3.4: Design Patterns for Mobile & Desktop

Learning Goals

Compare desktop and mobile design patterns

 Estimated Read Time: 30 Minutes.

Introduction

Hello again! Great to have you back. In the previous Exercise, you learned about common navigation heuristics and patterns as well as what can be reused to fix common usability issues related to navigation. Now that you've had some good practice with creating your own navigation solution, we’re going to broaden the scope and look at the most common design patterns for mobile and desktop. This will also help you begin conceptualizing and shaping the design and look of your application.

Let’s get started!

What’s a Design Pattern?

Simply put, a **design pattern** is a reusable, well-known solution to a usability problem. As humans, we look for familiarity in our everyday lives. Cloud formations might look like dogs and bunny rabbits, star constellations take human form, and we see familiar faces in seas of people. These are all patterns we look for instinctively, or perhaps even haphazardly.

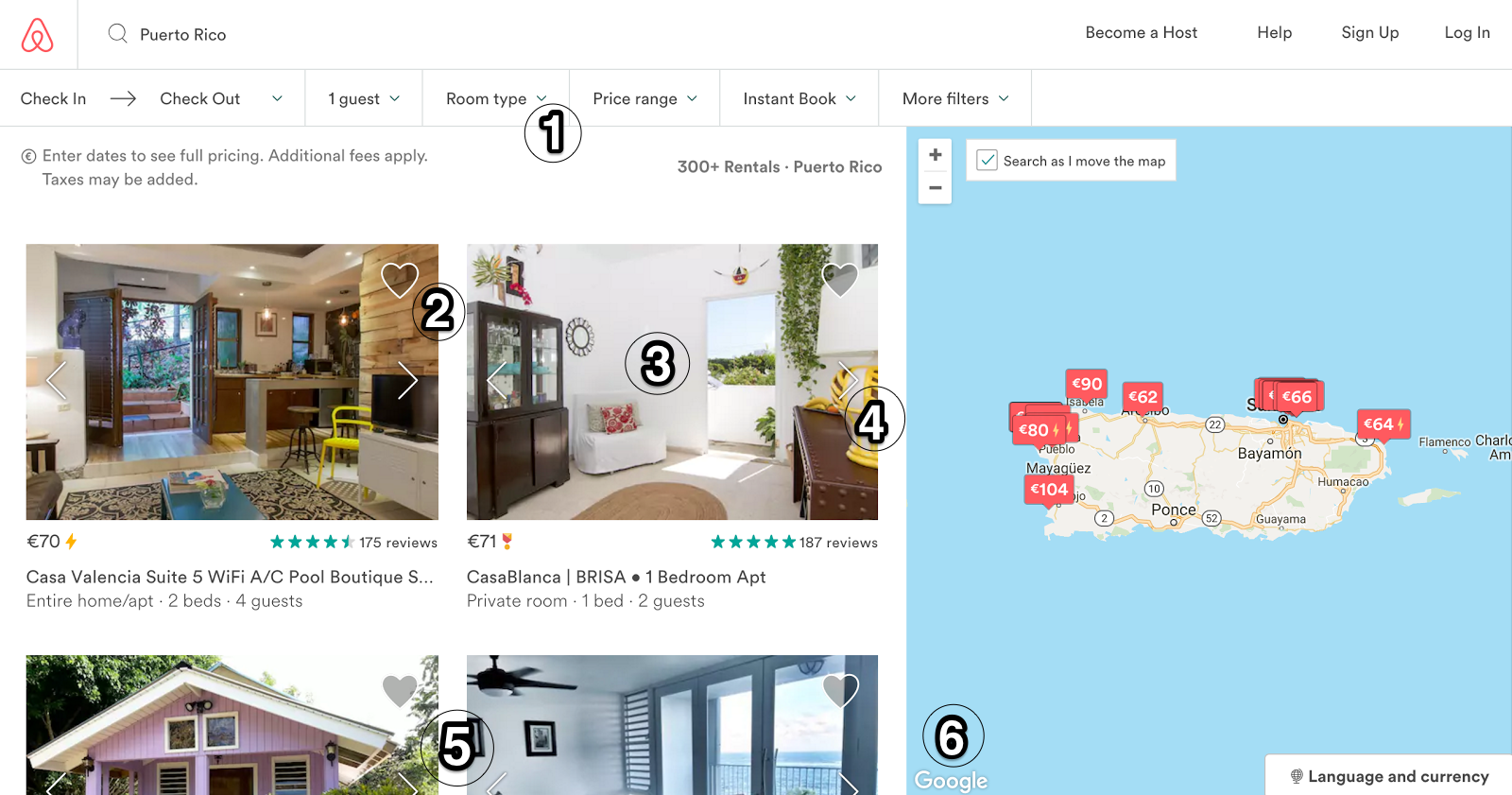
Just like in the natural world, people look for familiar patterns in the tools and software they use. This is great news for UX designers because taking advantage of this tendency can, quite frankly, make our jobs a lot easier.

People have been trained to expect certain design patterns based on the visual cues we receive from a system. Most of the time, they don’t even know this is happening! For example, when someone sees a photo on a mobile device, they just assume you can pinch to zoom in. If that doesn’t work as expected, the user may feel frustrated (or even a bit silly). When we dig into this expectation, we uncover a couple of the basic cues that led to it:

* It’s standard for mobile devices to offer multi-touch gestures.
* Nearly every photo you encounter on a mobile device can be zoomed with a pinch gesture.

Because of these two simple cues, the user expected that photo (and all photos) to adhere to the same design pattern they encountered in the past.

Now, let’s take a look at some design patterns in action. Below is a screenshot of a search for Puerto Rico homes on Airbnb. If you look closely, you'll find a number of design patterns at play that are very common and familiar to most users.



1. The down arrow next to “Room type” and other filters implies there are more options to discover either by hovering or clicking.
2. Each photo has a heart to the top right, which we can safely assume is a sort of favouriting system so that you can mark the places you’re interested in.
3. We can also safely assume that clicking any listing’s image will give us more detail about that listing as that’s how pages like this typically function on other sites.
4. The little arrow to the right of a listing implies that you can see more images without clicking on the listing itself.
5. In the left pane, we see a grid with four listings, but the bottom two are cut off. This implies that we must scroll down in order to see more.
6. The map on the right looks like a Google map and even has their watermark in the bottom left corner; therefore, we can guess that this map will have the same functionality as a Google map.

Knowing the most common design patterns and how to implement them gives us UX designers a huge leg up. Design patterns are reusable, well-known solutions to a usability problem. So, if we stick to these patterns, users will be able to understand how to use our software without reading a tutorial and without having ever used our software before. Yeah, design patterns are that powerful.

Universal Design Patterns

Before you can understand specific patterns and how they differ from mobile to desktop, you must understand the **universal design patterns** found amongst all platforms. In Exercise 3.5: Usability Heuristics & Interaction Design, we’ll discuss four principles that determine quality and usability for an application: learnability, efficiency, error management, and satisfaction. Before we do that, however, let’s zoom out for a bird’s-eye-view and talk about universal design patterns that can be used to make sure your interface is adhering to these principles.

