Role of an iOS Engineer

by Devmountain Tutorials

WHAT'S COVERED

This section will explore the role of an iOS Engineer by discussing:

1. PACE OF CHANGE IS FAST
2. XCODE AND SWIFT
3. COMPLEXITY OF MOBILE DEVICES
4. iOS FEATURE DEVELOPMENT PROCESS

iOS Engineer

Hi, I'm Camilla. I'm the iOS Engineer on the team. I've been here for 11 months. Before Poodle Jumper, I worked at another small tech company in San Francisco. I've been coding iOS apps for a few years now and I have several of my own apps in the iOS App Store. I got interested in coding in college, but I was disappointed they didn't offer more classes for modern programming languages. I was formally trained in the programming language C++. I dabbled in coding between jobs, but I didn't find my groove until a friend showed me an iOS app they made. I didn't know that the programming languages Apple uses were so similar to what I learned. I got involved in Apple's developer community and built my first app in my spare time. It was pretty simple and not a very impressive app, but once it was published, I was hooked. I love how Apple provides an entire ecosystem of tools for developers to use. When I'm not working, I love to tinker with 3D printing. I'm designing my own figures for a tabletop game I'm creating.

1. PACE OF CHANGE IS FAST

Being an iOS developer is so rewarding! The daily challenges we face evolve because the pace of change in mobile technology is on fire. iOS engineers leverage the same engineering concepts as other programming languages, but there are some differences that I'll explain.

If you want to be an iOS engineer, you need to be willing to continually learn. The primary language we write in, Swift, changes every year. So, there are always new frameworks and new syntax to learn. As Swift changes and evolves so do the size and features of iOS devices. With every new screen size, resolution, or change to the device features, we need to learn how to adjust our views and code to match the new iOS feel.

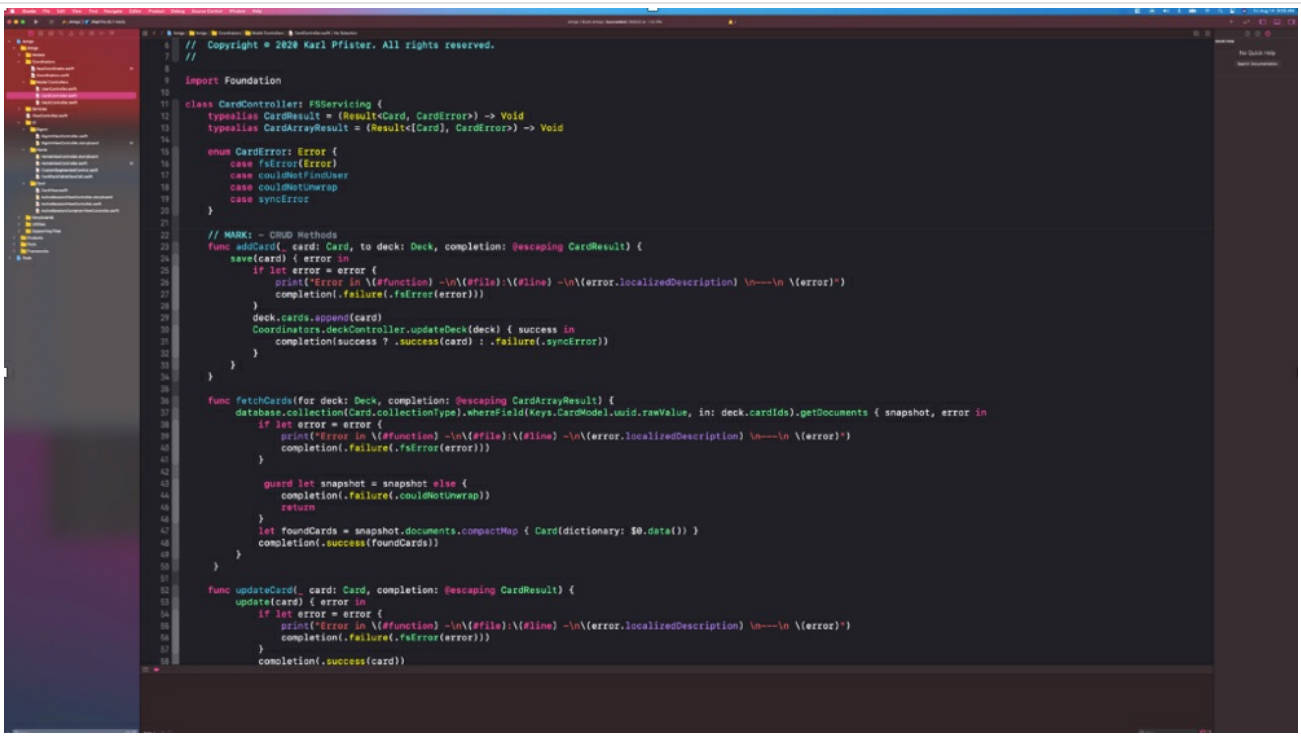
When you build an iOS application or an app, as we like to call them, you get to do the work that may be handled by many individuals in a traditional software application.

