/10.1191/1478088706qp0630a>
- [8] Anders Bruun, Niels van Berkel, Dimitrios Raptis, and Effie L-C Law. 2025. Coordination Mechanisms in AI Development: Practitioner Experiences on Integrating UX Activities. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems* (CHI '25). Association for Computing Machinery, New York, NY, USA, Article 796, 14 pages. <https://doi.org/10.1145/3706598.3713200>
- [9] Erik Brynjolfsson, Danielle Li, and Lindsey Raymond. 2025. Generative AI at work. *The Quarterly Journal of Economics* 140, 2 (2025), 889–942.
- [10] Sara Bunian, Kai Li, Chaima Jemmali, Casper Harteveld, Yun Fu, and Magy Seif Seif El-Nasr. 2021. VINS: Visual Search for Mobile User Interface Design. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) (CHI '21). Association for Computing Machinery, New York, NY, USA, Article 423, 14 pages. <https://doi.org/10.1145/3411764.3445762>
- [11] Bill Buxton. 2010. *Sketching user experiences: getting the design right and the right design*. Morgan kaufmann, San FranciscoCAUnited States.
- [12] Kelly Caine. 2016. Local Standards for Sample Size at CHI. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (San Jose, California, USA) (CHI '16). Association for Computing Machinery, New York, NY, USA, 981–992. <https://doi.org/10.1145/2858036.2858498>
- [13] Inha Cha and Richmond Y. Wong. 2025. Understanding Socio-technical Factors Configuring AI Non-Use in UX Work Practices. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems* (CHI '25). Association for Computing Machinery, New York, NY, USA, Article 1110, 17 pages. <https://doi.org/10.1145/3706598.3713140>
- [14] Minsuk Chang, Stefania Druga, Alexander J. Fiannaca, Pedro Vergani, Chinmay Kulkarni, Carrie J Cai, and Michael Terry. 2023. The Prompt Artists. In *Proceedings of the 15th Conference on Creativity and Cognition* (Virtual Event, USA) (C&C '23). Association for Computing Machinery, New York, NY, USA, 75–87. <https://doi.org/10.1145/3591196.3593515>
- [15] Li-Yuan Chiou, Peng-Kai Hung, Rung-Huei Liang, and Chun-Teng Wang. 2023. Designing with AI: An Exploration of Co-Ideation with Image Generators. In *Proceedings of the 2023 ACM Designing Interactive Systems Conference* (Pittsburgh, PA, USA) (DIS '23). Association for Computing Machinery, New York, NY, USA, 1941–1954. <https://doi.org/10.1145/3563657.3596001>
- [16] Marios Constantinides, Himanshu Verma, Shadan Sadeghian, and Abdallah El Ali. 2025. The Future of Work is Blended, Not Hybrid. In *Proceedings of the 4th Annual Symposium on Human-Computer Interaction for Work* (CHIWORK '25). Association for Computing Machinery, New York, NY, USA, Article 28, 13 pages. <https://doi.org/10.1145/3729176.3729202>
- [17] Tiago Silva Da Silva, Milene Selbach Silveira, Frank Maurer, and Theodore Hellmann. 2012. User experience design and agile development: From theory to practice. *Journal of Software Engineering and Applications* 5, 10 (2012), 743–751.
- [18] Simone Daniotti, Johannes Wachs, Xiangnan Feng, and Frank Neffke. 2026. Who is using AI to code? Global diffusion and impact of generative AI. *Science* 391 (2026), 831–835. Issue 6787. <https://doi.org/10.1126/science.adz9311> arXiv:<https://www.science.org/doi/pdf/10.1126/science.adz9311>
- [19] Valdemar Danry, Pat Pataranutaporn, Yaoli Mao, and Pattie Maes. 2023. Don't Just Tell Me, Ask Me: AI Systems that Intelligently Frame Explanations as Questions Improve Human Logical Discernment Accuracy over Causal AI Explanations. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems* (CHI '23). Association for Computing Machinery, New York, NY, USA, Article 352, 13 pages. <https://doi.org/10.1145/3544548.3580672>
- [20] Fabrizio Dell'Acqua, Charles Ayoubi, Hila Lifshitz, Raffaella Sadun, Ethan Mollick, Lilach Mollick, Yi Han, Jeff Goldman, Hari Nair, Stewart Taub, et al. 2025. *The cybernetic teammate: A field experiment on generative AI reshaping teamwork and expertise*. Technical Report. National Bureau of Economic Research.