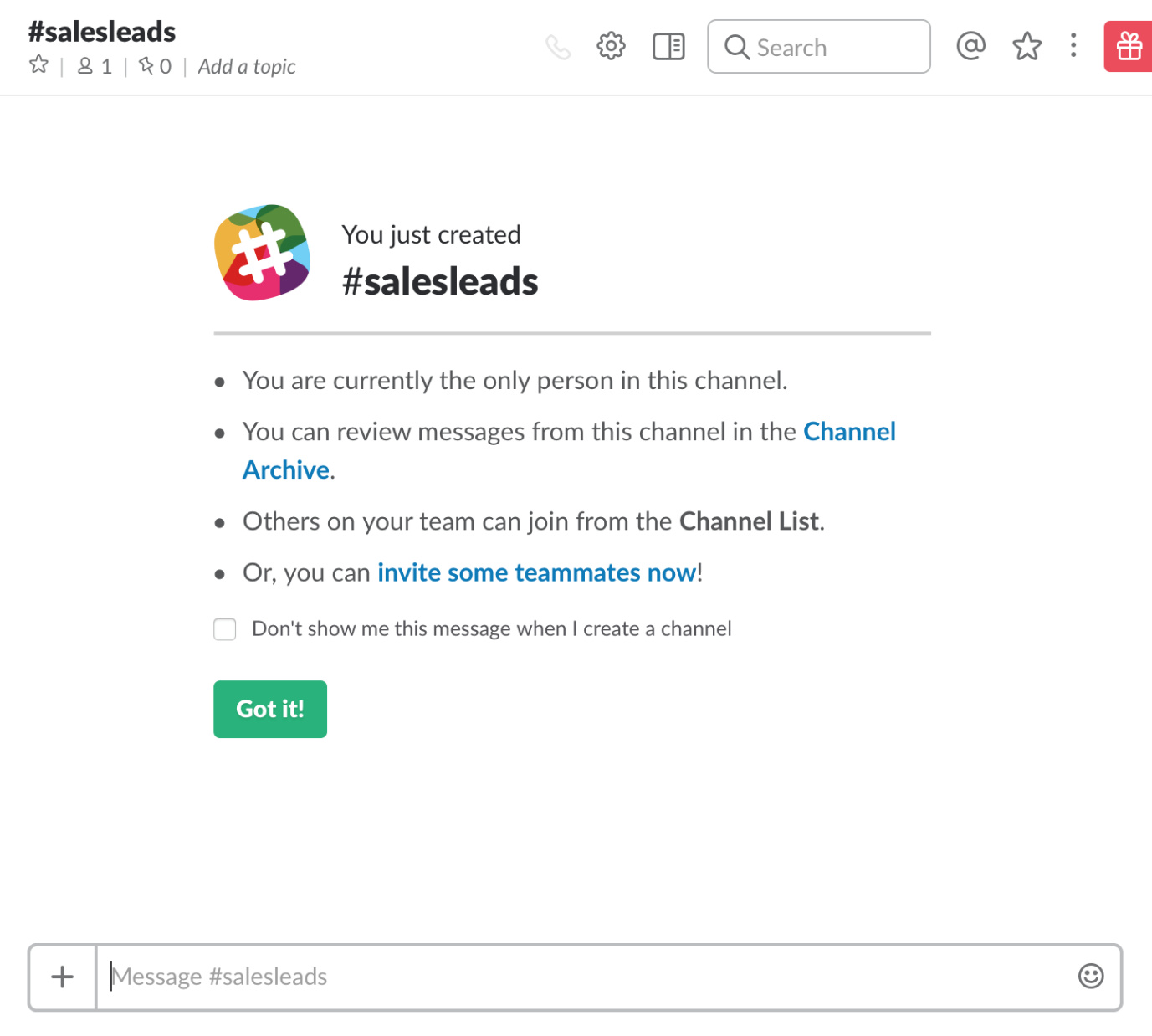
Learnability

How easy is it for a user to learn how to use your software? As mentioned previously, sticking to known design patterns can ensure that your software is as easy to learn as possible. Here are some patterns that specifically help improve **learnability**:

**Clear Empty States**

It’s very common, especially when a user is brand new, to be faced with empty screens that are yet to be filled. Things like favourites, contacts, chat messages, and notifications all typically have some sort of empty state that displays when there's no information available. Rather than just showing the user an empty screen, empty states can be used to *inform* the user.

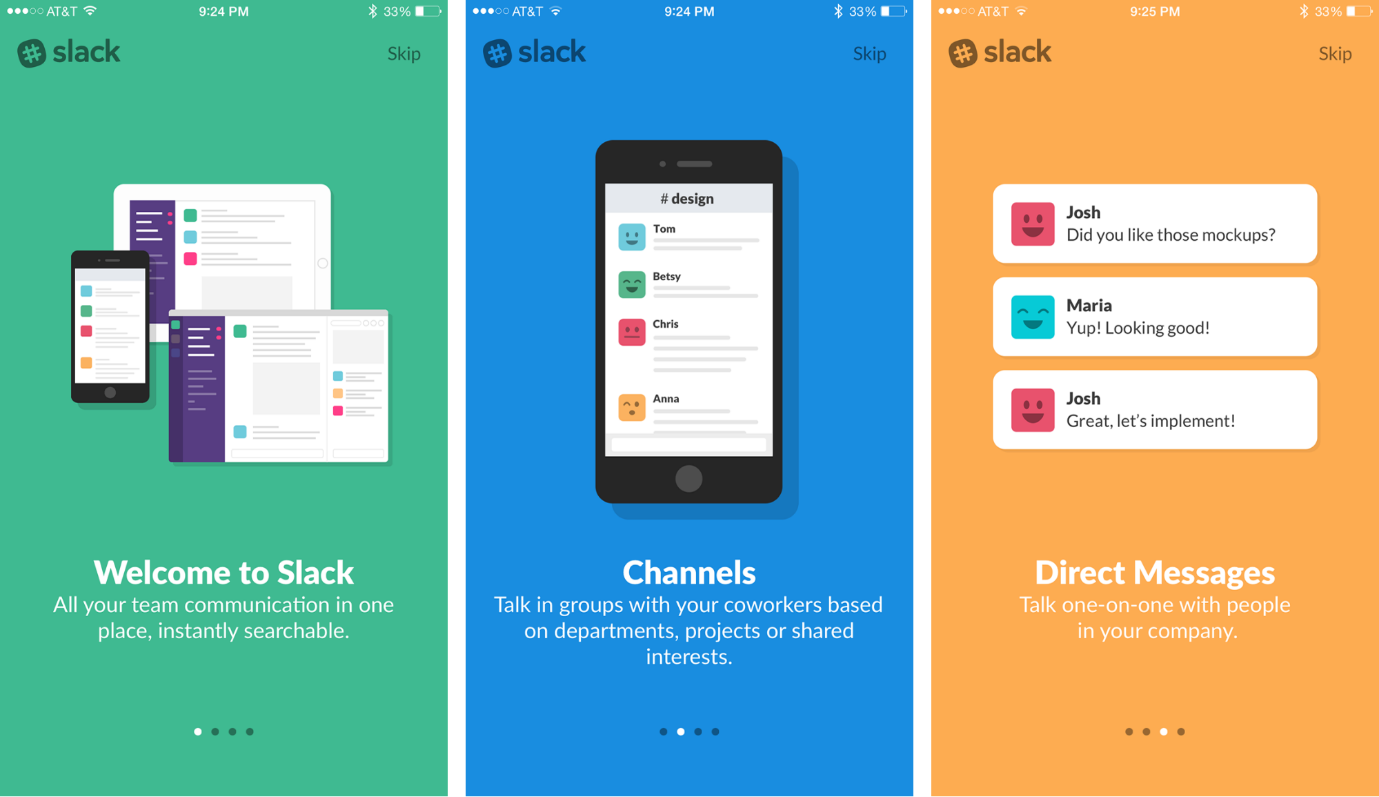
Take this empty state in Slack, for example. When you create a new channel, they present you with a list of useful information that helps you understand how these channels work.



**Onboarding**

**Onboarding** is the process of showing the user how your software works via an information walkthrough. Good onboarding uses plain language and pleasant visuals to describe features, giving you a sense of progress as you go so you know how many steps are left.

Go back to Slack for a moment and see if the onboarding in their app meets all the above criteria.