iOS engineering includes:

- Building layouts and views like a web developer
- Creating interactions like a UX Designer
- Coding automation like a QA Engineer
- Creating and managing databases like a Software Engineer or Database Administrator
- Building network requests to send and receive data like a Back-end Developer or Network Engineer

2. XCODE AND SWIFT

I primarily code in an IDE (Integrated Development Environment) named Xcode, using the programming language Swift, both of which were created by Apple, to work seamlessly together. The Xcode IDE includes a code editor, Interface Builder, a debugger, documentation, version control, tools to publish your app in the App Store, and even a device emulator. XCode contains everything you need to build iOS and Mac apps.



Swift Code in Xcode Application

When an app is developed leveraging the tools, programming language, and patterns for one platform, it is known as a native app. The app I've created for Poodle Jumper in Xcode using the Swift is specifically designed for iPhones. It can't be released on the Google Play Store for Android devices to download.

There are advantages and disadvantages to every mobile development method. It can be more expensive to develop native apps, especially if you need to provide an iOS and Android version. We decided the performance benefits, accessibility, and integrations into Apple's ecosystem made a native app the right choice for us.

Swift provides a full programming language that can build programs for the Mac, Applications for iPhone, and iPad, Apple Watch, Apple TV. Apple provides you with standard components to use, but if you want to make custom components it is significantly more work.

Knowing the constraints for how Apple wants you to make apps is an important part of creating efficient code. This can make some UX Designers crazy when they want to step outside the box. Luckily, Mori (UX) is a big advocate for accessibility, and he knows that following the **Human Interface Guidelines** provided by Apple gives the app amazing accessibility with very little effort.

DID YOU KNOW

The "OS" part of iOS stands for operating system. An operating system is software that supports the basic functions of computers that we might take for granted. A login screen, the home page, navigation, even the clock. But have you ever wondered what the "i" in iOS, iPod, iPhone stands for? Apple first introduced the naming pattern with the iMac in 1998. At the time he explained the letter i was for internet, individual, instruct, inform, and inspire.

3. 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 battery.

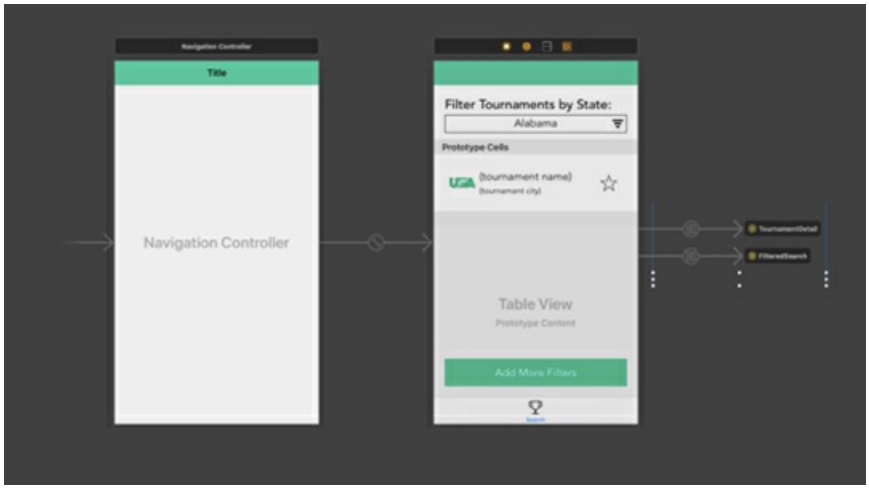
Privacy and permissions are very structured in iOS. I need to ask the user for permission to send notifications, access their camera, or save a contact to their device. If I send too many notifications, users will turn them off. I need to be mindful of the size of my app, if it takes up too much space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

4. iOS FEATURE DEVELOPMENT PROCESS

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.

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 using real data. I connect the information on the screen to a database within the app.

You can think of a **database** like 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 home screen. This creates a bad experience and developers try to avoid them and prevent them from happening.

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.

TERM TO KNOW

Database

An electronic collection of organized information.

TERMS TO KNOW

Database

An electronic collection of organized information.