- [21] Fabrizio Dell'Acqua, Edward McFowland III, Ethan R Mollick, Hila Lifshitz-Assaf, Katherine Kellogg, Saran Rajendran, Lisa Kray, François Candelon, and Karim R Lakhani. 2023. Navigating the jagged technological frontier: Field experimental evidence of the effects of AI on knowledge worker productivity and quality. *Harvard Business School Technology & Operations Mgt. Unit Working Paper* 37, 2 (2023), 403–423.
- [22] Fiona Draxler, Anna Werner, Florian Lehmann, Matthias Hoppe, Albrecht Schmidt, Daniel Buschek, and Robin Welsch. 2024. The AI Ghostwriter Effect: When Users Do Not Perceive Ownership of AI-Generated Text but Self-Declare as Authors. *ACM Transactions on Computer-Human Interaction* 31, 1 (2024), 1–35. <https://doi.org/10.1145/3623150>
- [23] Peitong Duan, Jeremy Warner, Yang Li, and Bjoern Hartmann. 2024. Generating Automatic Feedback on UI Mockups with Large Language Models. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) (CHI '24). Association for Computing Machinery, New York, NY, USA, Article 6, 20 pages. <https://doi.org/10.1145/3613904.3642782>
- [24] Abdallah El Ali, Karthikeya Puttur Venkatraj, Sophie Morosoli, Laurens Naudts, Natali Helberger, and Pablo Cesar. 2024. Transparent AI Disclosure Obligations: Who, What, When, Where, Why, How. In *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) (CHI EA '24). Association for Computing Machinery, New York, NY, USA, Article 342, 11 pages. <https://doi.org/10.1145/3613905.3650750>
- [25] Passant Elagroudy, Jie Li, Kaisa Väänänen, Paul Lukowicz, Hiroshi Ishii, Wendy E. Mackay, Elizabeth F Churchill, Anicia Peters, Antti Oulasvirta, Rui Prada, Alexandra Diening, Giulia Barbareschi, Agnes Gruenerbl, Midori Kawaguchi, Abdallah El Ali, Fiona Draxler, Robin Welsch, and Albrecht Schmidt. 2024. Transforming HCI Research Cycles using Generative AI and “Large Whatever Models” (LWMs). In *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) (CHI EA '24). Association for Computing Machinery, New York, NY, USA, Article 584, 5 pages. <https://doi.org/10.1145/3613905.3643977>
- [26] Tyna Eloundou, Sam Manning, Pamela Mishkin, and Daniel Rock. 2024. GPTs are GPTs: Labor market impact potential of LLMs. *Science* 384, 6702 (2024), 1306–1308.
- [27] Ahmed Fawzy, Amjed Tahir, and Kelly Blincoe. 2025. Vibe Coding in Practice: Motivations, Challenges, and a Future Outlook – a Grey Literature Review. arXiv:2510.00328 [cs.SE] <https://arxiv.org/abs/2510.00328>
- [28] Sidong Feng, Mingyue Yuan, Jieshan Chen, Zhenchang Xing, and Chunyang Chen. 2023. Designing with Language: Wireframing UI Design Intent with Generative Large Language Models. arXiv:2312.07755.
