Sara Cagle | 2020

Bridging the Gap Between Designers & Engineers

a how-to guide for designers

Introduction

The gap between designers and engineers can be tough to bridge, especially in larger organizations where roles may be siloed and less knowledgeable about each other's work.

As both a full-time software developer and a part-time designer, I have witnessed the challenges of both roles, and in particular, the challenges of working together. Much like any other group of people, the spaces that divide us can easily be reduced by employing **empathy** and working together to understand each other's point of view.

I have put together this guide to help bridge that gap, particularly for designers working with engineers.

This guide is broken into **what a designer should say, write, ask, do, and not do when working with engineers** and why each point is important.

I have also included brief **descriptions of common technical terms** for which designers may not be familiar.

Finally, no coin is one-sided, so I have included some **tips on how designers can encourage engineers to practice better empathy** toward their design partners.

I will use the terms *engineers* and *developers* interchangeably. I will also reference non-designers, which can be engineers or other partners such as supervisors, product managers (PMs), or other.

I will also use the terms *app*, *service*, and *product* interchangeably to mean the production-ready tool that engineers are building from the design

Table of Contents

1 Cover
2 Introduction
2 Table of Contents
3 Say
4 Write
5 Ask
6 Do
7 Do Not
8 Further Tips
9 Contact

Say

After you have drafted any key states and screens for your app, block your calendar! It's time to **arrange a meeting with the developers** and other stakeholders you will be working with. Here are some talking points you should be sure to hit.

Conduct a complete (in-person) walkthrough of your designs and wireframes with the engineers or the PM for the engineering team.

By having an in-person conversation, engineers can ask clarifying questions directly and early, and your clear communication can limit any misunderstandings from integrating into your design. If you are part of a large company, it may be challenging to meet with all the engineers. In this case, work with the PM to ensure he or she understands your design expectations and can answer questions for the team as your proxy.

Explain any design limitations (such as being restricted to one design language), design decisions (colors, fonts, organization), and design requirements (accessibility tooling, localization).

Engineers may not completely understand your design decisions, especially if they were driven by external requirements that might not be as clear to teams working in more siloed environments. Explaining these decisions will help build the engineers' empathy for you and reduce the likelihood that someone may make (what they perceive as minor) changes to your design as it's built.

Write

Written documentation will make or break the development process of your designs! Create **clear and specific documents** to share with the engineering team to ensure a smooth and on-time development schedule.

Create clear and detailed specs, including:

- Percentages and pixel values for layouts
- Expected gestures
- Plans for failing gracefully

There can be several ways to code a product for the same outcome. But tiny differences from these different approaches can stack up and cause a design to look inconsistent. Or in the case of unexpected product behavior, such as a failure, if you do not provide a design, developers may implement their own look which may not match your expectations.

Send a readonly copy of your designs to the whole team. Even better, use a central file share service to save and share with the team. Even if you are unable to meet face-to-face with the developers and must delegate to another non-designer, ensure that your designs are still shared to the developers directly.

In large teams, it's easy to lose track of documents, especially if they're only shared with one colleague at a time (such as a team PM or engineering manager). Keeping readonly documents in a central location will avoid accidentally leaving anyone out of the loop. (This can also help reduce team politics!)

Make sure the designs are readonly so non-designers don't accidentally or purposefully make changes.

Want developer feedback? Instead of allowing edits to the file, consider using a commenting system and encourage non-designers to share their thoughts.

Mark out your color palette, including **hex codes** for each color, any changes to opacity, and each color's expected usage.

Hex Codes – Special codes for colors that are used when implementing a design. There are many online tools that can generate hex codes for your designs.

Variables – Reusable values saved in the code. Global variables are available throughout the app.

For consistency throughout the app, developers can set these as **global variables** that can be changed at the snap of a finger if the design is updated later. Having these details well-defined makes simple changes easier for everyone involved!

Ask

A strong designer **knows where and how to ask questions** of his or her target users. In a way, the target users of a design are the engineers who will be building it! Leverage their knowledge to create more robust product designs.

Ask about common **architecture** in the app that could enable or hinder design plans. For example, is there a page that must be loaded in the background before all other pages?

