

Daily Work of an iOS Engineer

by Devmountain Tutorials



WHAT'S COVERED

In this lesson, you will learn how to describe the common daily activities of an iOS engineer. Specifically, this lesson will cover:

1. Complexity of Mobile Devices

Just because something works on a website doesn't mean it will work in an app. Every day I get to work through the complexities that come from mobile devices. And, honestly, this is one of the biggest reasons I love developing iOS apps, it feels like there are endless problems to solve.

Mobile devices have unique navigation patterns, gestures, animations, and transitions that I need to take into account. The size of mobile devices changes between phones and tablets that can rotate to have a portrait or landscape view, and I have to figure out how to build one view that looks good on all of them.

Mobile apps manage user sessions differently to allow them to stay logged in (unless your app has sensitive information like a banking app). This is great for the users but introduces some complexity with how we manage the data and what happens if someone else logs in.

Speaking of data, I need to be careful about how much data the app sends and receives. The more data coming in and out of the app, the more networking and data usage it eats up while draining the device's battery.

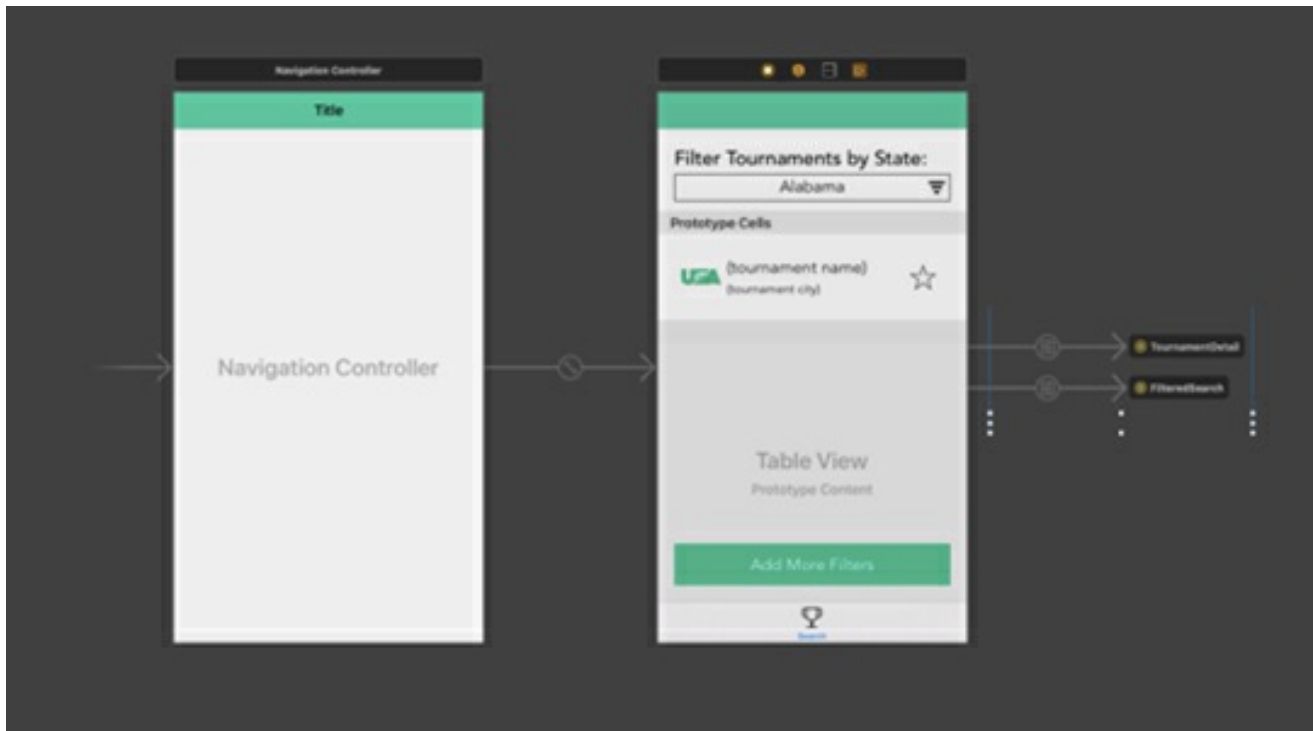
Privacy and permissions are very structured in iOS. I need to ask the user for permission to send notifications, access their camera, or save a contact to their device. If I send too many notifications, users will turn them off. I need to be mindful of the size of my app, if it takes up too much space on someone's phone, chances are they'll delete it before giving up their video from last weekend.

