Chapter 2. User Interface and User Experience Design

The world of software applications depends on beautiful *user interfaces (UI)* and intuitive *user experiences (UX)*. UI design is about how a product looks and how users interact with it visually. It focuses on layout, colors, buttons, typography, and everything that shapes the visual experience of a software product. UX design, on the other hand, is about how a product feels to the user and if it is intuitive and efficient to use. UX is about usability, navigation flow, and making sure the overall experience is smooth and user-friendly. Both work hand in hand to create great software products.

Producing those interfaces and experiences has been the job of UI/UX designers for the last few decades. They usually pair with software engineers to implement the functionality and the backend workflows that bring those designs to life as an integrated experience for users around the world. Transforming UI/UX designs into functional frontend code is a labor-intensive process, though. Traditionally, designers and engineers work closely to ensure that the visual aesthetics of a design are accurately translated into a responsive, interactive user interface. This process often involves multiple iterations, meticulous attention to detail, and a deep understanding of both design principles and coding standards.

Since the emergence of text-to-image generative AI models in 2022 and 2023, UI/UX design has witnessed an enormous transformation. As I write this in 2025, recent AI tools can now generate UI designs from ideas described in natural language *and* generate functional frontend code from UI designs. These tools are revolutionizing the design-to-code workflow by automating significant portions of the process. This has immense potential to reduce the time and effort required to bring designs to life and make design professionals much more efficient. AI tools can significantly reduce the gap between concept and implementation.

For example, the first iteration of wireframing and sketching designs for a new application can take more than a week to complete—sometimes more

than a month, depending on the complexity of the project. The new AI design generators can cut wireframing time to less than a day and make each feedback loop faster as well. These tools' ability to transform designs into functional frontend code means one person or team can go from describing an idea in natural language to generating functional frontend code within minutes (or hours, counting iterations), as opposed to weeks or months.

This chapter will examine some of the leading tools in both the domains of UI design and UX design, aiming to cover the wide range of activities in between.

Types of AI Tools for Design and Frontend Development

Throughout this chapter, I'll focus on two types of AI tools: those that transform ideas into designs and those that transform designs into code.

Tools that transform ideas into designs

This sounds like a UI/UX designer's job, but think about it: if creating designs becomes easier, faster, and more accessible to people without professional design skills, the average frontend developer will probably be able to create designs *and* code, as opposed to the current scope of work, which entails only writing code. That translates to massive empowerment for software engineers, especially those working on the frontend, who'll be able to create basic designs independently.

Tools that transform designs into code

This segment complements the tools covered in Chapter 1, which generate code from a natural-language prompt. The tools here generate code from image-based mockups or Figma design files. (If you aren't familiar with it, Figma is for UI/UX designers what Visual Studio Code is for software developers.)

Most available UI/UX design generator tools are fully browser- and cloud-based. They are simple to use without complex software installation and are easily accessible with a browser and a URL. Some tools, like QoQo.ai, come as plug-ins for design platforms like Figma.

Most of the products this chapter covers have been launched very recently and are in the early stages of product development. They'll undergo continuous development, rolling out new features and improved functionality in the months and years ahead. As I write this, such design tools are a year or two behind software development tools like GitHub Copilot in terms of maturity.

I expect that in the UI/UX space we'll also see more browser-based tools, more add-ons or plug-ins to popular design platforms, and more integration into collaboration tools used between designers and frontend development teams. (The analogous tool to Git might still be Figma for many teams, but for others it might be plug-ins to popular collaboration tools like Jira or Notion—we'll see.)

Pros and Cons of Using AI Tools for UI/UX

Like any tools, the tools in this chapter present advantages and disadvantages. Some of the advantages include:

Efficiency and speed

AI tools can process design files and components and generate corresponding code in a fraction of the time it would take a human developer. This rapid turnaround can accelerate project timelines and reduce costs.

Consistency

By automating the conversion process, AI tools help maintain consistent implementation of design elements across a project.

Accessibility

Individuals without technical or design expertise can pick up these tools and create functional interfaces, a shift that helps to democratize UI/UX design.

Prototyping and iteration

AI-generated code allows for faster prototyping. Designers and engineers can quickly test and iterate on their ideas to bring new

products to market much faster than ever before.

The drawbacks of the tools discussed in this chapter include:

Code quality

Code produced by AI tools does not always meet the standards of human-generated code. It can be less efficient and harder to read and maintain, so—as I've emphasized throughout this book—developers must look through and revise generated code thoroughly before releasing it into production.[1]

Device compatibility

AI-generated code may not be fully optimized for all browsers, operating systems, and screen sizes. This can lead to issues with responsiveness and cross-device compatibility that require additional effort to troubleshoot and fix.

Customization and flexibility

While AI tools can handle standard design patterns well, they may struggle with more complex, custom design elements. This greatly limits creativity and the ability to tailor the design to specific needs. For this reason, human creativity is still needed.