- [29] Jennifer Fereday and Eimear Muir-Cochrane. 2006. Demonstrating Rigor Using Thematic Analysis: A Hybrid Approach of Inductive and Deductive Coding and Theme Development. *International Journal of Qualitative Methods* 5, 1 (2006), 80–92. <https://doi.org/10.1177/160940690600500107> arXiv:<https://doi.org/10.1177/160940690600500107>
- [30] Claudia Flores-Saviaga, Benjamin V. Hanrahan, Kashif Imteyaz, Steven Clarke, and Saiph Savage*. 2025. The Impact of Generative AI Coding Assistants on Developers Who Are Visually Impaired. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems* (CHI '25). Association for Computing Machinery, New York, NY, USA, Article 1164, 17 pages. <https://doi.org/10.1145/3706598.3713140>

- 3706598.3714008
- [31] Fortune Staff. 2025. AI Coding Tool Replit Wiped Database, Called It a “Catastrophic Failure”. <https://fortune.com/2025/07/23/ai-coding-tool-replit-wiped-database-called-it-a-catastrophic-failure/>. Accessed: 2025-09-09.
 - [32] Jonas Frich, Lindsay MacDonald Vermeulen, Christian Remy, Michael Mose Biskjaer, and Peter Dalsgaard. 2019. Mapping the Landscape of Creativity Support Tools in HCI. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland UK) (CHI '19). Association for Computing Machinery, New York, NY, USA, 1–18. <https://doi.org/10.1145/3290605.3300619>
 - [33] Hana Fruckaj, Laura Dabbish, David Gray Widder, Huilian Sophie Qiu, and James D. Herbsleb. 2022. Gender and Participation in Open Source Software Development. *Proc. ACM Hum.-Comput. Interact.* 6, CSCW2, Article 299 (Nov. 2022), 31 pages. <https://doi.org/10.1145/3555190>
 - [34] Ariel Han and Zhenyao Cai. 2023. Design Implications of Generative AI Systems for Visual Storytelling for Young Learners. In *Proceedings of the 22nd Annual ACM Interaction Design and Children Conference* (Chicago, IL, USA) (IDC '23). Association for Computing Machinery, New York, NY, USA, 470–474. <https://doi.org/10.1145/3585088.3593867>
 - [35] Gunnar Harboe and Elaine M. Huang. 2015. Real-World Affinity Diagramming Practices: Bridging the Paper-Digital Gap. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (Seoul, Republic of Korea) (CHI '15). Association for Computing Machinery, New York, NY, USA, 95–104. <https://doi.org/10.1145/2702123.2702561>
 - [36] Gunnar Harboe and Elaine M. Huang. 2015. Real-World Affinity Diagramming Practices: Bridging the Paper-Digital Gap. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (Seoul, Republic of Korea) (CHI '15). Association for Computing Machinery, New York, NY, USA, 95–104. <https://doi.org/10.1145/2702123.2702561>
 - [37] Jessica He, Stephanie Houde, Gabriel E. Gonzalez, Darío Andrés Silva Moran, Steven I. Ross, Michael Muller, and Justin D. Weisz. 2024. AI and the Future of Collaborative Work: Group Ideation with an LLM in a Virtual Canvas. In *Proceedings of the 3rd Annual Meeting of the Symposium on Human-Computer Interaction for Work* (Newcastle upon Tyne, United Kingdom) (CHIWORK '24). Association for Computing Machinery, New York, NY, USA, Article 9, 14 pages. <https://doi.org/10.1145/3663384.3663398>
 - [38] Jessica He, Stephanie Houde, and Justin D. Weisz. 2025. Which Contributions Deserve Credit? Perceptions of Attribution in Human-AI Co-Creation. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems* (CHI '25). Association for Computing Machinery, New York, NY, USA, Article 540, 18 pages. <https://doi.org/10.1145/3706598.3713522>
 - [39] Monique Hennink and Bonnie N. Kaiser. 2022. Sample sizes for saturation in qualitative research: A systematic review of empirical tests. *Social Science & Medicine* 292 (2022), 114523. <https://doi.org/10.1016/j.socscimed.2021.114523>
 - [40] Marko Horvat. 2025. What is Vibe coding and when should you use it (or not)?