**Progressive Disclosure**

Your user shouldn’t be presented with any more information than what's necessary to accomplish the task at hand. This also plays into the concept of cognitive load, which we touched on back in Exercise 3.1: IA Principles & Frameworks. As you’ll remember, cognitive load refers to how much information a user can handle at a single time. Showing only pertinent information ensures the user isn’t overloaded.

The example from Slack’s iOS app above is a great example of this. Instead of showing all onboarding steps at once, they only show one at a time. This ensures you’re able to take your time and understand each step.

Efficiency

**Use Proper Hierarchy**

Having a clear hierarchy ensures that a user isn’t going to get lost. Breadcrumbs, a type of navigation scheme, allow a user to always know where they are and where they came from:



On mobile, it’s common for pages to slide in from the right or left. If you see an arrow in the corner of these new pages, it’s obvious that clicking it will take you back to where you came from in the hierarchy.

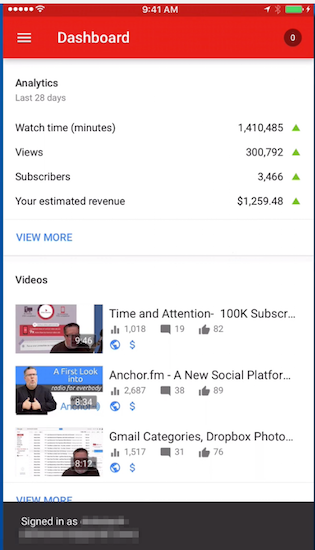


**Fewer Steps/Clicks**

While you shouldn't remove clicks just for the sake of removing clicks, it’s still important to scrutinize your user flows to determine if a particular task can be accomplished in less steps and, ultimately, fewer clicks.

**Simplicity**

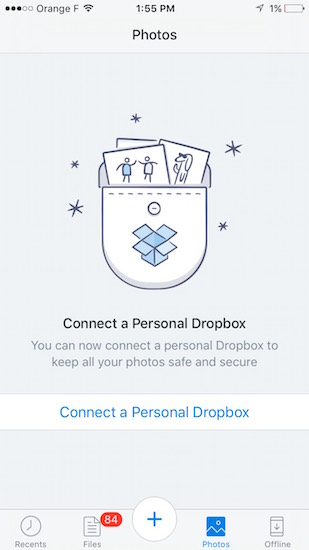
In the same vein as reducing the number of steps and clicks, it’s also important to make sure you're only showing the user information that's relevant and useful on a given screen. In the YouTube creator studio app, for example, they could stick every analytic they have on the dashboard, but instead, they only show you the most important ones and give you the option to view the rest if you’d like.



Error Management

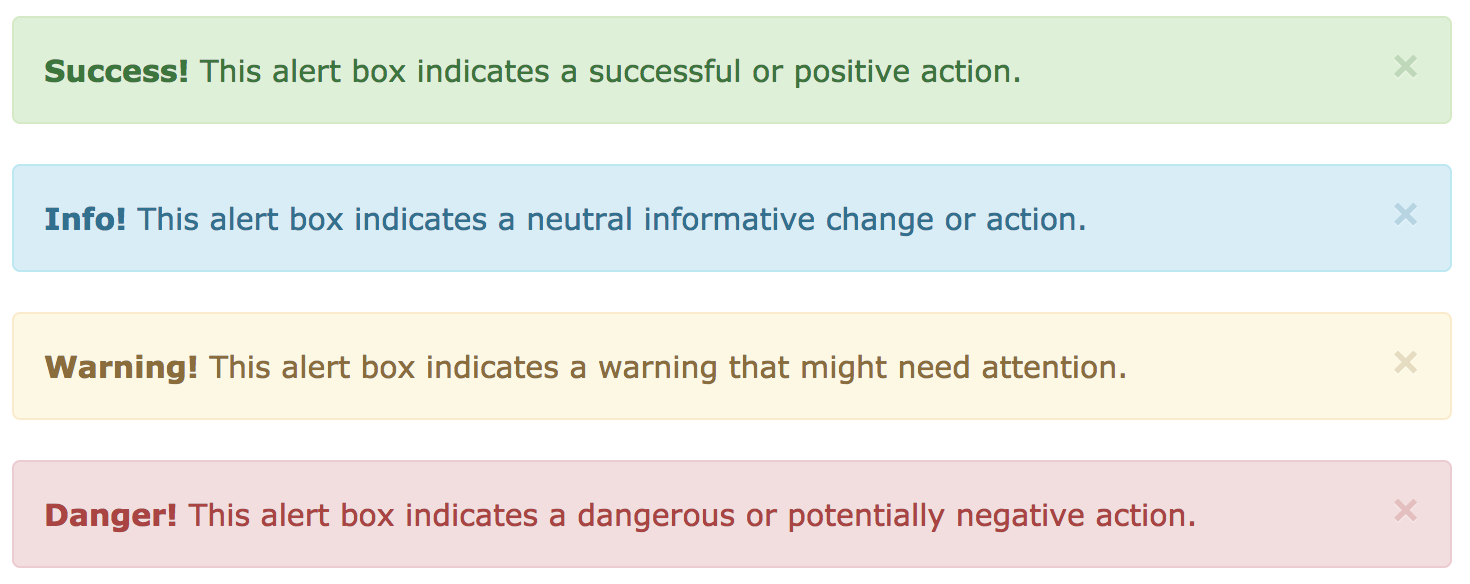
**Plain Language**

Always make sure you’re speaking the user’s language. If you use technical terms or dry language, it can be hard for the user to relate, making the software feel more robotic and less personable. Dropbox does a great job of speaking in a language that’s very easy to understand and not overly technical.