Lack of originality and uniqueness

AI tools often produce designs that resemble existing ones. These tools are trained on current designs and prevailing trends, which can lead to a lack of originality and distinctiveness, with some designs appearing generic or repetitive and failing to stand out. Although AI tools can offer a useful starting point or inspiration, they're far from matching the creative capabilities of a talented human designer.

Striking the right balance between leveraging AI for automation and relying on human talent for quality and creativity is key to successfully integrating these tools into the design workflow.

Use Cases for UI/UX AI Tools

UI/UX design-assistant tools aid different parts of the design process. They can be categorized as follows:

- Design ideation assistants
- User research and analysis tools
- Color palette and style generators
- Layout and component generators
- Accessibility checkers
- Personalization engines
- Voice-user interface (VUI) designers
- Gesture and motion design tools
- Wireframing and prototyping tools
- Design-to-code converters

This chapter focuses on the last two items, which lie at the intersection of UI and frontend code and are thus of most interest to software engineers. Other tools that target specific parts of the UX and product research workflows are usually more specific to designers and other roles that collaborate more loosely with software developers, like product managers, product analysts, and UX researchers.

Thus, I evaluate the tools in this chapter using the following use cases:

Rapid prototyping

AI tools can quickly generate wireframes and prototypes based on initial inputs and design descriptions. This is a major chunk of the design process, where designers spend hours in back-and-forth iterations. With these tools, creating prototypes is much faster and designers can iterate multiple design options quickly. This flexibility can also make sophisticated flows such as A/B testing much more accessible.

Designing from templates

AI tools can be trained on design principles like components, colors, and styles, to ensure consistency across different parts of a product by generating designs that adhere to established design systems, branding, and guidelines. This results in a cohesive user experience with uniform

design elements and styles throughout the application. While this "factory" approach doesn't foster creativity, it does bring significant value for use cases where all you need is to create new screens that look like the existing ones but fulfill new functionality. (In all honesty, for better or worse, most software development consists of this "factory" approach, as proven by the growing popularity of low-code and no-code tools.)

Accessibility assessments

AI tools can automatically check and improve the accessibility of UI designs for people with disabilities, ensuring compliance with standards like the Web Content Accessibility Guidelines (WCAG) and inclusive design practices—much as the tools covered in Chapter 3 do for software code. Such features can identify accessibility issues and suggest fixes, reducing the manual effort required.

User research and personalization

AI UI/UX design tools can analyze user data about behavior, needs, and preferences to tailor their output to specific user groups or individuals, improving engagement, retention, and satisfaction. By leveraging data-driven insights, these tools ensure user-centered design. This can massively extend existing use cases. For example, if the goal is to create new landing pages for specific customer demographics, with these tools you could create full product variations according to specific parameters. Without them, each product variation would require a heavy software development cycle.

Content generation

AI tools assist in generating content for UI elements, such as placeholder text, images, and icons. This quickly populates designs with realistic content, making prototypes more lifelike for testing and ensuring consistency with the design's style and tone. This frees designers to focus on the more creative aspects of the process.

Evaluation Process

My process in this chapter was as follows: I submitted a brief design prompt to each of the selected UI design tools, ran the same challenge on each tool,

and compared their output. I also evaluated their ability to convert the designs they generate into frontend HTML CSS, Next.js, and React code. I reviewed 20+ tools available in the market. As expected, they followed a long-tail distribution: a few worked as advertised, while others delivered poor output or had different issues that prevented me from running the test. Many tools in the AI space were developed by startups whose products are in very early stages of development.

After trying the tools, I selected the best four in each category that provided actual value for the use cases tested. I then rated each tool on a scale from 1 to 10, with 1 being the worst (a solution that errors out and doesn't run at all), 10 being flawless, and 5 being a design that didn't exactly fit the requirements. I look closely at the top product in each category, detailing its pros and cons, then provide more concise information on the runner-up.

It's also important to note that all tests described in this chapter were run in spring 2025. Given the fast pace of evolution of each of these tools and their underlying models, it's likely that you could get a different result at a later time for the same prompt.

UI Tools

I asked each of the AI tools covered in this chapter to generate screen designs for a mobile food delivery application. Here is the detailed prompt I provided:

```
Create a user-friendly food delivery app that allows cu
local restaurants, view menus, place orders, and track
in real-time. The app should include these screens: Log
and Menu Browsing, Order Placement, and Tracking.

Style: App should be modern and minimalist, focusing or
visual appeal. The design should incorporate a clean la
mouth-watering food images.
```

Let's see how the tools did.

Uizard

Uizard has a marketing slogan that I believe sets the standard for most other players in this space: “Forget no-code, here comes no-design.” The analogy between code automation and design automation makes a lot of sense. Just as no-code has promised the same functionality in a fraction of the time and budget, no-design could have an equal impact on the design process.

The Uizard product started as a machine learning research project called pix2code in 2017 in Copenhagen, Denmark. According to Uizard’s marketing collateral, “It allows developers to visualize product ideas quickly and easily with AI. This helps to fast-track the workflow of designers and software developers. Its mission is to democratize design and empower non-designers everywhere to build digital, interactive products.”