 - [41] Nanna Inie, Jeanette Falk, and Steve Tanimoto. 2023. Designing Participatory AI: Creative Professionals’ Worries and Expectations about Generative AI. In *Extended Abstracts of the 2023 CHI Conference on Human Factors in Computing Systems* (Hamburg, Germany) (CHI EA '23). Association for Computing Machinery, New York, NY, USA, Article 82, 8 pages. <https://doi.org/10.1145/3544549.3585657>
 - [42] Harry H. Jiang, Lauren Brown, Jessica Cheng, Mehtab Khan, Abhishek Gupta, Deja Workman, Alex Hanna, Johnathan Flowers, and Timnit Gebru. 2023. AI Art and Its Impact on Artists. In *Proceedings of the 2023 AAAI/ACM Conference on AI, Ethics, and Society* (Montréal, QC, Canada) (AI/ES '23). Association for Computing Machinery, New York, NY, USA, 363–374. <https://doi.org/10.1145/3600211.3604681>
 - [43] Zachary Kaiser. 2019. Creativity as computation: Teaching design in the age of automation. *Design and Culture* 11, 2 (2019), 173–192.
 - [44] Andrej Karpathy. 2025. There’s a new kind of coding I call “vibe coding”, where you fully give in to the vibes, embrace exponentials, and forget that the code even exists. <https://x.com/karpathy/status/1886192184808149383?lang=en>. Accessed: 2025-09-12.
 - [45] Majeed Kazemitabaar, Justin Chow, Carl Ka To Ma, Barbara J. Ericson, David Weintrop, and Tovi Grossman. 2023. Studying the effect of AI Code Generators on Supporting Novice Learners in Introductory Programming. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems* (Hamburg, Germany) (CHI '23). Association for Computing Machinery, New York, NY, USA, Article 455, 23 pages. <https://doi.org/10.1145/3544548.3580919>
 - [46] Abidullah Khan, Atefeh Shokrizadeh, and Jinghui Cheng. 2025. Beyond Automation: How Designers Perceive AI as a Creative Partner in the Divergent Thinking Stages of UI/UX Design. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems* (CHI '25). Association for Computing Machinery, New York, NY, USA, Article 1105, 12 pages. <https://doi.org/10.1145/3706598.3713500>
 - [47] Joke Kort, APOS Vermeeren, and Jenneke E Fokker. 2007. Conceptualizing and measuring user experience. , 57 pages.
 - [48] Chinmay Kulkarni, Stefania Druga, Minsuk Chang, Alex Fiannaca, Carrie Cai, and Michael Terry. 2023. A Word is Worth a Thousand Pictures: Prompts as AI Design Material. arXiv:2303.12647 [cs.HC]
 - [49] Harsh Kumar, Jonathan Vincentius, Ewan Jordan, and Ashton Anderson. 2025. Human Creativity in the Age of LLMs: Randomized Experiments on Divergent and Convergent Thinking. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems* (CHI '25). Association for Computing Machinery, New York, NY, USA, Article 23, 18 pages. <https://doi.org/10.1145/3706598.3714198>
 - [50] Amber Kusters, Pooja Prajod, Pablo Cesar, and Abdallah El Ali. 2026. More Human or More AI? Visualizing Human-AI Collaboration Disclosures in Journalistic News Production. arXiv:2601.11072 [cs.HC] <https://arxiv.org/abs/2601.11072>
 - [51] Tomas Lawton, Kazjon Grace, and Francisco J Ibarrola. 2023. When is a Tool a Tool? User Perceptions of System Agency in Human-AI Co-Creative Drawing. In *Proceedings of the 2023 ACM Designing Interactive Systems Conference* (Pittsburgh, PA, USA) (DIS '23). Association for Computing Machinery, New York, NY, USA, 1978–1996. <https://doi.org/10.1145/3563657.3595977>
 - [52] Jie Li, Hancheng Cao, Laura Lin, Youyang Hou, Ruihao Zhu, and Abdallah El Ali. 2024. User Experience Design Professionals’ Perceptions of Generative Artificial Intelligence. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) (CHI '24). Association for Computing Machinery, New York, NY, USA, Article 381, 18 pages. <https://doi.org/10.1145/3613904.3642114>
 - [53] Jenny T. Liang, Chenyang Yang, and Brad A. Myers. 2024. A Large-Scale Survey on the Usability of AI Programming Assistants: Successes and Challenges. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering* (Lisbon, Portugal) (ICSE '24). Association for Computing Machinery, New York, NY, USA, Article 52, 13 pages. <https://doi.org/10.1145/3597503.3608128>
 - [54] Weixin Liang, Yaohui Zhang, Mihai Codreanu, Jiayu Wang, Hancheng Cao, and James Zou. 2025. The widespread adoption of large language model-assisted writing across society.
