builder.io

How to choose the right vibe coding solution for your enterprise

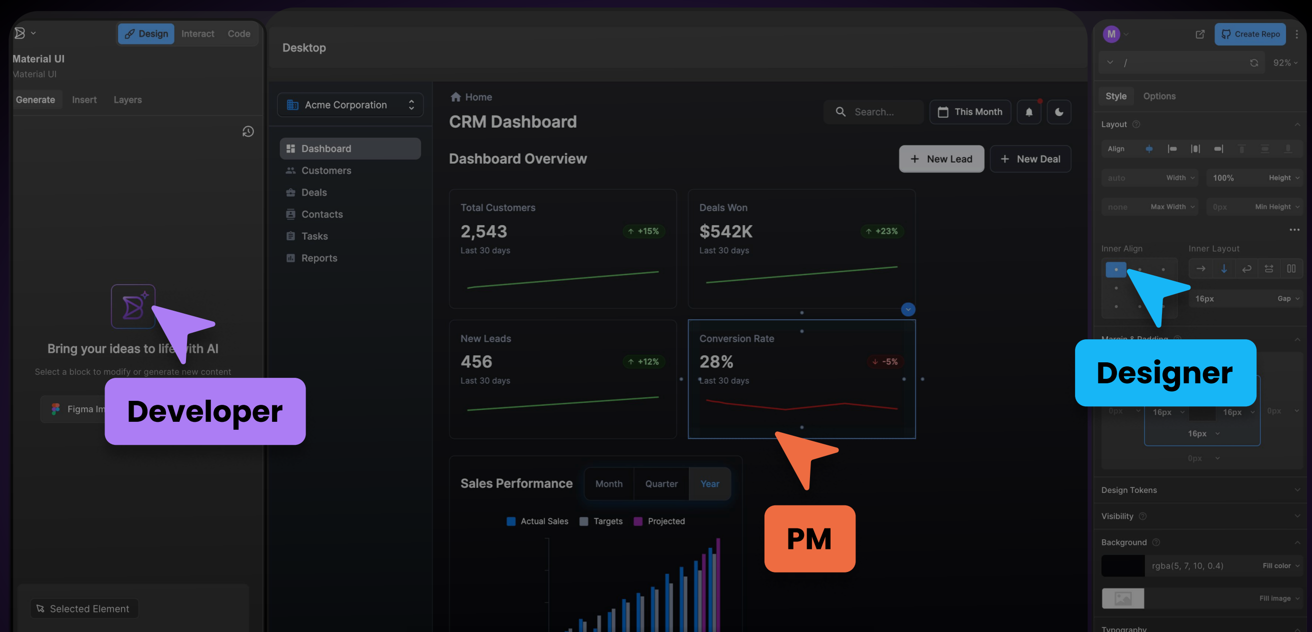
Every day, thousands of developers leverage vibe coding solutions to quickly spin up prototypes and web apps.

Many “vibe coding” apps are impressively powerful, but they don't often meet the security and governance requirements enterprises have or suit production use cases.

Builder recently concluded interviews with users in different organizations who have tried multiple AI “vibe coding” solutions. In almost every case, they leveraged them for personal or hobby projects but virtually none of them used these tools to accomplish tasks at work.

The major limiting factor they cited was that the solutions they tried didn't integrate with their organization's design system, code, or existing workflows.

To help you evaluate vibe coding solutions we borrowed a framework from Forrester Research. This workbook will ensure the solution you POC doesn't just work at the hobbyist level, but that it can also scale to accelerate the entire software development lifecycle in your organization.



The vibe coding landscape

From production-grade IDEs to lightweight app builders, here's how the major categories compare and where they fit in your workflow.