At the prompt screen (Figure 2-1), I asked Uizard to generate screens for a food delivery app using the prompt from the preceding section.

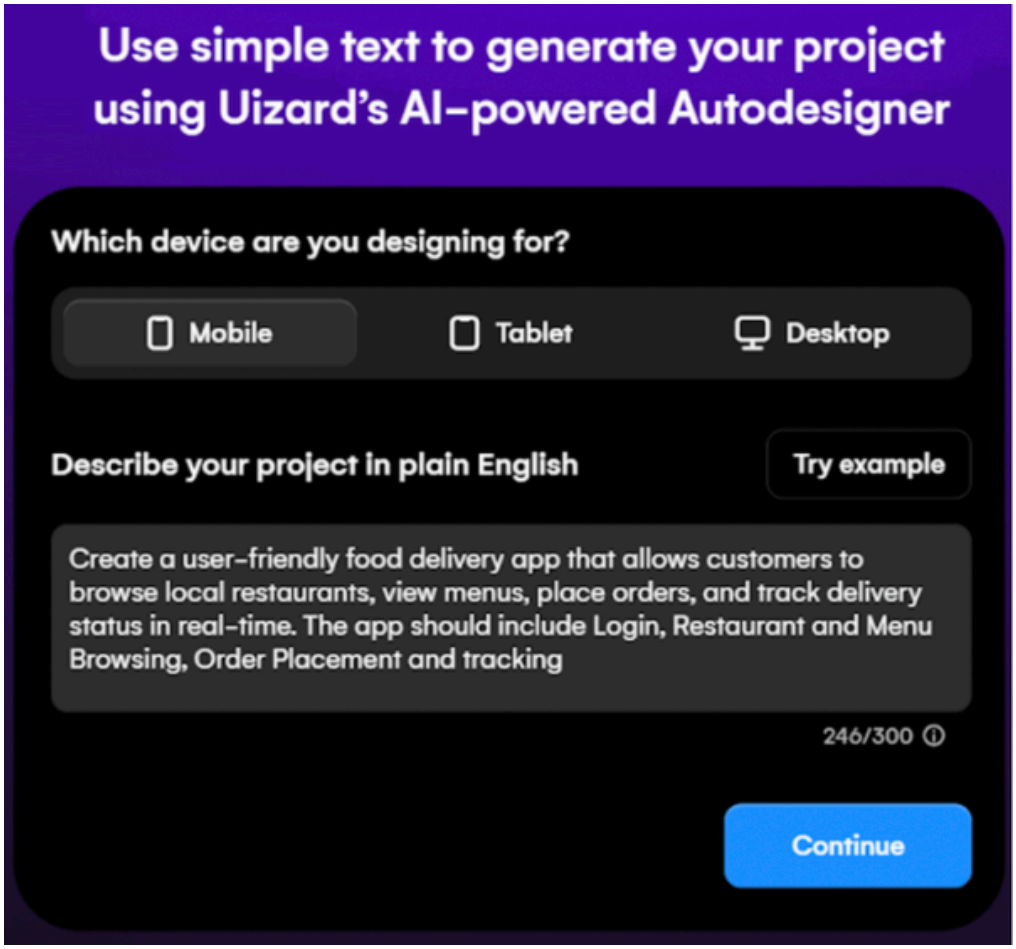

Figure 2-1. The Uizard Autodesigner prompt screen

Figure 2-2 shows the designs it generated.