A beautiful, simple design can be very complex under the hood! Changing things such as where a popup appears or navigation expectations between pages can be much more challenging than expected.
Understanding the underlying architecture at a high level can help you design for future functionality and scalability. Similarly, understanding the basic architecture can help you communicate the future design expectations with the engineers.

Architecture – The "bones" of an app, typically code that is challenging to update since it provides the stability of a backbone.

Ask where the data will come from. Is it user-submitted? Does it come from a database or other **API**?

Designing a static service is straightforward, but once you start including dynamic data into your experience, your designs must expand to include problems such as

- Data submitted incorrectly by users
- Network issues when getting data from an API
- Missing data from a database
- Latency problems from external sources

A high level understanding of where app data is sourced can help you better prepare for and design error scenarios.

API – An interface used by developers to fetch data from other services. For example, Instagram's API allows developers to check how many likes a photo has.

Telemetry – Logs that an app creates and keeps about usage and performance. Typically stored in large databases and reviewed on custom-made dashboards. May or may not be anonymized.

Ask about **telemetry** in the app. Are there any trends engineers have noticed that aren't apparent? For example, are users clicking a button at a surprising rate? Are users frequently navigating to one page before another?

While PMs and general user knowledge can help with your design, telemetry in the app can highlight trends that may not be clear from an external view. This information can help optimize smaller things in your design, such as button placement or navigation labels. Such specific telemetry often remains locked inside the engineering silo—reach out and ask to use all the tools at your disposal!

Do

Beyond arranging meetings, drafting written documents, and asking the right questions, there are a few more things you should **consider as a designer working with engineers**. Keep these in mind throughout the life cycle of your design process.

Design for reusability.

Reusable components in a design means reusable code. With less code, the product can be made faster and have a more consistent experience. Consider using or creating a design system. Many existing design toolkits have libraries of reusable components read to go.

Design for scalability.

Developers are also likely writing code for scalability—a scalable design to pair with scalable code will future-proof the product if its usage suddenly expands from one thousand users to one million. For example, do you have a plan for adding additional lines to a table, or further items on a navigation bar?

Familiarize yourself with and design for possible technology.

For example:

- ✓ You have seen a feature before on another product
- ✓ You read about it on a tech blog
- ✗ You imagined it, and it sounds too futuristic

But DON'T let this stop you from being innovative!

Sometimes, the technology just doesn't exist yet and your designs are ahead of their time. This doesn't mean the engineers can't build it!

However, it means the time to implement the design may take significantly longer than planned and might not come out the way you envisioned.

By understanding current technology limits, you can design more thoughtfully and engineers can develop faster.

Do Not

Be aware of common taboos! Be sure to **avoid these common pitfalls**, lest you dampen the professional relationship you are building with an engineering team.

Don't tell a developer his or her work is *"just code"*. Code can be written and deleted quickly, but the thought process and underlying architecture design to conform to a visual design can take several weeks to achieve.

Understanding both the problem, the necessary code to solve the problem, and then writing and debugging the code takes a long time!

It is insulting to tell a developer they are *"just coding"*. How would you feel if someone said you're *"just designing"*?

Don't expect the product to be finished by tomorrow.

A simple design concept can require a complex algorithm to power the technology behind it.

Developing, testing, and debugging complex code can take a bit of time. Patience is key!

Further tips on building a designer-engineer relationship

A strong engineering team understands that a healthy relationship with all stakeholders, including their design partners, is important for building great products. In case your relationship is still lacking, here are a couple ways to encourage engineers to improve their empathy with you as a designer.

Designing is not easy! Explain the design process, the planning, and iteration required for each design. Just as engineers are not *"just coding,"* **highlight that you're not *"just designing"*.** The iterative process is relatable across both roles.

If you are part of a central design group in a company, **explain that you may be working on multiple projects** at the same time.

Your design work is likely a priority for your manager. But an engineering manager wants to prioritize his or her team's work. If you find yourself butting heads with engineers about project timelines and requirements, **remind the team that you may have different priorities being assigned from above**. In a perfect world, everyone would have the same requirements for a project, but in the work-place, **diplomacy is key and compromise may be necessary.**

Thanks!

I hope this guide brings you closer to your engineers!

For questions, comments, and further discussion, please reach out.

Contact

Sara Cagle
sacagle @ UW.edu
linkedin.com/in/scagle
sacagle @ Microsoft

Thank you Erik Hofer at UW for your guidance and encouragement