Vibe Design Assistant: Towards Design Intent Representation for Collaborative Vibe Coding

AASHISH DHAKAL, University at Buffalo, USA
QINGXIAO ZHENG, University at Buffalo, USA
JINJUN XIONG, University at Buffalo, USA

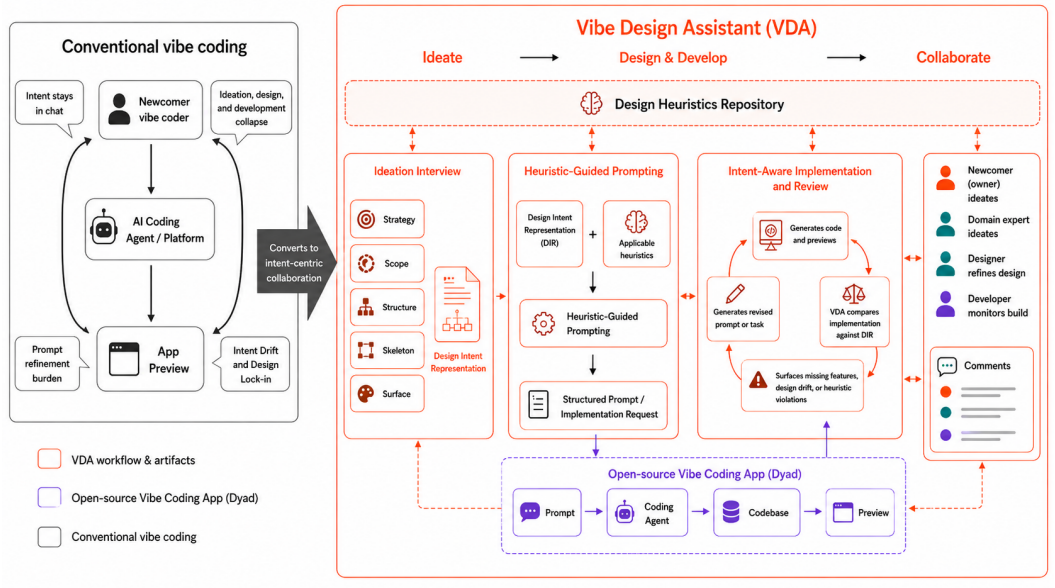

Fig. 1. **Demo workflow of Vibe Design Assistant (VDA).** The left panel shows a conventional vibe coding workflow, where newcomers rely on a prompt–preview loop and project intent remains fragmented across chat, code, and preview. The right panel shows VDA's workflow: the Ideation Interview helps users articulate their design intent into an editable Design Intent Representation (DIR) which preserves this intent as a shared artifact; Heuristic-Guided Prompting uses the DIR to generate structured prompts; Intent-Aware Implementation and Review checks generated code and interface changes against the DIR; and Collaborator Review lets all collaborators comment on the same representation.

Existing vibe coding workflows treat prompts as the primary interaction object. However, prompts are ephemeral, implementation-oriented, and poorly suited for supporting shared understanding across collaborators. As a result, many collaboration challenges exist in current vibe coding workflows: intent is buried inside chat history, AI-generated changes erase rationale continuity, collaboration becomes output-centric rather than intent-centric, and shared understanding collapses because prompts are not stable boundary objects. To address these collaboration challenges, we propose to transform vibe coding from prompt-centric interaction into an intent-centric collaboration through Vibe Design Assistant (VDA), a framework that turns "design intent" into a shared, persistent, and negotiable coordination artifact across heterogeneous collaborators with different expertise levels.

CCS Concepts: • **Human-centered computing** → **Collaborative interaction**; **Interactive systems and tools**; • **Software and its engineering** → *Integrated and visual development environments*; • **Computing methodologies** → *Artificial intelligence.*

Authors' Contact Information: Aashish Dhakal, University at Buffalo, Buffalo, USA, adhakal2@buffalo.edu; Qingxiao Zheng, University at Buffalo, Buffalo, USA; Jinjun Xiong, University at Buffalo, Buffalo, NY, USA.