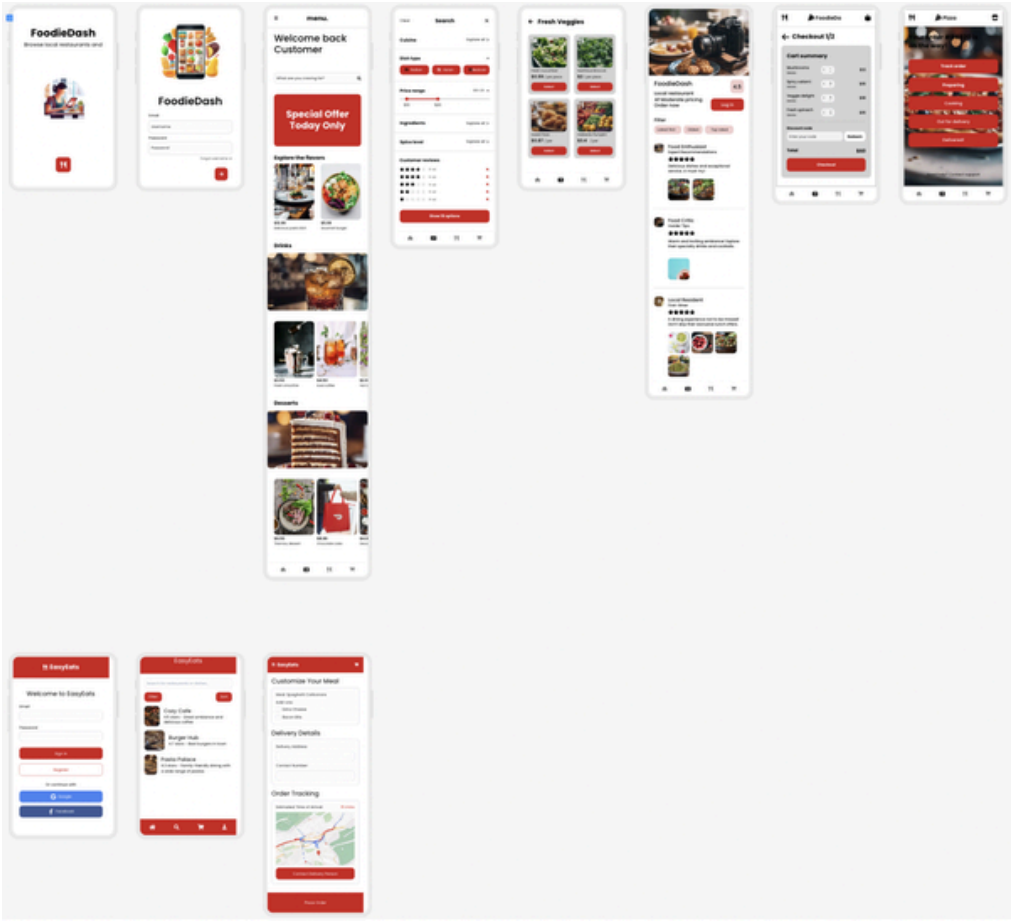

Figure 2-2. Screens generated by Uizard (11 in total)

You can see all of the screens it generated, along with the interactions, as previews on Uizard's site.

One of Uizard's unique features is that it creates an interactive demo of the designs, with actual interactions and transitions between screens. This makes it a go-to website for transforming an idea into a demonstrable concept, even before writing any code. On the other hand, Uizard doesn't generate code to the level of other tools in this space, such as Bolt.new or Lovable, which generate working code alongside the designs and make it easy to commit the code to GitHub and even to deploy the working app right away.

I rate Uizard an 8 out of 10. Its design component is very competent, but it falls short in generating working code.

Bolt.new

Bolt.new is a browser-based, AI-powered development tool launched in October 2024 by StackBlitz. It enables users, including those without technical skills, to create frontend web and mobile applications using natural-language prompts. By integrating technologies like WebContainers, Supabase, and Netlify, Bolt.new allows for seamless code generation, authentication, and

deployment directly from the browser. This approach simplifies the development process, making it accessible to a broader audience.

The platform has experienced rapid growth, achieving $20 million in annual recurring revenue (ARR) within two months of launch and reaching $40 million ARR by February 2025.[2] With over three million registered users and one million monthly active users, Bolt.new has become one of the fastest-growing AI tools in history. Its success is attributed to its user-friendly interface, efficient development workflow, and the ability to swiftly transform ideas into functional applications.

I gave the same prompt to Bolt.new, and it generated the app designs seen in Figure 2-3.

The screens generated are very solid, with the pages, navigation, and hierarchy I'd expect in such an application. It generated realistic mock text and images for each screen, making them look good. There's an obvious glitch in the footer, with both missing icon images and creeping navigation, but other than that the designs are very good.

What's best about Bolt.new is that it can help *beyond* generating designs from a natural-language prompt: it also generates the underlying code. In this case, it generated frontend code in TypeScript.