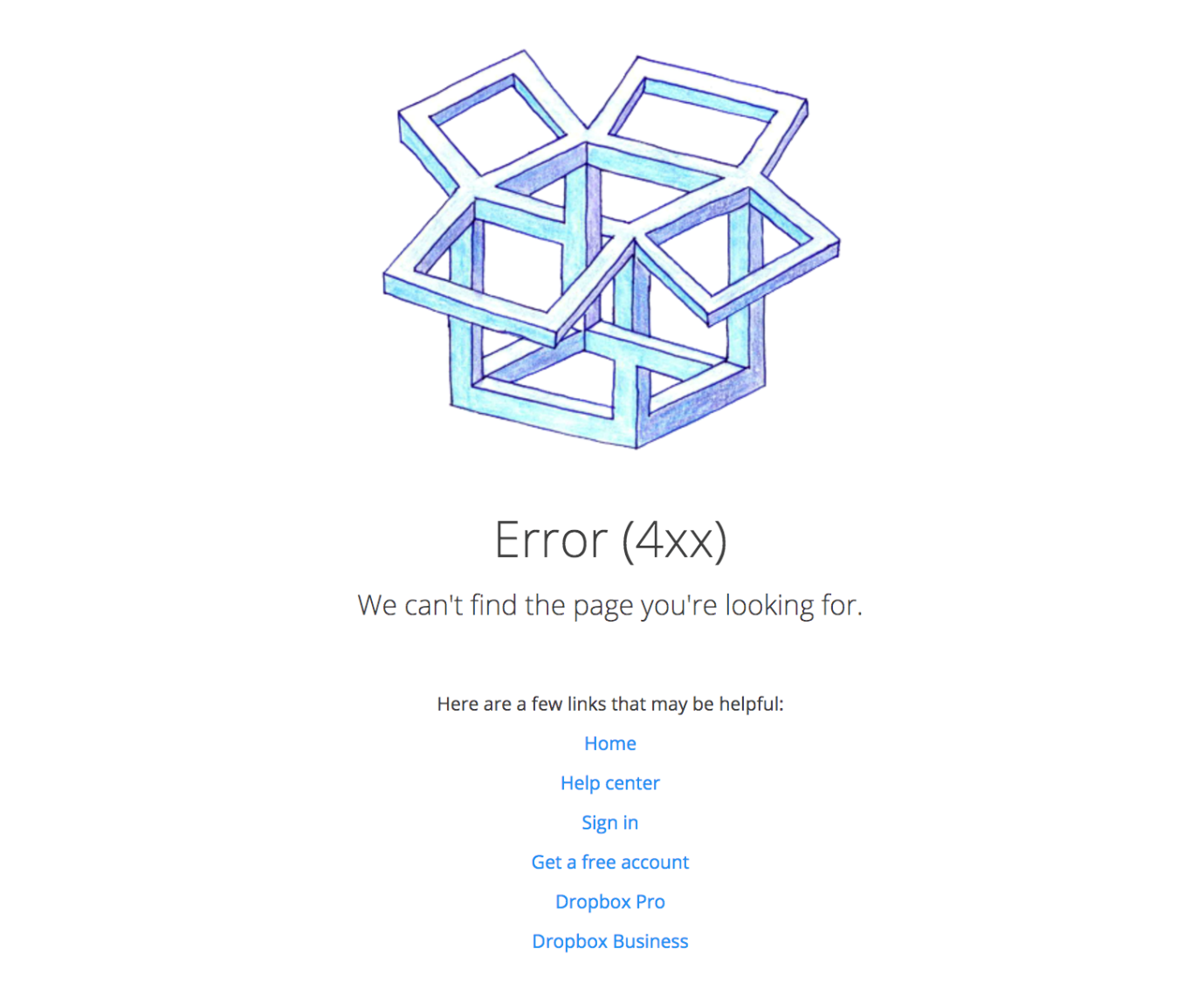
**Appropriate Colours**

Users have associations with specific colours based on their prior interaction with other interfaces. For example, red often represents errors that are critical and must be resolved, while yellow is often used for alerts or warnings that aren’t quite as critical of an error. Below, you’ll see the default alert error colours that come along with Bootstrap, which is a framework for designing web applications.

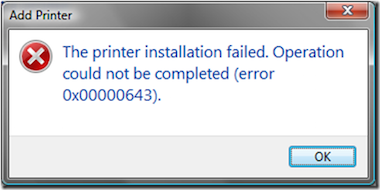


**Helpful Errors**

Errors should not only explain what went wrong, they should also offer courses of action to correct an error or move on from it.

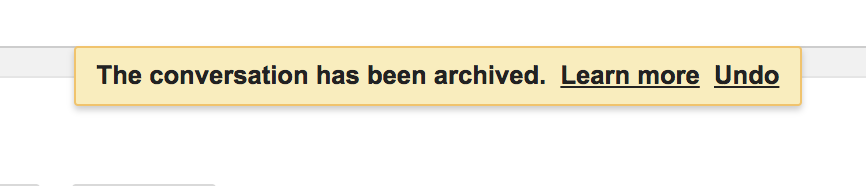


Avoid error messages like the one below. It offers no help and uses overly technical language.



**Undo vs. Confirm**

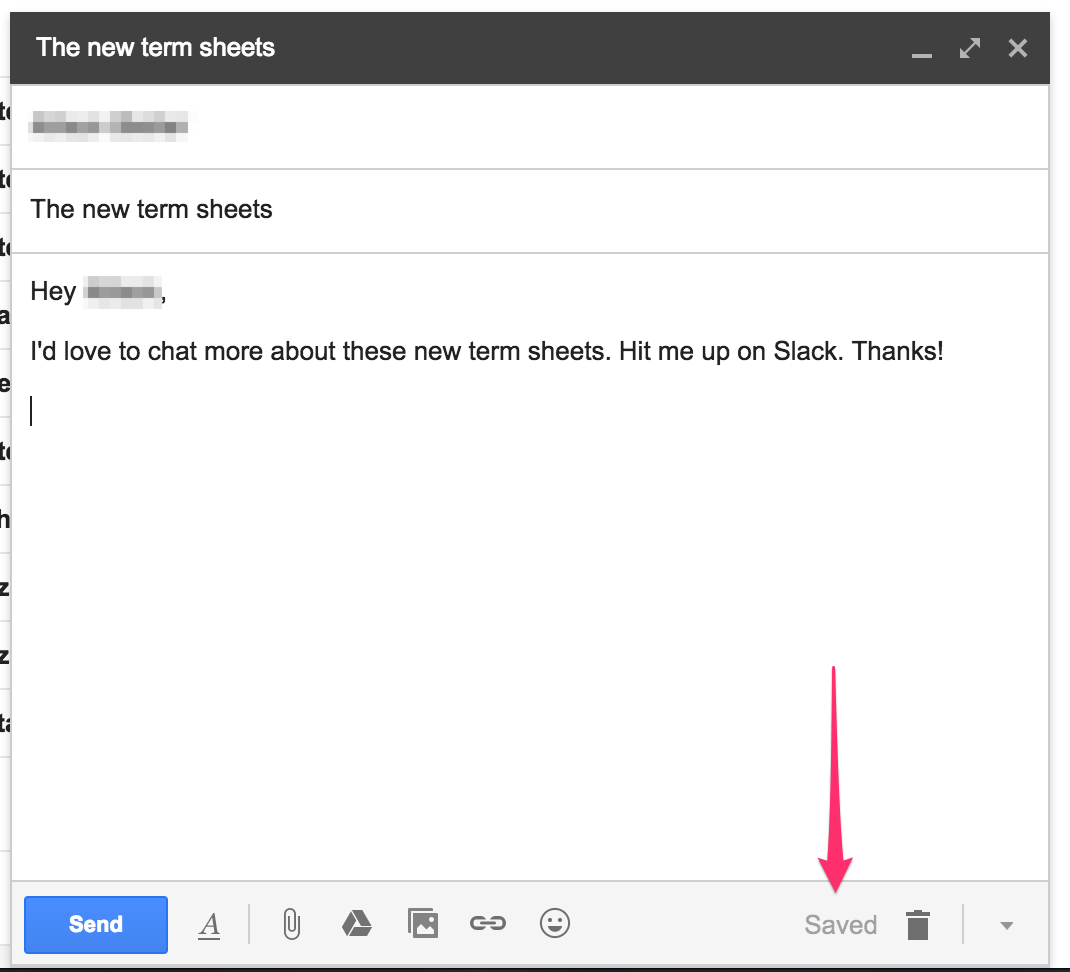
Sometimes, instead of confirming a user’s action, it’s better to let them go ahead and do the action, but give them a way out in case it was done in error. When deleting an email in Gmail, for example, you’re given the option of undoing rather than confirming a deletion. This works well because confirming each deletion could become tedious if you’re repeating this action regularly.



**Autosaving**

Another great pattern that helps account for user error is autosave. Before autosave was commonplace, forgetting to click save could easily result in lost data if your program stopped responding or your browser was accidentally closed.

Gmail also does a great job of this. When you're composing a new email, you never have to worry about accidentally closing the compose box or the browser tab or even your computer shutting down! The autosave feature ensures your email is always safe in your drafts folder.