 - [55] Yuyu Lin, Jiahao Guo, Yang Chen, Cheng Yao, and Fangtian Ying. 2020. It Is Your Turn: Collaborative Ideation With a Co-Creative Robot through Sketch. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) (CHI '20). Association for Computing Machinery, New York, NY, USA, 1–14. <https://doi.org/10.1145/3313831.3376258>
 - [56] Zhiyu Lin, Upol Ehsan, Rohan Agarwal, Samihan Dani, Vidushi Vashishth, and Mark Riedl. 2023. Beyond Prompts: Exploring the Design Space of Mixed-Initiative Co-Creativity Systems.
 - [57] Michael Xieyang Liu, Advait Sarkar, Carina Negreanu, Benjamin Zorn, Jack Williams, Neil Toronto, and Andrew D. Gordon. 2023. “What It Wants Me To Say”: Bridging the Abstraction Gap Between End-User Programmers and Code-Generating Large Language Models. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems* (Hamburg, Germany) (CHI '23). Association for Computing Machinery, New York, NY, USA, Article 598, 31 pages. <https://doi.org/10.1145/3544548.3580817>
 - [58] Robyn Longhurst. 2003. Semi-structured interviews and focus groups. *Key methods in geography* 3, 2 (2003), 143–156.
 - [59] Andreas Matthias. 2004. The responsibility gap: Ascribing responsibility for the actions of learning automata. *Ethics and Inf. Technol.* 6, 3 (Sept. 2004), 175–183. <https://doi.org/10.1007/s10676-004-3422-1>
 - [60] Nora McDonald, Sarita Schoenebeck, and Andrea Forte. 2019. Reliability and Inter-rater Reliability in Qualitative Research: Norms and Guidelines for CSCW and HCI Practice. *Proceedings of the ACM on Human-Computer Interaction* 3 (11 2019), 1–23. <https://doi.org/10.1145/3359174>
 - [61] Christian Meske, Tobias Hermanns, Esther von der Weiden, Kai-Uwe Loser, and Thorsten Berger. 2025. Vibe Coding as a Reconfiguration of Intent Mediation in Software Development: Definition, Implications, and Research Agenda. arXiv:2507.21928 [cs.SE] <https://arxiv.org/abs/2507.21928>
 - [62] Bin Ning, Fang Liu, and Zhixiong Liu. 2023. Creativity support in AI co-creative tools: Current research, challenges and opportunities. , 5–10 pages.
 - [63] Lorelli S. Nowell, Jill M. Norris, Deborah E. White, and Nancy J. Moules. 2017. Thematic Analysis: Striving to Meet the Trustworthiness Criteria. *International Journal of Qualitative Methods* 16, 1 (2017), 1609406917733847. <https://doi.org/10.1177/1609406917733847> arXiv:https://doi.org/10.1177/1609406917733847
 - [64] Shakked Noy and Whitney Zhang. 2023. Experimental Evidence on the Productivity Effects of Generative Artificial Intelligence. *Science* 381, 6654 (2023), 187–192. <https://doi.org/10.1126/science.adh2586>
 - [65] Jay A. Olson, Johnny Nahas, Denis Chmoulevitch, Simon J. Cropper, and Margaret E. Webb. 2021. Naming unrelated words predicts creativity. *Proceedings of the National Academy of Sciences of the United States of America* 118, 25 (2021), e2022340118. <https://doi.org/10.1073/pnas.2022340118>
 - [66] Alexandra Olteanu, Michael Ekstrand, Carlos Castillo, and Jina Suh. 2023. Responsible AI Research Needs Impact Statements Too. arXiv:2311.11776 [cs.AI] <https://arxiv.org/abs/2311.11776>
 - [67] Sida Peng, Eirini Kalliamvakou, Peter Cihon, and Mert Demirel. 2023. The impact of ai on developer productivity: Evidence from github copilot.