Additional Key Words and Phrases: Vibe coding, design intent, human-AI collaboration, end-user development, AI-assisted programming, collaborative software development

1 Introduction

Vibe coding platforms such as Lovable[1] and Replit[2] suggest a new mode of software creation in which users describe what they want in natural language and AI generates or modifies the resulting software artifact. Although this workflow promises to democratize software creation, successful use remains difficult for novice vibe coders: people who can use vibe coding systems to generate software, but lack the design, technical, or domain expertise needed to reliably evaluate and direct the resulting artifact [5]. This is particularly visible when vibe coders build user-facing web applications. Effective design frameworks, such as the Double Diamond design process and Garrett's five planes of design [2], separate ideation, design, and development into distinct phases; vibe coding, by contrast, collapses these activities into a single prompting layer. As a result, design decisions that would normally be articulated through discussions and negotiations remain implicit in chat history or are rarely articulated at all. In this way, vibe coding forces the collaborative web design and development process into an individualistic endeavor, introducing several challenges especially for novice vibe coders who would especially benefit from collaborating with experts to extract and express their design intent.

Moreover, users may struggle to express intent in a form that an AI coding system can reliably execute, because effective prompting requires both prompt engineering competence [3, 4, 9] and technical knowledge [4, 5, 10]. Additionally, generated applications may drift from the user's intent over time, producing misalignment between user intent and model output [9], unintended features [4], repeated prompt repair [3, 4], and shallow vibe coding sessions [3, 6, 9] which eventually leads to project abandonment [9]. Beyond challenges in prompting quality and implementation drift, vibe coding also introduces new collaboration breakdowns. In current workflows, project intent is often fragmented across prompts, generated outputs, chat histories, and partially implemented code, making it difficult for collaborators to establish shared understanding around why particular design and implementation decisions were made. This becomes especially problematic when newcomers collaborate with domain experts, designers, and developers who operate at different levels of abstraction and may not directly participate in prompting interactions. As prompts increasingly become the primary interaction mechanism for AI-assisted software creation, they simultaneously function as unstable and ephemeral coordination artifacts that are poorly suited for supporting long-term collaborative reasoning, negotiation, and accountability.

Therefore, we argue that collaborative vibe coding requires shifting from prompt-centric interaction toward persistent representations of design intent that can support coordination between humans and AI throughout ideation, design, development, and review. To this end, we present **Vibe Design Assistant (VDA)**, a framework that turns "design intent" into a shared, persistent, negotiable coordination artifact across heterogeneous collaborators with different expertise levels. Through VDA, we propose to transform vibe coding from a prompt-centric interaction into an intent-centric collaboration. Our contributions are as follows:

(1) A conceptual reframing of vibe coding from prompt-centric generation to intent-centric collaborative coordination.
(2) Design intent representation (DIR), a persistent editable coordination artifact supporting cross-role collaboration between newcomers, designers, developers, and domain experts.

[1] https://lovable.dev
[2] https://replit.com/ai

(3) Workflow architecture that integrates ideation scaffolding, heuristic-guided prompting, and intent-aware implementation and review into AI-assisted web development.

2 Vibe Design Assistant (VDA) System

VDA is built on top of Dyad[3], an open-source vibe coding platform that provides the prompt execution environment. VDA maintains two persistent artifacts: the Design Intent Representation and the Design Heuristics Repository.