Satisfaction

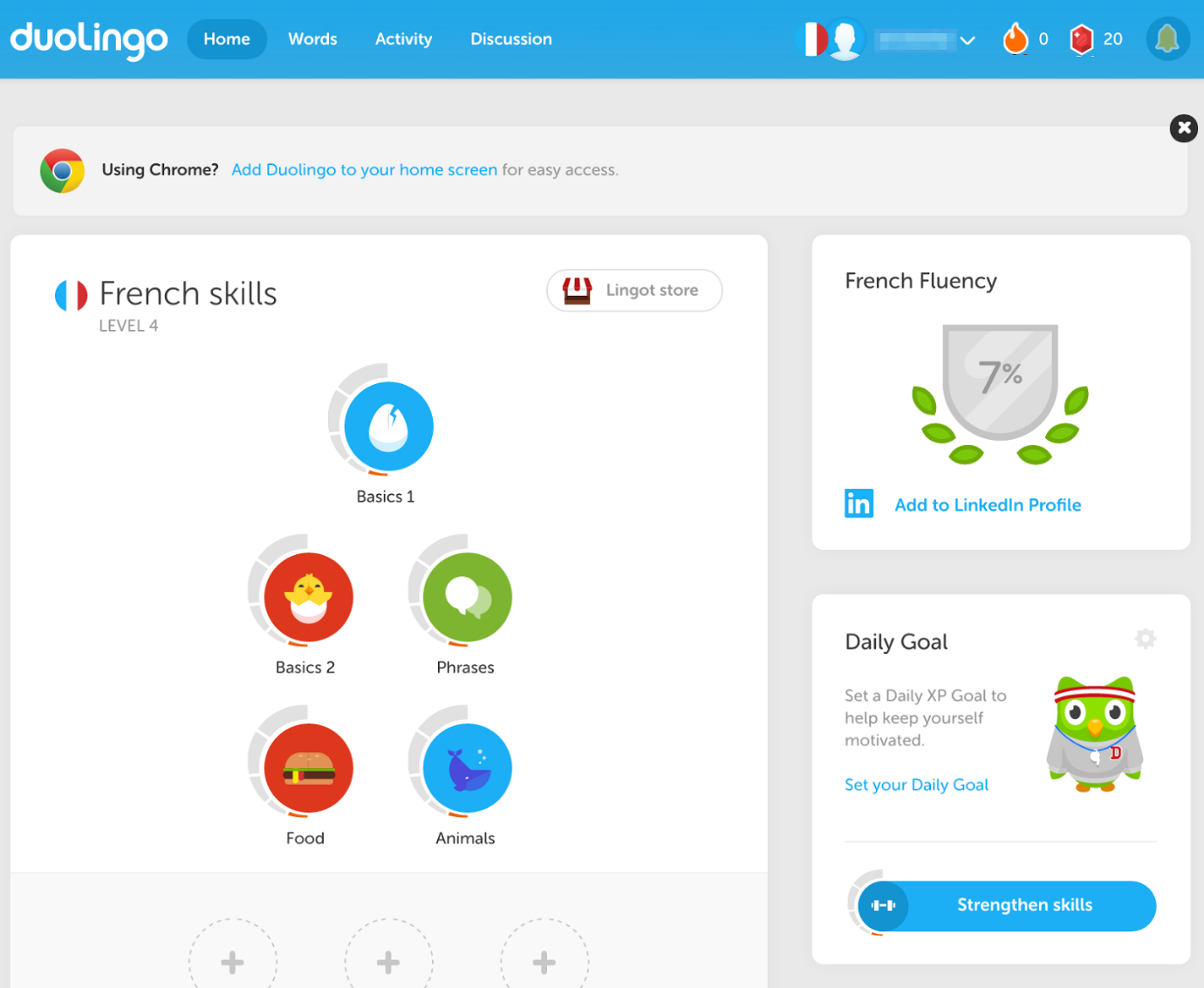
In Exercise 1.3: Human Needs & Motivations, we introduced Maslow’s Hierarchy of Needs, which states that only after a human’s basic needs are met—things like food, water, and shelter—can they worry about meeting higher level needs like creativity and emotion. This concept translates well to the world of UX design, where functionality and reliability form the base and more abstract concepts like satisfaction sit on top.

This doesn’t necessarily mean satisfaction is any less important than functionality and reliability, but that those needs must be met in addition to these higher-level needs.

**Fun and Personality**

Interfaces shouldn’t feel cold and robotic. They should be fun to use and full of personality! If a user has fun when they use an application, they associate the app with that fun feeling. This reinforces a habit loop where the user is rewarded for using the app (more on the habit loop in the Resources section at the end of this Exercise).

Duolingo is a language learning web application that does this very well. The interface is full of bright colours, fun characters, and memorable sounds that make language learning feel like a fun game. Users inevitably see the green owl character as their little sidekick, and it can be quite enjoyable!

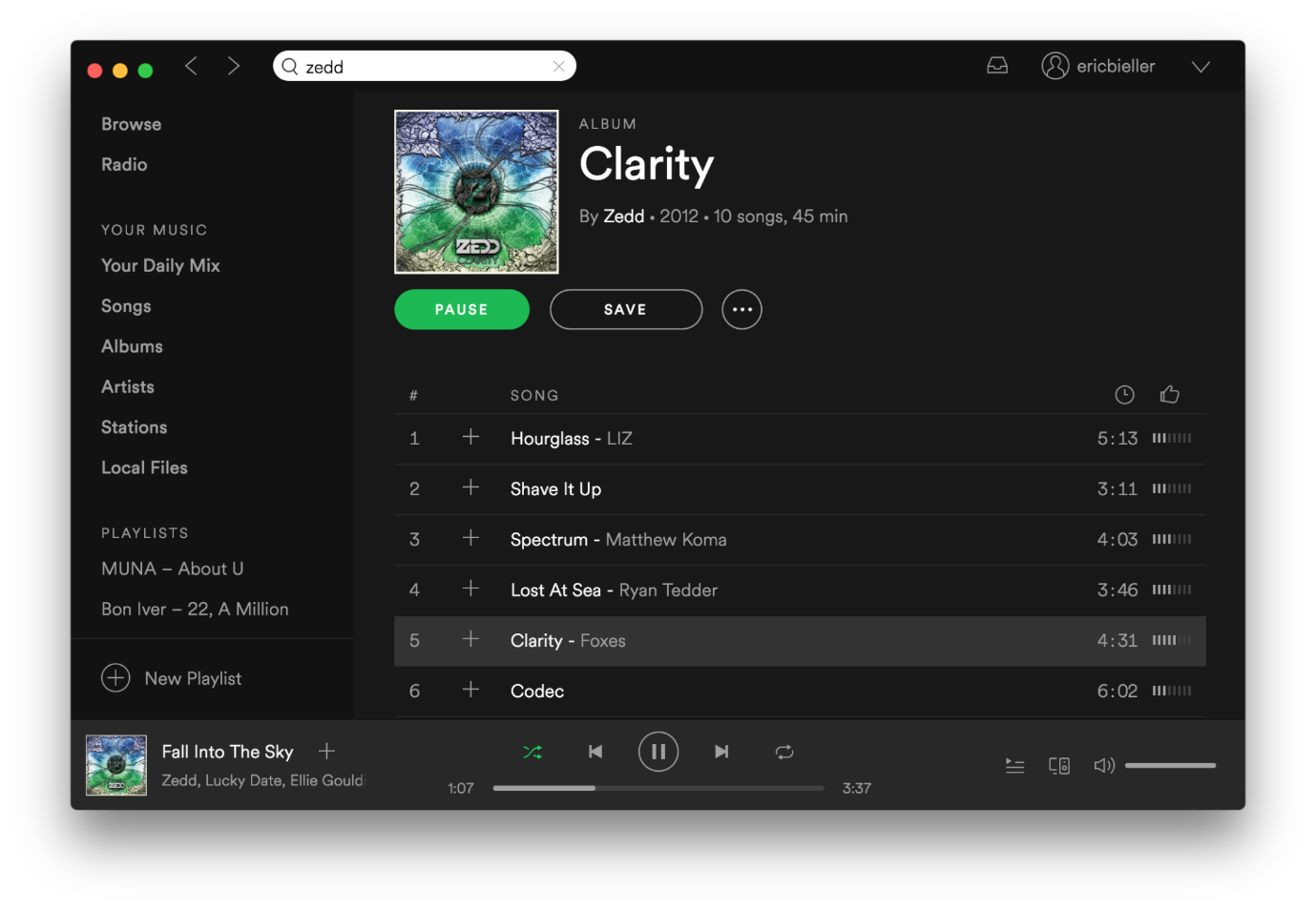


Mobile vs. Desktop Patterns

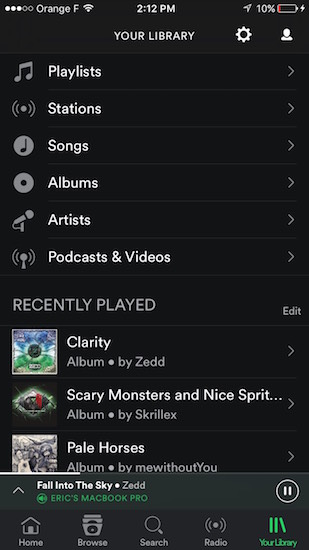
Many common patterns are consistent across desktop and mobile platforms. There are, however, subtle differences afforded by the platforms. A desktop application, for example, has a much bigger screen to work with. For this reason, the interfaces are generally more complex and offer more options on a single screen. Conversely, mobile interfaces must be simpler as there's considerably less screen real estate to work with.