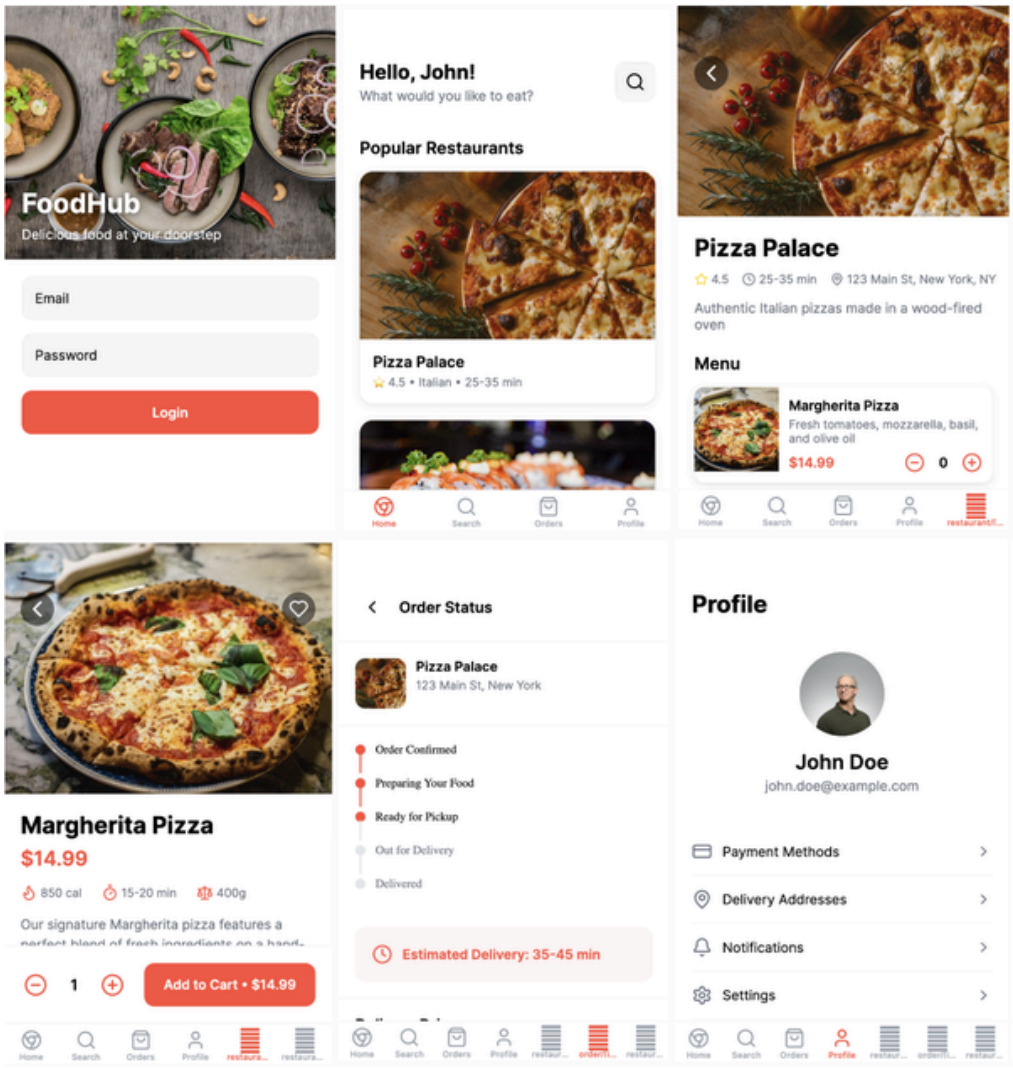

Figure 2-3. Screens generated by Bolt.new

As seen in Figure 2-4, the Bolt.new workspace consists of:

- A prompt area on the left, where I can ask for new screens, changes, fixes, etc.
- A code editor, where I can see the actual code being generated and make live changes
- A console, where I can see errors, logs, and anything else resulting from the code execution
- A preview area where I can see all screens, such as the screenshots in Figure 2-3
- Options to export and deploy my application

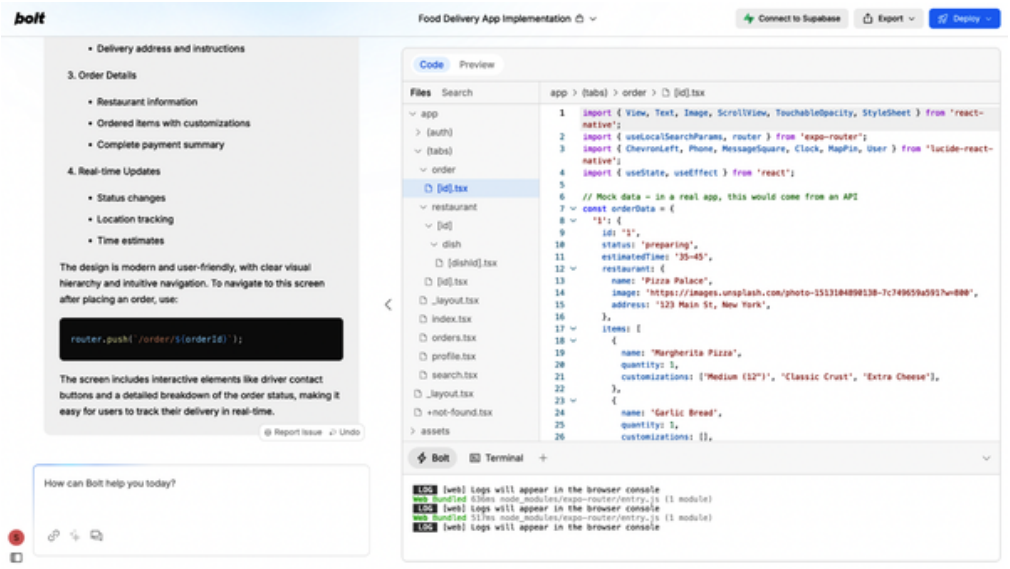

Figure 2-4. The Bolt.new workspace

Bolt.new is a one-stop-shop that aims to fully replace not just designers, but also IDEs and cloud providers—since, when I published my app, it was hosted on Bolt.new's servers. Given the comprehensive feature set, I rate Bolt.new a 10 out of 10.