2.1 Design Intent Representation (DIR)

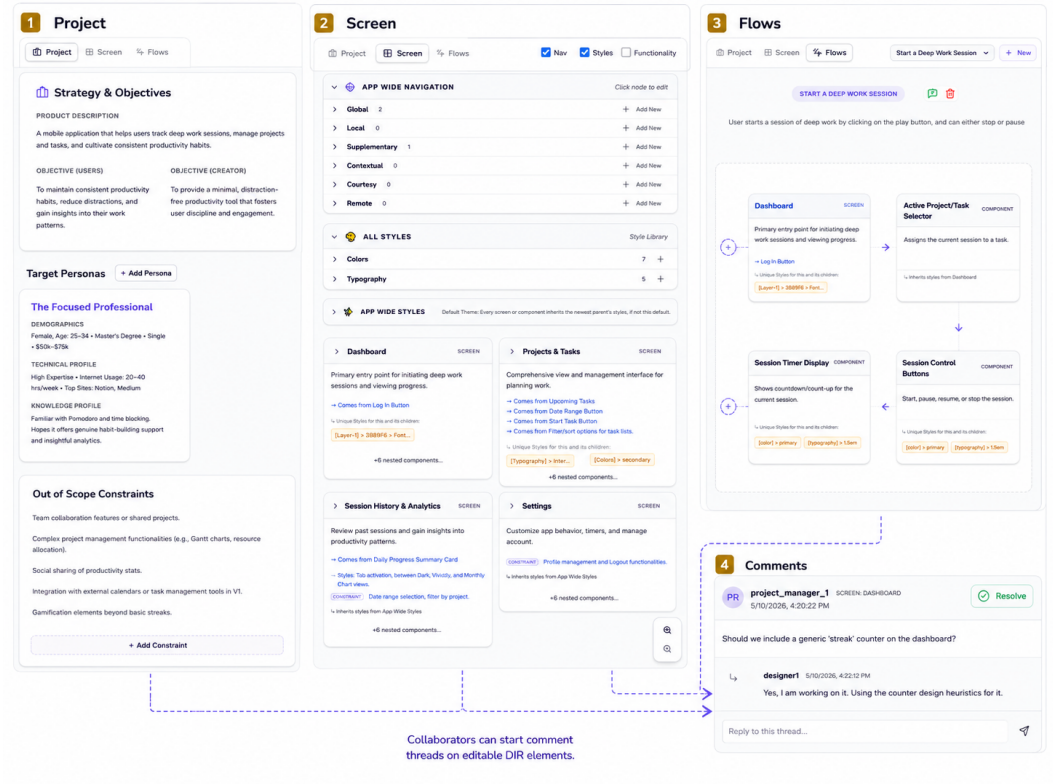

Fig. 2. **Design Intent Representation as a Shared Collaboration Artifact.** The DIR UI organizes project intent into three editable views: Project, Screens, and Flows. (1) The Project view captures product goals, target users, scope, and constraints. (2) The Screens view represents the application through screens, components, navigation patterns, styles, and associated functionality. (2) The Flows view represents key user journeys as editable node-based diagrams, allowing collaborators to inspect, revise, or rearrange interaction steps. (4) Because comments are attached to specific DIR elements, collaborators can discuss design intent directly rather than reconstructing decisions from prompt history, generated code, or application previews.

The Design Intent Representation (DIR) is VDA's central artifact. It externalizes what is usually hidden in prompt history, including the application's goals, users, screens, components, flows, styles, interactions, and out-of-scope constraints. DIR is not simply a project specification or

[3]https://www.dyad.sh/

documentation layer, but a persistent coordination artifact for AI-mediated software creation. In current vibe coding workflows, design rationale, implementation assumptions, and evolving project intent are often embedded within transient prompt histories that are difficult for collaborators to inspect, interpret, or negotiate. This creates challenges for shared understanding across collaborators with different expertise, such as newcomers, domain experts, designers, and developers, who often operate at different levels of abstraction and may not directly participate in prompting interactions. DIR externalizes design intent into a shared, editable representation that supports communication, negotiation, and accountability across both human collaborators and AI-assisted generation processes.

DIR is created through the Ideation Interview, the first user-facing step in VDA's workflow. In this interview, VDA uses Garrett's five planes (strategy, scope, structure, skeleton, and surface) [2] to help newcomers think more deliberately about their project. This interview strengthens the generative quality of the prompt–preview loop by shifting early interaction away from implementation and toward conceptual design intent, helping reduce premature design lock-in [5] and supporting more detailed prompt generation later.

As the project evolves, DIR maintains continuity between evolving implementation decisions, AI-generated changes, and collaborator contributions, helping collaborators inspect how design intent shifts over time. This helps address a key challenge in vibe coding: making authorship, accountability, and transparency visible across the design process [4, 6, 8].