Many applications such as Slack, Dropbox, or Spotify offer the same functionality in their desktop apps as they do in their mobile apps. The details of how certain tasks are accomplished, however, as well as the application’s layout, may be considerably different.

Let’s take a quick look at Spotify. You’ll notice right off the bat that the desktop app has a lot more room and, therefore, a lot more options on a single screen. They even have a left-hand navigation sidebar that stays with you throughout the app, giving you access to important items no matter where you are.

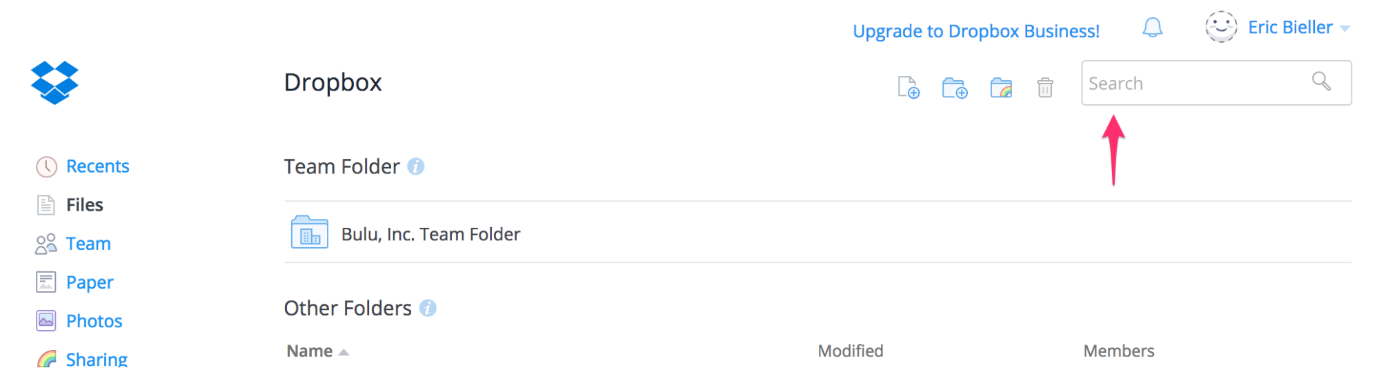


The Spotify iOS app, on the other hand, hides some less important information in order to keep your screen uncluttered. For example, you must tap a menu item to get to your playlists, while on the desktop, you can see your playlists in the left-hand sidebar at any time.

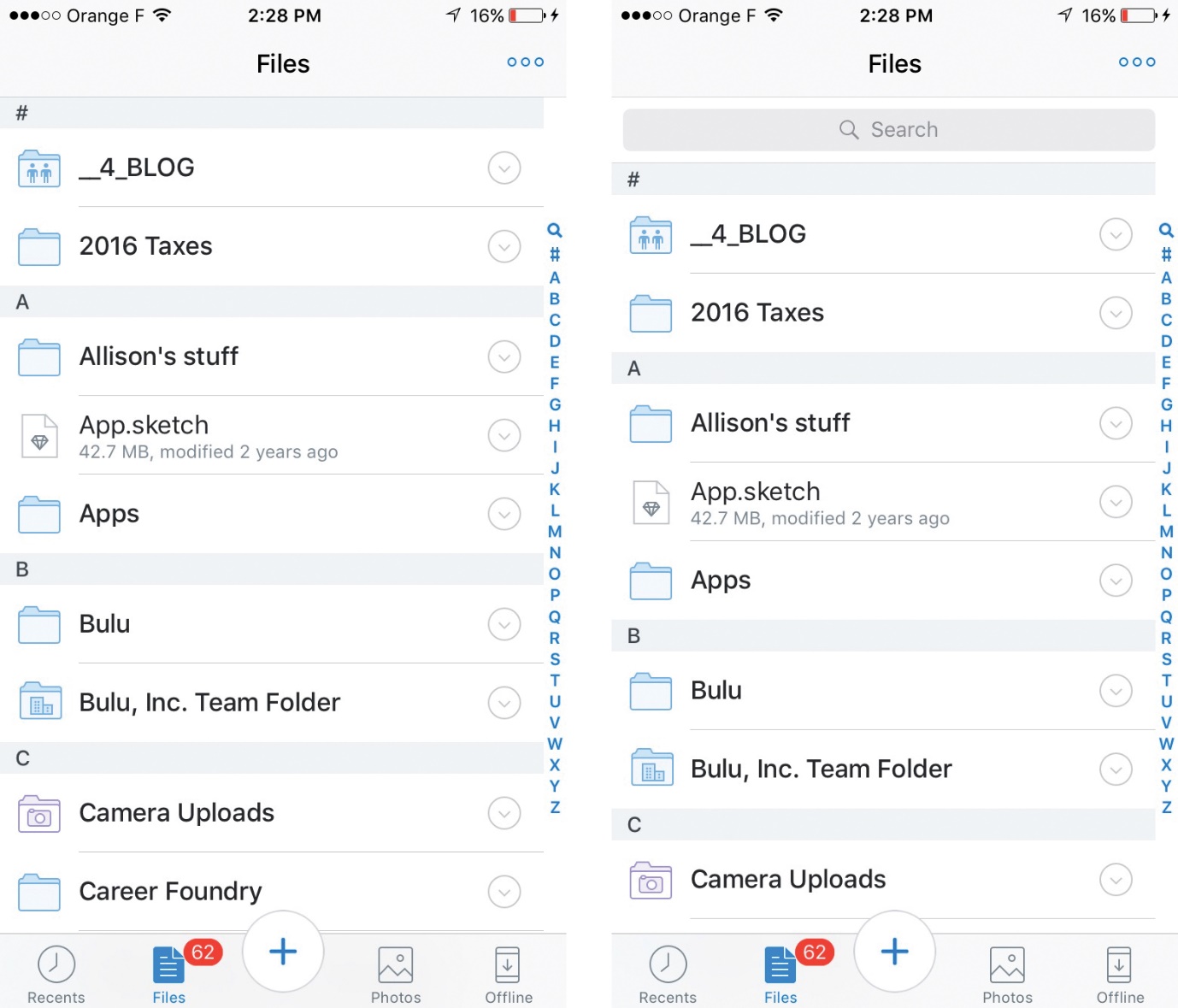


Let’s take Dropbox as another example, which offers both a web application and a mobile app. On either platform, a very common user action is to search for files. The *way* this is accomplished, however, is slightly different.

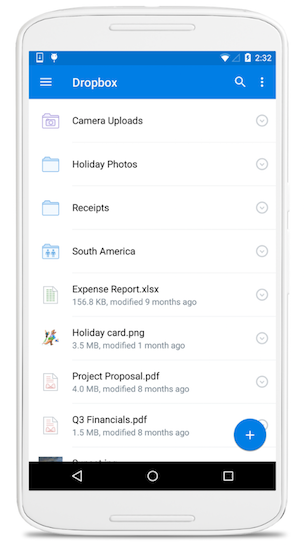
On the web, the search bar sits permanently at the top right. This is a very common spot for search bars to be found within a web application, so this makes a lot of sense and feels pretty intuitive.