Lovable

Lovable is a browser-based AI development platform launched in November 2024 by the Swedish startup of the same name. It empowers users, both technical and nontechnical, to build full stack web apps using natural-language prompts. By combining tools like Supabase for backend services and GitHub for version control, Lovable automates everything from database setup to deployment. The result is a dramatically simplified development experience, allowing anyone with an idea to launch functional software directly from their browser.

In just three months, Lovable reached $17 million in ARR and attracted over 30,000 paying subscribers. With more than 50,000 new projects started each day, it's one of the fastest-growing AI dev tools to date.[3] Much of its momentum comes from its intuitive interface and ability to abstract away the traditional complexity of software development, making the field—especially frontend interfaces—more accessible than ever.

I gave the same prompt for the food delivery app, and Lovable generated the app designs shown in Figure 2-5.

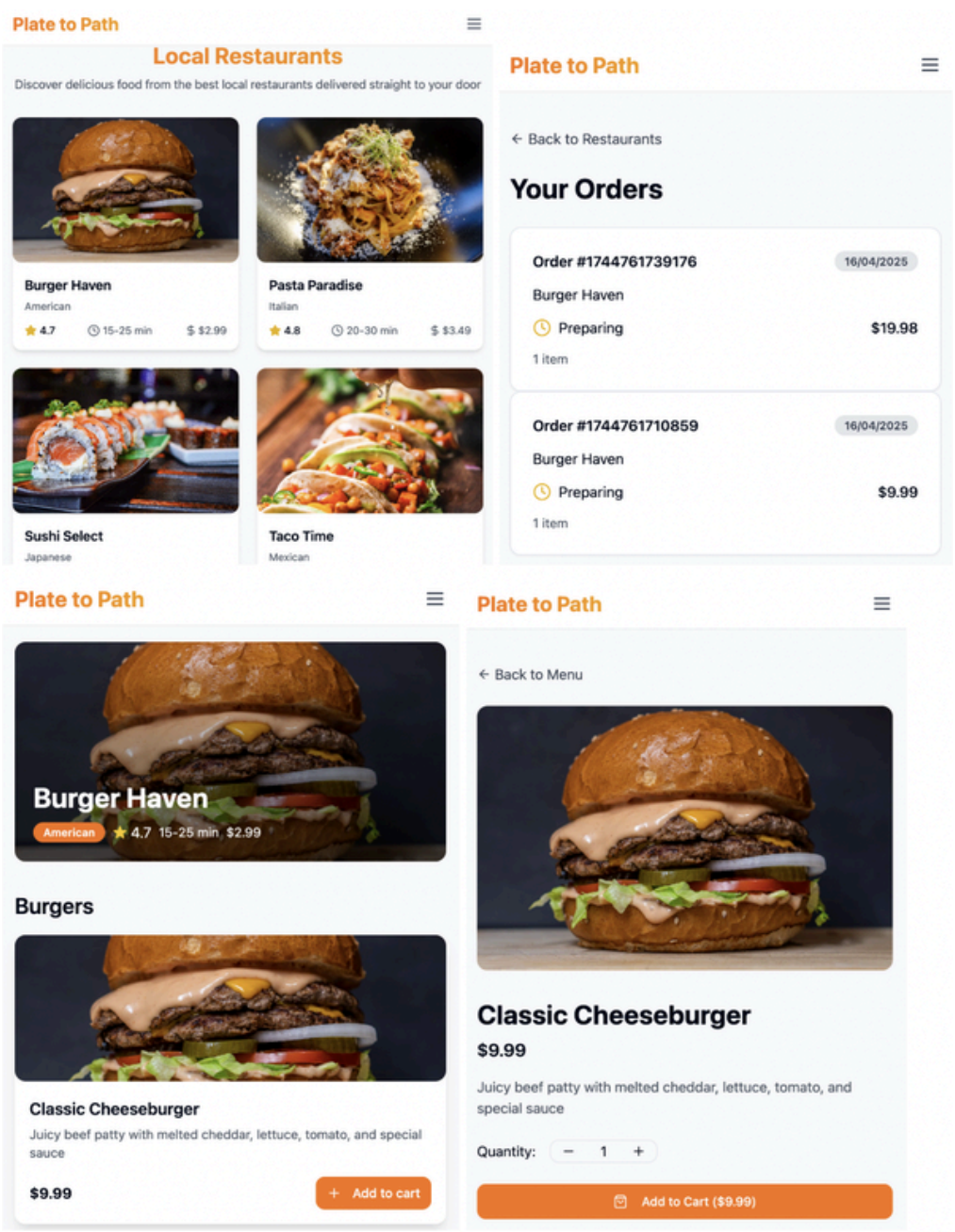

Figure 2-5. Screens generated by Lovable

Very similar to Bolt.new, Lovable has a chatbot UI on the left side of the screen, allowing the user to prompt with natural language for new screens, changes, fixes, etc. It also has a toggle on the top to switch between preview and "Dev mode." The preview mode shows the generated designs, as seen in Figure 2-5, while Dev mode shows the code produced. On the top, there are buttons that allow users to easily sync the code with GitHub or deploy the app.