2.2 Design Heuristics Repository

Decades of HCI and design research have produced a large body of design heuristics. However, this design knowledge is often inaccessible to newcomers while they are prompting. Recent HCI systems embed domain-specific quality criteria, such as accessibility [7] or sustainability [1] directly into AI-assisted development workflows, however, these efforts remain fragmented and are not integrated into a broader collaborative workflow.. To address this gap, VDA maintains a repository of design heuristics encoded as filterable Anthropic Skills[4] and retrieves guidance relevant to the current work during any collaboration.

2.3 Prompt Improvement and Guided Build

VDA supports both local prompt refinement and systematic implementation review. In Prompt Improvement, users provide a raw prompt, and VDA asks clarifying questions, applies relevant design heuristics, and generates a more complete implementation prompt for Dyad. In Guided Build, VDA compares the current codebase against the DIR and relevant heuristics, then generates prioritized build tasks for missing features, partial implementations, or heuristic violations. Users can send these tasks to Dyad, edit them manually, discuss and refine them with VDA, or verify completion after the codebase changes.

3 Vibe Collaboration Scenario

Consider a teacher using a vibe coding platform to create a classroom activity dashboard for students. The teacher can describe the educational goals of the dashboard, the kinds of activities students should complete, and the classroom workflows the tool should support. However, they may struggle to communicate interaction details, navigation structure, accessibility expectations, and implementation constraints through prompts alone. As the AI-generated application evolves, a designer may join to refine the navigation and interaction flow, while a developer may review feasibility, missing functionality, and implementation risks. In a conventional vibe coding workflow,

[4] https://github.com/anthropics/skills

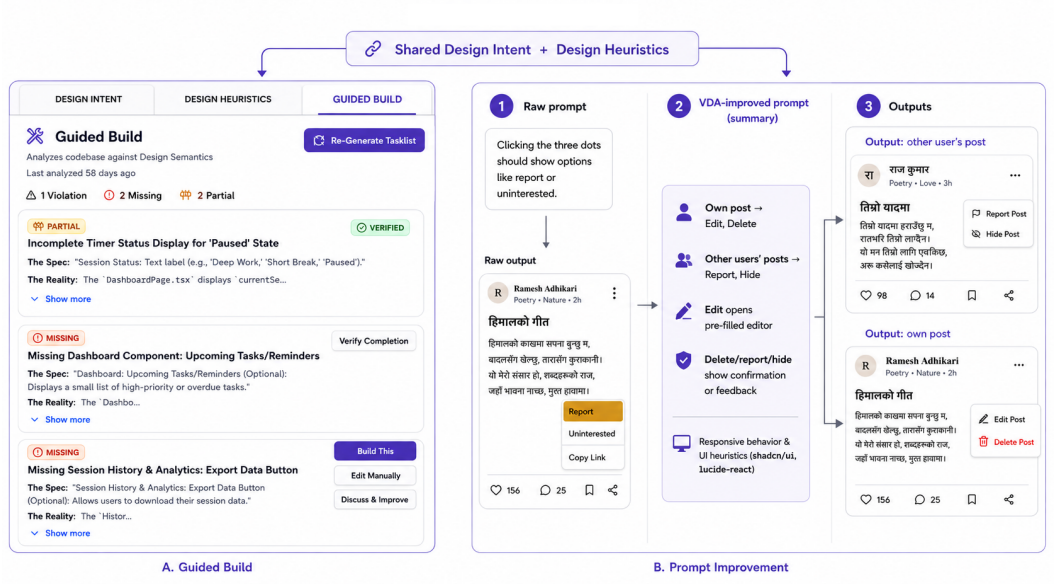

Fig. 3. **(A) Guided Build.** VDA analyzes the codebase against the DIR and classifies gaps as missing features, partial implementations, or heuristic violations. Each task can be sent directly to Dyad through **Build This**, edited manually, or refined through **Discuss & Improve**. After Dyad modifies the codebase, the user can select **Verify Completion**. VDA then rechecks whether the task has been implemented. If not, it can generate a follow-up prompt targeted to the remaining gap. **(B) Prompt Improvement Walkthrough.** A raw prompt for a social media post menu produces a limited dropdown because key interaction requirements, such as ownership states and destructive actions, remain implicit.interface changes, such as refining the behavior or presentation of a single component. VDA reframes the request around post ownership, destructive actions, and feedback behavior, producing a more complete interface implementation.

these collaborators must reconstruct project intent from fragmented prompt histories, generated outputs, application previews, and partially implemented code. Prior decisions are often difficult to inspect: collaborators may not know why a certain screen was added, whether a feature reflects the teacher's original goal, or how the application's intent has shifted across iterations. This creates breakdowns in shared understanding. The teacher may describe a "student progress view" in terms of classroom use, the designer may interpret it as an information architecture problem, and the developer may understand it as a data and state-management problem. Without a shared representation of intent, collaboration becomes centered on interpreting outputs rather than negotiating the design rationale behind them.