On mobile, however, a common pattern is to hide the search above the content you’ll be searching. A user must reveal this search bar by scrolling down in order to use it. You might be thinking “hey, doesn’t that make the search bar really hard to find?” which would definitely be a fair argument; however, this pattern was made popular within iOS itself and slowly became a common mechanism in many applications, so the user has been trained to look for this functionality!

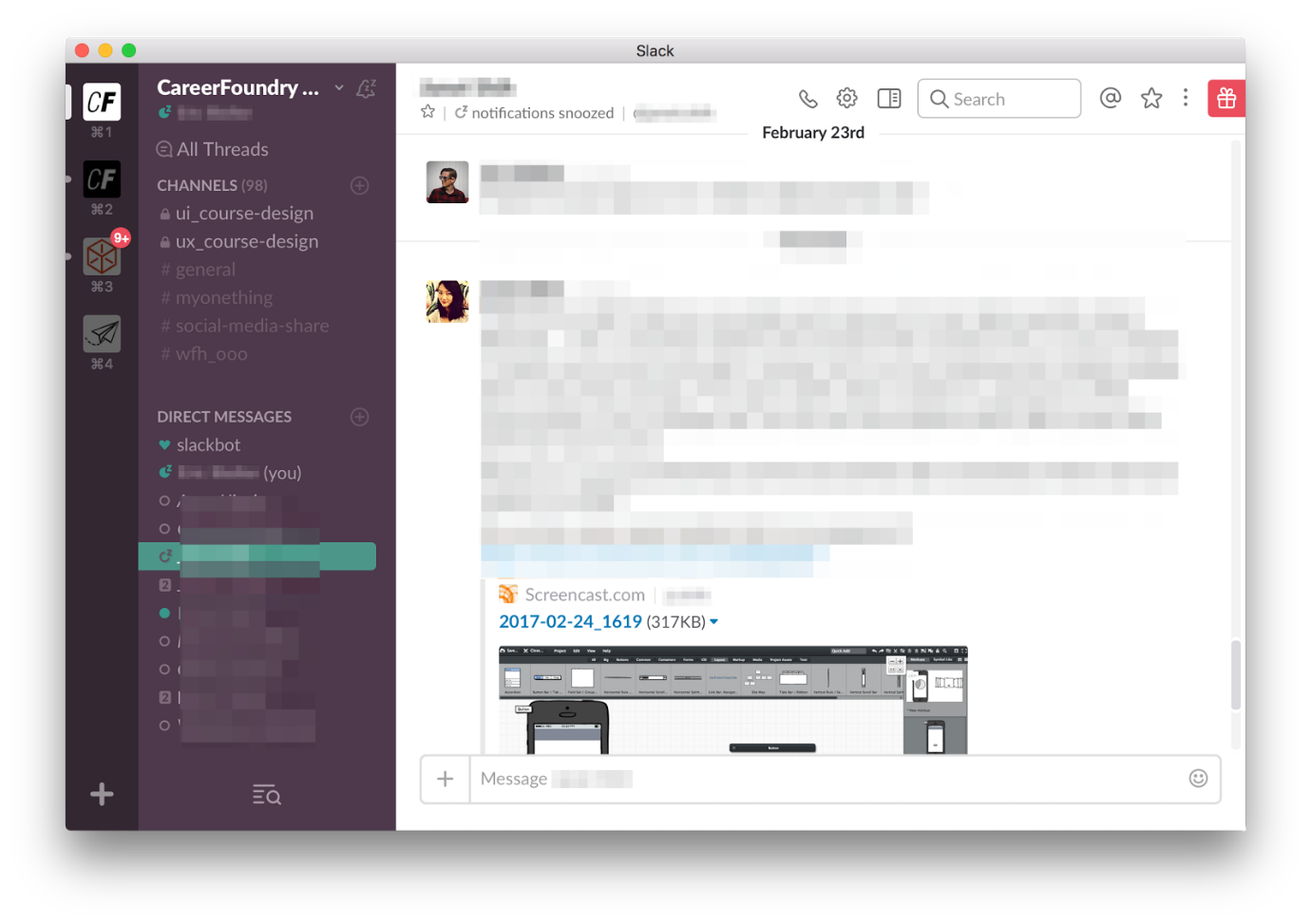


While this pattern may be popular on iOS, on Android it’s more common for search functionality to live in the title bar, so that’s what Dropbox does.

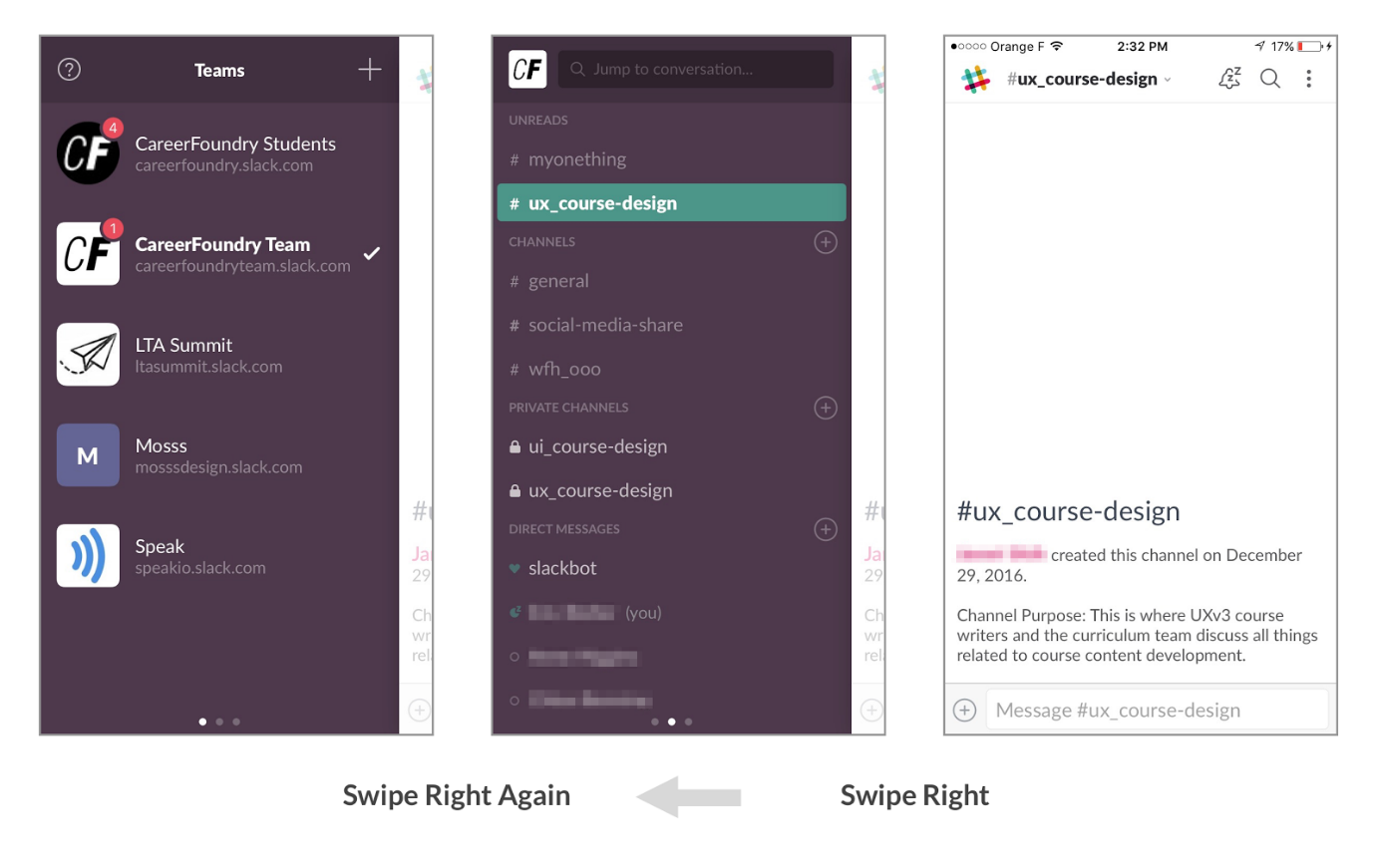


Source: [Dropbox](https://www.dropbox.com/android)

Lastly, let’s take a look at Slack. On the desktop, you’re able to see your channels and teams simultaneously on the left-hand side of the screen. This works well since there’s much more screen space. It means you can have this information on-screen without the user feeling overwhelmed. In addition, Slack has done a great job with their visual hierarchy so the information is easy to digest and very accessible.