I also rate Lovable a 10 out of 10. I think it's the leading player in this space alongside Bolt.new. Both fulfill the use case of getting professional designs and working code from a natural-language prompt.

UX Tools

In this short section, I cover tools for the UX part of the UI/UX design workflow: user research, product workflows, and usability.

Here is the prompt I used for evaluating UX tools:

```
Create a user persona for an 18-25-year-old university
to order food online from a restaurant using our applic

Description: 18-25, Female, Male, University undergradu
Los Angeles, California, single, and no children. Looki
user-friendly platform to order food very fast, track c
make payments
```

QoQo.ai

QoQo.ai (pronounced “cocoa”) is an AI-for-UX-design Figma plug-in that helps in the early stages of design. Designers can use it to generate UX personas, journey maps, site maps, and UX copywriting. UI/UX designers, product managers, and product researchers, who usually do these tasks, will save time with this tool; it can also make these tasks more accessible to software engineers, helping them better engage with users.

I provided QoQo with the preceding prompt, and within seconds the AI generated a detailed user persona (shown in Figure 2-6).

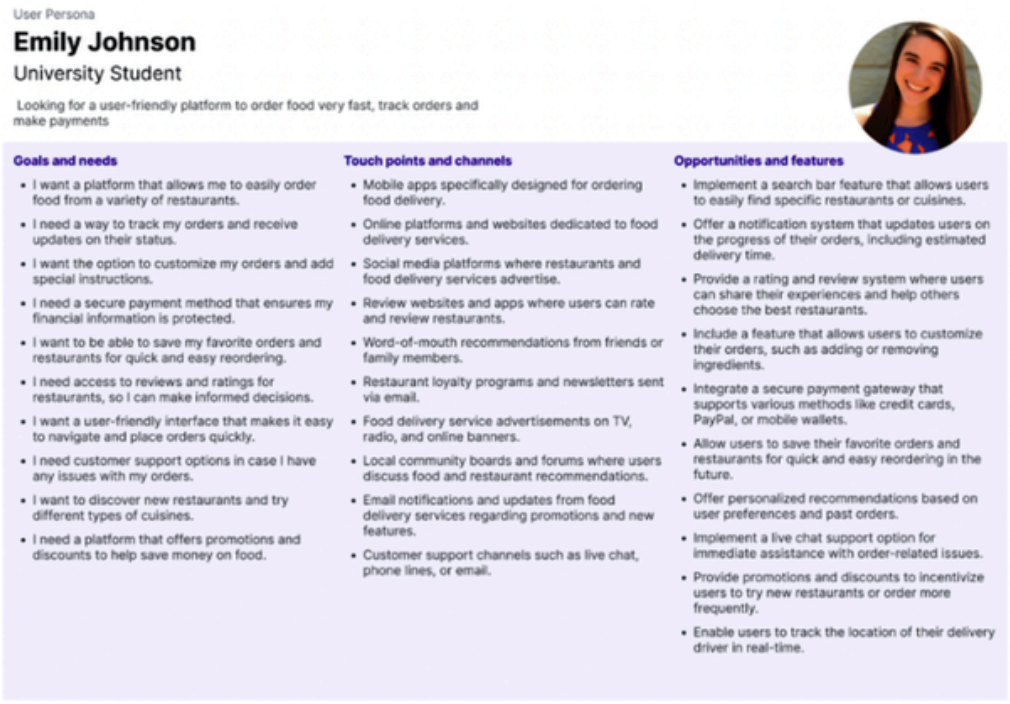

Figure 2-6. QoQo user persona

I rate QoQo an 8 out of 10. It gets the job done, and there is really no learning curve involved. It does feel like a GPT wrapper, in the sense that it's generating text and placing it into predefined brackets, which can provide tremendous value if you are starting to design the UI/UX of a software product.

Research Studio

Research Studio is an AI-powered application designed to automate the customer and user research analysis process. This tool is particularly useful for researchers and designers who need to transform user interviews into actionable insights, analyze satisfaction, and generate branded reports swiftly. Traditionally, UX researchers spend hours combing through vast amounts of data to extract insights. Research Studio streamlines this process, making it faster and more efficient.

Research Studio's platform allows you to choose between three open source LLMs: Claude AI, Mistral, or OpenAI's GPT-4o. You upload your user interviews and it extracts insights into a report. It can also use the information it has garnered from the interviews and data to write a report. Several report types are available as outputs, including user flows, feature maps, and question analysis.