 - [68] Bogdana Rakova, Jingying Yang, Henriette Cramer, and Rumman Chowdhury. 2021. Where Responsible AI Meets Reality: Practitioner Perspectives on Enablers for Shifting Organizational Practices. *Proc. ACM Hum.-Comput. Interact.* 5, CSCW1, Article 7 (apr 2021), 23 pages. <https://doi.org/10.1145/3449081>

- [69] Pasi Ritala, Miika Ruokonen, and Lakshmy Ramaul. 2023. Transforming Boundaries: How Does ChatGPT Change Knowledge Work? *Journal of Business Strategy* 45, 5 (2023), 214–220. <https://doi.org/10.1108/JBS-05-2023-0094>
- [70] Armin Ronacher. 2026. Agent Psychosis: Are We Going Insane? Online blog post. <https://lucumr.pocoo.org/2026/1/18/agent-psychosis/> Accessed: 2026-02-02.
- [71] Steven I. Ross, Fernando Martinez, Stephanie Houde, Michael Muller, and Justin D. Weisz. 2023. The Programmer's Assistant: Conversational Interaction with a Large Language Model for Software Development. In *Proceedings of the 28th International Conference on Intelligent User Interfaces* (Sydney, NSW, Australia) (IUI '23). Association for Computing Machinery, New York, NY, USA, 491–514. <https://doi.org/10.1145/3581641.3584037>
- [72] Ranjan Sapkota, Konstantinos I. Roumeliotis, and Manoj Karkee. 2025. Vibe Coding vs. Agentic Coding: Fundamentals and Practical Implications of Agentic AI. arXiv:2505.19443 [cs.SE] <https://arxiv.org/abs/2505.19443>
- [73] Advait Sarkar and Ian Drosos. 2025. Vibe coding: programming through conversation with artificial intelligence. arXiv:2506.23253 [cs.HC] <https://arxiv.org/abs/2506.23253>
- [74] Agnia Sergeyuk, Yaroslav Golubev, Timofey Bryksin, and Iftekhar Ahmed. 2025. Using AI-based coding assistants in practice: State of affairs, perceptions, and ways forward. *Information and Software Technology* 178 (2025), 107610. <https://doi.org/10.1016/j.infsof.2024.107610>
- [75] Judy Hanwen Shen and Alex Tamkin. 2026. How AI Impacts Skill Formation. arXiv:2601.20245 [cs.CY] <https://arxiv.org/abs/2601.20245>
- [76] Yang Shi, Tian Gao, Xiaohan Jiao, and Nan Cao. 2023. Understanding Design Collaboration Between Designers and Artificial Intelligence: A Systematic Literature Review. *Proceedings of the ACM on Human-Computer Interaction* 7, CSCW2 (2023), 1–35. <https://doi.org/10.1145/3610217>
- [77] Joongi Shin, Michael A. Hedderich, Bartłomiej Jakub Rey, Andrés Lucero, and Antti Oulasvirta. 2024. Understanding Human-AI Workflows for Generating Personas. In *Proceedings of the 2024 ACM Designing Interactive Systems Conference* (Copenhagen, Denmark) (DIS '24). Association for Computing Machinery, New York, NY, USA, 757–781. <https://doi.org/10.1145/3643834.3660729>
- [78] Joongi Shin, Anna Polyanskaya, Andrés Lucero, and Antti Oulasvirta. 2025. No Evidence for LLMs Being Useful in Problem Reframing. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems* (CHI '25). Association for Computing Machinery, New York, NY, USA, Article 243, 25 pages. <https://doi.org/10.1145/3706598.3713273>
- [79] Atefeh Shokrizadeh, Bonifata Bahati Tadjuidje, Shivam Kumar, Sohan Kamble, and Jinghui Cheng. 2025. Dancing With Chains: Ideating Under Constraints With UIDEC in UI/UX Design. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems* (CHI '25). Association for Computing Machinery, New York, NY, USA, Article 1106, 23 pages. <https://doi.org/10.1145/3706598.3713785>
- [80] Irene Solaiman, Zeerak Talat, William Agnew, Lama Ahmad, Dylan Baker, Su Lin Blodgett, Hal Daumé III, Jesse Dodge, Ellie Evans, Sara Hooker, et al. 2023. Evaluating the Social Impact of Generative AI Systems in Systems and Society.