	AI-powered IDEs	AI visual IDEs	AI app builders
Primary users	Developers	Product managers, designers	Founders, solopreneurs, hobbyists
Interface	Code-first	Visual-first	Visual-first
Workflow	Developer workflows only	Designer-developer collaboration	Solo creator workflows
Tech stack compatibility	Any stack/framework	Any stack/framework	Restrictions on stack/framework options
Common use cases	• Full-stack development • Complex applications • Backend services	• Front-end development • Rapid prototyping • Design-to-code workflows	• MVPs • Personal projects • Simple greenfield web app
Example offerings	Cursor, Claude Code, GitHub Copilot	Fusion, by Builder.io FUSION	Bolt, Lovable, V0, Replit

builder.io

Evaluate vibe coding solutions across four categories:

CATEGORY	DESCRIPTION + WHY IT MATTERS
Context providers	Context providers are integrations you can plug into AI to give it context. Leading teams focus less on “prompt engineering” and more on “context engineering.” They pull internal organizational context into AI to boost understanding and output quality.
Ease of use	“Quality of generation” is hard to benchmark as it often depends on context. As a proxy, assess “ease of use.” Avoid tools that trap you in endless prompt revisions. Hobbyist solutions excel here, enterprise tools should too.
Developer experience	Even if the tool will be used mainly by non-developers, you must evaluate the developer experience. To truly speed the software development lifecycle, devs must easily integrate context, review, test, manage, and work with vibe-coded experiences.
Security and scale	It’s not just about SOC compliance, SSO, and AI training exclusion. To scale any solution across a global organization evaluate the collaborative features, ability to set up and iterate on org-wide templates, and RBAC.

You can use the following scorecard to evaluate when shortlisting a vibe coding solution for your enterprise:

Download the template

builder.io

How to POC a vibe coding solution in your enterprise:

We recommend a seven step approach to POCing a vibe coding solution for your enterprise:

	TASK	DETAILS + RECOMMENDATION
1	Define the roles and category of vibe coding solution you want to start with	We recommend POCing: • IDEs with developers • Visual IDEs with Product Designers and Product Managers
2	Define the workflows you want to optimize	We recommend POCing: • **For developers:** Generating a UI from scratch and generating a new feature from scratch • **For UX designers and product managers:** Generating a prototype from scratch and visually iterating the front-end of an existing feature/UI • **For UX designers in a design system team:** Visually iterating a code component in a component library, creating a new code component in a component library, visually adding new test cases to a Storybook
3	Set up your environment context for success	This should include: • Giving team members a starting point that includes your existing design system including code components, styles, etc. • If you're leveraging Figma designs, use Figma designs that use auto-layout and design best practices • Including relevant MCPs. For example, if your Product Managers work in Jira, directly connect to Jira via an MCP

builder.io

	TASK	DETAILS + RECOMMENDATION
4	Educate your team on best practices for prompting	Best practices include: 1. Using screenshots 2. Mentioning components by name (if you know them) 3. Selecting the exact portion of the experience you want to iterate 4. Use Figma designs to provide context 5. Leverage MCPs to save time downloading or copying and pasting large context 6. Saving custom instructions for things that keep coming up
5	Use real repos (if possible)	We realize in the POC stage some organizations cannot connect their real repos to a vibe coding solution, but there's something immensely powerful for a non-developer to be able to iterate a real codebase and see something they created show up in their customer's/end user's workflow.
6	Setup a Slack/ Teams/etc. channel and share video recordings of experiences	Sharing video recordings of your team's experience in-context can help others chime in with support, celebrate accomplishments, or gather feedback to bring to your vendor. This is a best practice among engineering leaders we've talked to who are evaluating AI coding tools.
7	Quantify impact across time to market, resolved features/ tickets, and efficiency across the whole team	Some metrics we've seen include: • PRs shipped by non-developers • Ticket resolution time • Lines of code generated • Developer/designer/PM efficiency Even if you're exclusively POCing workflows for product managers or UX designers, it's possible that the majority of the productivity benefit could actually be in the development phase; be sure to measure accordingly.