This problem becomes more consequential in larger enterprise projects, where vibe coding may involve product managers, designers, developers, domain experts, compliance reviewers, and other stakeholders working across different stages of a product lifecycle. In such settings, the challenge becomes preserving shared understanding as product goals, user flows, constraints, and implementation decisions evolve over time.

VDA addresses this limitation by positioning the Design Intent Representation (DIR) as a boundary object for collaborative vibe coding. Product managers can define goals and feature scope, domain experts can add contextual requirements, designers can refine screens and flows, developers can review feasibility and implementation completeness, and reviewers can comment on risks,

constraints, or compliance concerns. Because these contributions are anchored to the same structured representation of design intent, VDA supports continuity across handoffs, reduces the need to repeatedly reconstruct project rationale, and makes AI-assisted development more accountable in collaborative settings.

References

[1] André Barrocas, Nuno Jardim Nunes, Valentina Nisi, and Nikolas Martelaro. 2026. EcoAssist: Embedding Sustainability into AI-Assisted Frontend Development. In *Proceedings of the 2026 CHI Conference on Human Factors in Computing Systems (CHI '26)*. Association for Computing Machinery, New York, NY, USA, Article 589, 16 pages. doi:10.1145/3772318.3791330

[2] Jesse James Garrett. 2010. *The Elements of User Experience: User-Centered Design for the Web and Beyond* (2nd ed.). New Riders Publishing, USA.

[3] Francis Geng, Anshul Shah, Haolin Li, Nawab Mulla, Steven Swanson, Gerald Soosai Raj, Daniel Zingaro, and Leo Porter. 2025. Exploring Student-AI Interactions in Vibe Coding. arXiv:2507.22614 [cs.HC] https://arxiv.org/abs/2507.22614

[4] Jie Li, Youyang Hou, Laura Lin, Ruihao Zhu, Hancheng Cao, and Abdallah El Ali. 2025. Vibe Coding for UX Design: Understanding UX Professionals' Perceptions of AI-Assisted Design and Development. arXiv:2509.10652 [cs.HC] https://arxiv.org/abs/2509.10652

[5] Tianyi Li, Tanay Maheshwari, and Alex Voelker. 2025. User-Centered Design with AI in the Loop: A Case Study of Rapid User Interface Prototyping with ""Vibe Coding"". arXiv:2507.21012 [cs.HC] https://arxiv.org/abs/2507.21012

[6] 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

[7] Peya Mowar, Yi-Hao Peng, Jason Wu, Aaron Steinfeld, and Jeffrey P Bigham. 2025. CodeA11y: Making AI Coding Assistants Useful for Accessible Web Development. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems (CHI '25)*. Association for Computing Machinery, New York, NY, USA, Article 45, 15 pages. doi:10.1145/3706598.3713335

[8] Veronica Pimenova, Sarah Fakhoury, Christian Bird, Margaret-Anne Storey, and Madeline Endres. 2025. Good Vibrations? A Qualitative Study of Co-Creation, Communication, Flow, and Trust in Vibe Coding. arXiv:2509.12491 [cs.SE] https://arxiv.org/abs/2509.12491

[9] 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

[10] Sverrir Thorgeirsson, Theo B. Weidmann, and Zhendong Su. 2026. Computer Science Achievement and Writing Skills Predict Vibe Coding Proficiency. In *Proceedings of the 2026 CHI Conference on Human Factors in Computing Systems (CHI '26)*. Association for Computing Machinery, New York, NY, USA, Article 947, 17 pages. doi:10.1145/3772318.3791666