This same functionality exists on mobile, but the pattern is tweaked to fit the platform. In Slack for iOS, you can only see your current channel. Providing more information than that would make things feel very cramped! In order to access your channel list, you can either swipe to the right or press the Slack icon at the top left.



It’s very common on iOS (and Android) for a menu icon to live at the top left-hand side of the screen, giving you access to other sections of the app. It’s also become common that users can swipe to reveal this menu. After all, it’s basically living on that left-hand side in the app’s hierarchy, so swiping to reveal it makes a lot of sense!

Taking advantage of this pattern means Slack can make their app feel far less cluttered while still ensuring usability within the app.

TRY IT!  
Fire up your favourite app on your phone and see if you notice any of these common patterns yourself. Then, open the desktop version of the app and see how these patterns may have changed.

Understanding Platform Differences

The difference between platforms goes way beyond screen size and aesthetics. A platform affords a certain level of functionality which, in turn, requires a different user experience and ultimately leads to different user behavior.

For example, research from Microsoft and the Financial Times shows that mobile usage is typically highest during mornings and evenings (6 a.m. to 9 a.m., 6 p.m. to 11 p.m.), while desktop usage peaks during the middle hours of the day (9 a.m. to 6 p.m.). This is certainly something which needs to be taken into account when designing experiences that span these two platforms.

Another difference to consider is the nature of the user’s objective. When a person hops behind a desktop or laptop, they're more willing to invest time in discovering information and digging into topics. Mobile users, on the other hand, are usually looking for a specific piece of information.

Understanding the differences between these platforms and the way they shape user behavior is critical to success as a user experience designer.

Wireframing Your Features

In the Task for this Exercise, you'll be asked to create wireframes for the three key features of your app. Do you remember what a feature is? A feature is something your app can do. It can be as short and quick as something like the "autosave" feature we discussed as above or as complex as a "comment" feature (which could include clicking buttons, writing, previewing, editing, and submitting user input among others).

Think back to Exercise 2.6 where you created your user flows (or take a look at your user flows, now!). These will give you a good idea of what features are the most important to your app. Did you have some kind of login screen? What about a search function? Or a review function? Perhaps some sort of profile page for users, locations, artists, etc.? Which of the features do you feel are most essential to the core functionality of your app? A good idea to start with would be those features shared between multiple personas.

From here, you'll be creating a single, base wireframe for each. What makes up the essence of the features you've chosen? Take a search feature, for instance. The essence of a search feature would likely be a page of search results. A review feature? Either the actual review entry page or the review list page (or, perhaps, a combination of both, depending on what you've designed). Remember: this won't be the only wireframe you design for your features! After all, most features will require more than one or two steps for a user to complete, meaning you'll need to design multiple wireframes for different screens and options that appear throughout the flow. However, this single wireframe is a good starting point, as it will allow you to mock up the most essential part of the feature flow. When you move on to creating the rest of the screens for each feature in Exercise 3.6: Low-Fidelity Prototyping, you'll be able to use this single wireframe as a jumping off point!

Summary

In this Exercise, you learned about universal design patterns which, when adhered to, can help ensure a usable product. You also learned about the difference between mobile and desktop platforms, how they affect user behavior, and how design patterns change between platforms in order to provide a better user experience.

Now that you’re armed with a solid understanding of design patterns across mobile and desktop platforms, you’ll be applying what you've learned to your own design project!

Resources

* [Mobile vs. Desktop UX Techniques For Web Designers](http://www.bittbox.com/graphic-design/mobile-vs-desktop-webdesign)
* [Android Material Guidelines](https://material.io/guidelines/)
* [iOS Human Interface Guidelines](https://developer.apple.com/ios/human-interface-guidelines/overview/design-principles/)
* [Mobile vs. Desktop: 13 Essential User Behaviors](https://www.appticles.com/blog/2016/03/mobile-vs-desktop-13-essential-user-behaviors/)
* [User Interface Design Patterns](http://ui-patterns.com/patterns)
* [Example Desktop Wireframe from the Course Demo Project: Triply](https://s3.amazonaws.com/coach-courses-us/public/courses/ux-immersion/A3/E4/A3E4_prototypedesktop.png)
* [Example Mobile Wireframe from the Course Demo Project: Triply](https://s3.amazonaws.com/coach-courses-us/public/courses/ux-immersion/A3/E4/A3E4_prototypemobile.png)

Take the quiz to test your knowledge on this Exercise.

Take Quiz

Task

* [DIRECTIONS](https://careerfoundry.com/en/course/become-a-ux-designer/exercise/design-patterns#directions)
* [SUBMISSION HISTORY](https://careerfoundry.com/en/course/become-a-ux-designer/exercise/design-patterns#step_submission_history)

 Estimated Task Time: 6 Hours.

Create both mobile and desktop wireframes for 3 of the core features of your own project. Focus on how the placement of elements change when faced with more or less screen real estate.

Feel free to reference other mobile and desktop apps to get inspiration for your own project. If you can’t find a good example, take a look at Slack, Dropbox, Spotify, Yelp, Facebook, Gmail, or Google Maps. Be sure to reference the articles in the Resources section to better understand the nuances of each platform.

By the end of the Task, you should have a total of 6 wireframes: 3 for mobile and 3 for desktop. For inspiration, take a look at [this desktop wireframe](https://s3.amazonaws.com/coach-courses-us/public/courses/ux-immersion/A3/E4/A3E4_prototypedesktop.png) and [this mobile wireframe](https://s3.amazonaws.com/coach-courses-us/public/courses/ux-immersion/A3/E4/A3E4_prototypemobile.png) for the course demo project, Triply.

**Directions**

1. Think through the most important features of your app and choose the top 3. These are the features you’ll be wireframing.
2. Decide which wireframing tool you want to use. If you haven’t already, you might want to give Balsamiq a try! It’s an industry-standard tool that came highly recommended by CF Tutors and Mentors. As a CF student, you'll have full access to Balsamiq’s software features (check the [Course Prep](https://careerfoundry.com/en/course/58/course_prep) for more details). Another tool that lets you create your wireframes quickly is Adobe XD—you can either draw your wireframes completely from scratch or use one of the many UI kits available for free. With XD, you then have the option of iterating on your wireframes later and turning them into an interactive prototype directly within the tool. Alternatively, you can also use Sketch or Photoshop or plain old pencil and paper.
3. Referencing the design patterns and heuristics from this Exercise, create your wireframes one by one. Remember, you should end up with 3 mobile wireframes and 3 desktop wireframes, and each wireframe should represent the core essence of the feature you've chosen (you'll be creating more wireframes branching off from these core screens in Exercise 3.6).
4. Combine your wireframes into one PDF and upload them here for you and your Tutor to discuss.