- [81] Stuart Spence. 2025. Why I'm Declining Your AI-Generated MR. <https://blog.stuartspence.ca/2025-08-declining-ai-slop-mr.html> Blog post.
- [82] Åsne Stige, Efpraxia D Zamani, Patrick Mikalef, and Yuzhen Zhu. 2023. Artificial intelligence (AI) for user experience (UX) design: a systematic literature review and future research agenda. *Information Technology & People* 37 (2023), 2324–2352. Issue 6.
- [83] Hari Subramonyam, Divy Thakkar, Andrew Ku, Juergen Dieber, and Anoop K. Sinha. 2025. Prototyping with Prompts: Emerging Approaches and Challenges in Generative AI Design for Collaborative Software Teams. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems* (CHI '25). Association for Computing Machinery, New York, NY, USA, Article 882, 22 pages. <https://doi.org/10.1145/3706598.3713166>
- [84] Vivek Sunkara. 2025. The Vibe Coding Imperative for Product Managers. Blog@CACM, Communications of the ACM. <https://cacm.acm.org/blogcacm/the-vibe-coding-imperative-for-product-managers/> Accessed 2026-02-01.
- [85] Alexey Svyatkovskiy, Shao Kun Deng, Shengyu Fu, and Neel Sundaresan. 2020. IntelliCode compose: code generation using transformer. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (Virtual Event, USA) (ESEC/FSE 2020). Association for Computing Machinery, New York, NY, USA, 1433–1443. <https://doi.org/10.1145/3368089.3417058>
- [86] Macy Takaffoli, Sijia Li, and Ville Mäkelä. 2024. Generative AI in User Experience Design and Research: How Do UX Practitioners, Teams, and Companies Use GenAI in Industry?. In *Proceedings of the 2024 ACM Designing Interactive Systems Conference* (Copenhagen, Denmark) (DIS '24). Association for Computing Machinery, New York, NY, USA, 1579–1593. <https://doi.org/10.1145/3643834.3660720>
- [87] Sirui Tao, Ivan Liang, Cindy Peng, Zhiqing Wang, Srishti Palani, and Steven P. Dow. 2025. DesignWeaver: Dimensional Scaffolding for Text-to-Image Product Design. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems* (CHI '25). Association for Computing Machinery, New York, NY, USA, Article 425, 26 pages. <https://doi.org/10.1145/3706598.3714211>
- [88] Jakob Tholander and Martin Jonsson. 2023. Design Ideation with AI - Sketching, Thinking and Talking with Generative Machine Learning Models. In *Proceedings of the 2023 ACM Designing Interactive Systems Conference* (Pittsburgh, PA, USA) (DIS '23). Association for Computing Machinery, New York, NY, USA, 1930–1940. <https://doi.org/10.1145/3563657.3596014>
- [89] Maryam Tohidi, William Buxton, Ronald Baecker, and Abigail Sellen. 2006. Getting the right design and the design right. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Montréal, Québec, Canada) (CHI '06). Association for Computing Machinery, New York, NY, USA, 1243–1252. <https://doi.org/10.1145/1124772.1124960>
- [90] Priyan Vaithilingam, Tianyi Zhang, and Elena L. Glassman. 2022. Expectation vs. Experience: Evaluating the Usability of Code Generation Tools Powered by Large Language Models. In *Extended Abstracts of the 2022 CHI Conference on Human Factors in Computing Systems* (New Orleans, LA, USA) (CHI EA '22). Association for Computing Machinery, New York, NY, USA, Article 332, 7 pages. <https://doi.org/10.1145/3491101.3519665>