The day-to-day tasks an iOS Engineer works on are determined by the product's next release. Amanda (QA) mentioned our daily check-ins and the regular planning meetings we have as a team. In our planning meetings, we break down large features into the smaller development tasks needed. We review the bugs or issues that need to be addressed. We discuss the problem and then determine the solution. Jose (Product Manager) enters these details into requirements, but it is up to the engineers to determine how to build the solution in code.

2. iOS Feature Development Process

Before I start developing a new feature, I need to understand what devices my users have, what device features they have, and what version of the operating system it is using. This allows me to create backward compatibility, which is what allows users with older devices or operating systems to download the new version of our app.

With that in mind, I start creating the user flow by following the designs from Mori (UX). I create the screens, buttons, actions, and alerts that users see—we call this the interface. After creating the interface, I need to test it to make sure it works and flows as expected.



Iteration of Mobile Screens

Even the most beautifully executed screen isn't useful until it is integrated and uses real data. I connect the information on the screen to a database within the app.

You can think of a **database** as a powerful spreadsheet that has tables and rows of information. Storing data within an app is known as caching the data. I'll tell you more about this in the next section, but for now, it is helpful to know that apps have their own copy of the data that they need to manage and keep in sync. This happens through integrations, and I write the code that ties it all together.

When I'm not working on new features, I spend a good portion of my time fixing bugs, improving an existing feature, or solving crashes. An app crash is when the app dies and the user is sent to the device's home screen. This creates a bad experience, so developers try to avoid them and prevent them from happening. Occasionally, a developer may choose to force a crash when a user is doing something that shouldn't be possible; it's almost like an intentional dead end. Developers can get logs of the crashes that happen in the app; this helps us understand where the user was and what happened.



DID YOU KNOW

When something goes wrong in software, we can still provide a good experience. It is called failing gracefully. Engineers have to write code to handle these exceptions, that's why you see error messages like "Oops, something unexpected happened. Do you want to try again?"

When all the features are in place and the bugs are resolved, I compile my code to make a build of the app. I use ad hoc distribution to share the build to the registered devices our team uses for testing. This allows Amanda (QA) to run her tests, Mori (UX) to verify the design, and Jose (Product Manager) to see everything working.

When the team is happy with the release, I upload the build to the Apple App Store with release notes and images and submit them for approval. Apple has a review process in place to make sure apps don't contain viruses but also to ensure they follow the guidelines.

If you don't meet all of the guidelines, Apple will reject your app and it can't be released until you resolve the issues. It usually takes a day or two for the review process. Once the release is approved, we can push the release live to users.

Every release requires the users to manually download the update if they don't have auto-update turned on. As a developer, this means I need to be mindful of previous versions of my app that users still have installed and how the app manages data between updates.



HINT

There are a few different ways to distribute iOS apps. Most iOS apps are downloaded from the Apple App Store. But companies can distribute an app internally to the devices they manage with in-house enterprise deployment, or they can release an app to other businesses with Apple's Volume Purchase Program.



TERM TO KNOW

Database

An electronic collection of organized information.



SUMMARY

This lesson discussed the **complexity of mobile devices** and the **iOS feature development process**. Mobile devices have unique navigation patterns, gestures, animations, and transitions, and managing user sessions, data usage, privacy, and permissions in iOS introduces additional complexity. Before developing a new feature, iOS engineers need to understand the devices, features, and operating systems their users have. They then create the user flow by following the UX designs, testing the interface, and connecting the information on the screen to a database within the app. Bugs, crashes, and compatibility issues are also addressed. Finally, the app is compiled and distributed to the Apple App Store for approval, with developers needing to be mindful of previous app versions and data management between updates.

Source: This tutorial was authored by DEV MOUNTAIN and Sophia Learning. Please see our [Terms of Use](#).



TERMS TO KNOW

Database

An electronic collection of organized information.