To evaluate this tool, I used Claude AI (separately) to generate some raw input. I asked it to generate 50 survey responses from users of our dummy

food-delivery app, using the following prompt:

```
Generate 50 survey responses from users of a food deliv
thoughts after using the app for some time. The respons
positive and negative views, including any problems the
for features that could improve the app's efficiency ar
```

These responses included both positive and negative feedback, as well as suggestions for improvements. Here are a few examples from Claude's output:

```
1. "The app is fast and easy to use. Love the variety c

2. "Delivery times are often longer than estimated. Ple

3. "I wish there was an option to schedule orders in ac

4. "The customer service is excellent, always helpful w
```

I passed the responses into Research Studio to extract insights (Figure 2-7) and asked it to write a comprehensive report based on its built-in report types (Figure 2-8).

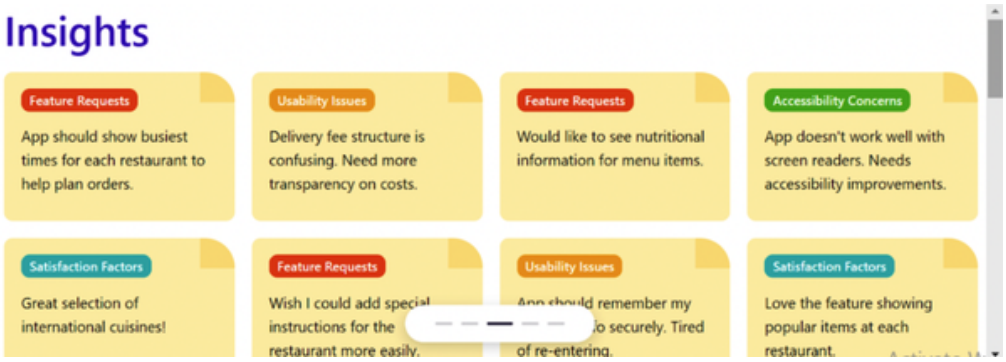

Figure 2-7. Research Studio transformed raw user reviews into a map of "insights"

Figure 2-8. An insight map generated by Research Studio from survey responses

I rate Research Studio a 7 out of 10. While it feels like a simple LLM wrapper, it does a good job with the workflow, and it does deliver value for anyone who wants to crunch massive amounts of data into structured product insights to use in their next software-development iteration.

Tool Comparison

Galileo AI emerged from my evaluation as the best text-to-UI tool among the 20+ tools I evaluated, including the four featured in this chapter. Its ability to take an image and remake it into a beautiful design also makes it stand out. As I've noted, this space is still in its very early days, so I can only imagine how far these tools will evolve in the coming years.

The UX tools I evaluated are less complex in terms of underlying technology, but that's not a critique: they worked well enough for their use cases.

Table 2-1. AI UI/UX tools overview

Tool	Purpose	Test performance
Uizard	UI generation	7/10
Bolt.new	UI and code generation	10/10
Lovable	UI and code generation	10/10
QoQo.ai	UX automation	8/10
Research Studio	UX automation	7/10

Conclusion

I began writing this chapter with the wrong expectations. I didn't expect the tools in this space to be very useful in their current state, given their newness. I was wrong. I'm now very excited about this space and the evolution in software development processes we'll see in the coming months and years.

Generating frontend code from designs, with the increasingly dominant use case of generating both design and working code from a prompt (as seen in Bolt.new and Lovable), is certainly the angle that best speaks to software

engineers, especially those working on the frontend. It is a game-changer that will save software engineers around the world millions of hours.

The other use cases seem to cater less to software engineers, as they are designers' work. But think about it: our tools are becoming more powerful, and if it takes less time to do the same work, software engineers can manage a larger portion of design workflows. Indeed, our employers and team leads will likely ask us to do so more and more.

These tools make software engineers capable of producing working designs, and given the staggering pace of their development, I can foresee that many teams will no longer need specialized designers to produce lots of screens. Of course, specialized interactions, branded UI elements, or complex user-experience nuances will still be best served by specialized human UI/UX designers. But I can imagine that most standard app flows, such as registration flows, listings, and profile pages, will be well served by these AI tools.

Never before have software engineers been so empowered to collect feedback from the users of their products, structure that feedback into actionable insights, generate improved designs, transform them into code, and push a new iteration to production. Product iteration cycles will shorten significantly thanks to these tools. Professionals who master them will have much more market demand than those who stick to only writing code. As I write these lines in early 2025, I'm seeing big growth in the Product Engineer job title in recent months, which caters exactly to this opportunity I'm describing: software engineers who can understand the client's needs, convert them into product feature descriptions, and ship the code that fulfils those descriptions.

[1] This is yet another occurrence of the trade-off between saving time writing code and spending more time debugging, fixing, and improving code. It's not much different, conceptually, from the trade-offs of the HTML code generated by Dreamweaver in the early 2000s, or the more recent advent of low-code tools that generate templated code, which is often hard to customize and adapt to one's specific needs.

[2] Akshay from Startup Spells. March 22, 2025. "Bolt.new: The 2nd Fastest-Growing Product in History (Behind ChatGPT)". *Startup Spells.*

[3] Stephane at Lovable. February 25, 2025. "Announcing $15m Added Funding to Build the Last Piece of Software". *Lovable* (blog).