- [91] Qian Wan and Zhicong Lu. 2023. GANCollage: A GAN-Driven Digital Mood Board to Facilitate Ideation in Creativity Support. In *Proceedings of the 2023 ACM Designing Interactive Systems Conference* (Pittsburgh, PA, USA) (DIS '23). Association for Computing Machinery, New York, NY, USA, 136–146. <https://doi.org/10.1145/3563657.3596072>
- [92] Weiguang Wang, Guodong (Gordon) Gao, and Ritu Agarwal. 2024. Friend or Foe? Teaming Between Artificial Intelligence and Workers with Variation in Experience. *Manage. Sci.* 70, 9 (Sept. 2024), 5753–5775. <https://doi.org/10.1287/mnsc.2021.00588>
- [93] Wen-Fan Wang, Chien-Ting Lu, Nil Ponsa i Campanyà, Bing-Yu Chen, and Mike Y. Chen. 2025. Aldeation: Designing a Human-AI Collaborative Ideation System for Concept Designers. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems* (CHI '25). Association for Computing Machinery, New York, NY, USA, Article 21, 28 pages. <https://doi.org/10.1145/3706598.3714148>
- [94] Wen Xu and Katina Zammit. 2020. Applying Thematic Analysis to Education: A Hybrid Approach to Interpreting Data in Practitioner Research. *International Journal of Qualitative Methods* 19 (2020), 1609406920918810. <https://doi.org/10.1177/1609406920918810> arXiv:https://doi.org/10.1177/1609406920918810
- [95] Xiaotong (Tone) Xu, Arina Konnova, Bianca Gao, Cindy Peng, Dave Vo, and Steven P. Dow. 2025. Productive vs. Reflective: How Different Ways of Integrating AI into Design Workflows Affect Cognition and Motivation. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems* (CHI '25). Association for Computing Machinery, New York, NY, USA, Article 24, 15 pages. <https://doi.org/10.1145/3706598.3713649>
- [96] Chuan Yan, John Joon Young Chung, Yoon Kiheon, Yotam Gingold, Eytan Adar, and Sungsoo Ray Hong. 2022. FlatMagic: Improving Flat Colorization through AI-driven Design for Digital Comic Professionals. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems* (New Orleans, LA, USA) (CHI '22). Association for Computing Machinery, New York, NY, USA, Article 380, 17 pages. <https://doi.org/10.1145/3491102.3502075>
- [97] Nur Yildirim, Alex Kass, Teresa Tung, Connor Upton, Donnacha Costello, Robert Giusti, Sinem Lacin, Sara Lovic, James M O'Neill, Rudi O'Reilly Meehan, Eoin Ó Loideáin, Azzurra Pini, Medb Corcoran, Jeremiah Hayes, Diarmuid J Cahalane, Gaurav Shivhare, Luigi Castoro, Giovanni Caruso, Changhoon Oh, James McCann, Jodi Forlizzi, and John Zimmerman. 2022. How Experienced Designers of Enterprise Applications Engage AI as a Design Material. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems* (New Orleans, LA, USA) (CHI '22). Association for Computing Machinery, New York, NY, USA, Article 483, 13 pages. <https://doi.org/10.1145/3491102.3517491>
- [98] J.D. Zamfirescu-Pereira, Eunice Jun, Michael Terry, Qian Yang, and Bjoern Hartmann. 2025. Beyond Code Generation: LLM-supported Exploration of the Program Design Space. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems* (CHI '25). Association for Computing Machinery, New York, NY, USA, Article 153, 17 pages. <https://doi.org/10.1145/3706598.3714154>
- [99] J.D. Zamfirescu-Pereira, Richmond Y. Wong, Bjoern Hartmann, and Qian Yang. 2023. Why Johnny Can't Prompt: How Non-AI Experts Try (and Fail) to Design LLM Prompts. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems* (Hamburg, Germany) (CHI '23). Association for Computing Machinery, New York, NY, USA, Article 437, 21 pages. <https://doi.org/10.1145/3544548.3581388>