From Concept to Product

Define Your App

An *app definition statement* is a concise, concrete declaration of an app's main purpose and its intended audience.

Create an app definition statement early in your development effort to help you turn an idea and a list of features into a coherent product that people want to own. Throughout development, use the definition statement to decide if potential features and behaviors make sense. Take the following steps to create a robust app definition statement.

1. List All the Features You Think Users Might Like

Go ahead and brainstorm here. At this point, you're trying to capture all the tasks related to your main product idea. Don't worry if your list is long; you'll narrow it down later.

Imagine that your initial idea is to develop an app that helps people shop for groceries. As you think about this activity, you come up with a list of related tasks—that is, potential features—that users might be interested in. For example:

- Creating lists
- Getting recipes
- Comparing prices
- Locating stores
- Annotating recipes
- Getting and using coupons
- Viewing cooking demos
- Exploring different cuisines
- Finding ingredient substitutions

2. Determine Who Your Users Are

Now you need to figure out what distinguishes your app's users from all other iOS users. In the context of your main idea, what's most important to them? Using the grocery-shopping example, you might ask whether your users:

- Usually cook at home or prefer ready-made meals
- Are committed coupon-users or think that coupons aren't worth the effort
- Enjoy hunting for speciality ingredients or seldom venture beyond the basics
- Follow recipes strictly or use recipes as inspiration
- Buy small amounts frequently or buy in bulk infrequently
- Want to keep several in-progress lists for different purposes or just want to remember a few things to buy on the way home
- Insist on specific brands or make do with the most convenient alternatives
- Tend to buy a similar set of items on each shopping trip or buy items listed in a recipe

After musing on these questions, imagine that you decide on three characteristics that best describe your target audience: Love to experiment with recipes, are often in a hurry, and are thrifty if it doesn't take too much effort.

3. Filter the Feature List Through the Audience Definition

If, after deciding on some audience characteristics, you end up with just a few app features, you're on the right track: Great iOS apps have a laser focus on the task they help users accomplish.

For example, consider the long list of possible features you came up with in Step 1. Even though these are all useful features, not all of them are likely to be appreciated by the audience you defined in Step 2.

When you examine your feature list in the context of your target audience, you conclude that your app should focus on three main features: Creating lists, getting and using coupons, and getting recipes.

Now you can craft your app definition statement, concretely summarizing what the app does and for whom. A good app definition statement for this grocery-shopping app might be:

"A shopping list creation tool for thrifty people who love to cook."

4. Don't Stop There

Use your app definition statement throughout the development process to determine the suitability of features, controls, and terminology. For example:

As you consider adding a new feature, ask yourself whether it's essential to the main purpose of your app and to your target audience. If it isn't, set it aside; it might form the basis of a different app. For example, you've decided that your users are interested in adventurous cooking, so emphasizing boxed cake mixes and ready-made meals would probably not be appreciated.

As you consider the look and behavior of the UI, ask yourself whether your users appreciate a simple, streamlined style or a more overtly thematic style. Be guided by what people might expect to accomplish with your app, such as the ability to accomplish a serious task, to get a quick answer, to delve into comprehensive content, or to be entertained. For example, although your grocery list app needs to be easy to understand and quick to use, your audience would probably appreciate a themed UI that displays plenty of beautiful pictures of ingredients and meals.

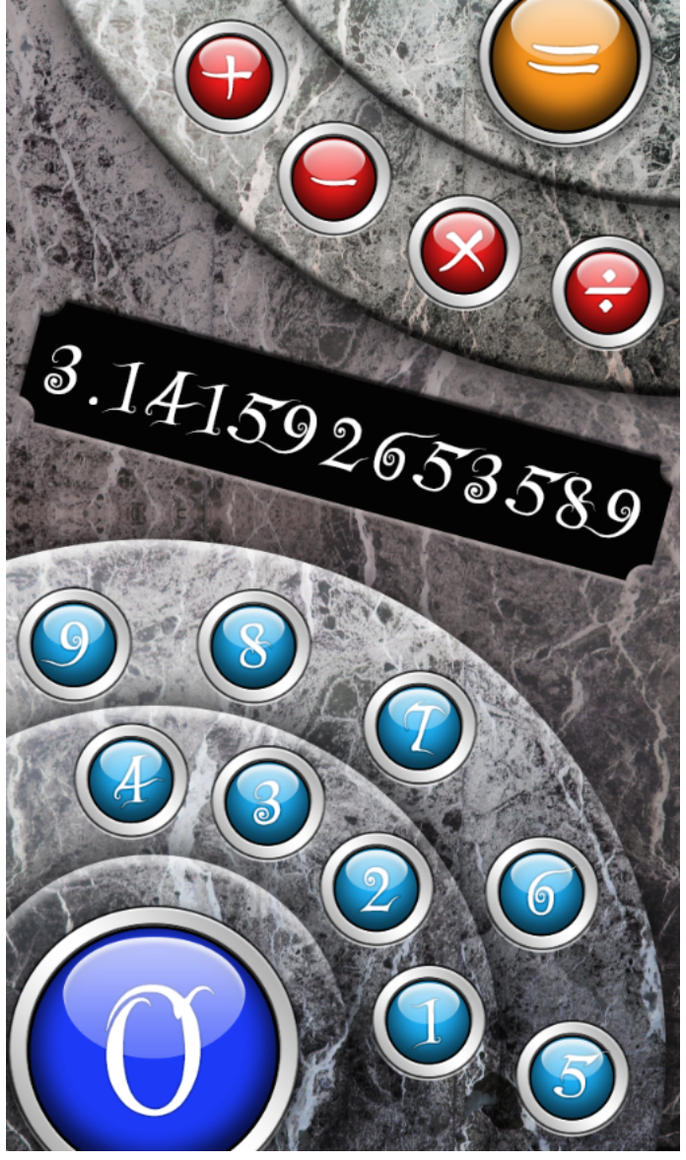
As you consider the terminology to use, strive to match your audience's expertise with the subject. For example, even though your audience might not be made up of expert chefs, you're fairly confident that they want to see the proper terms for ingredients and techniques.

Tailor Customization to the Task

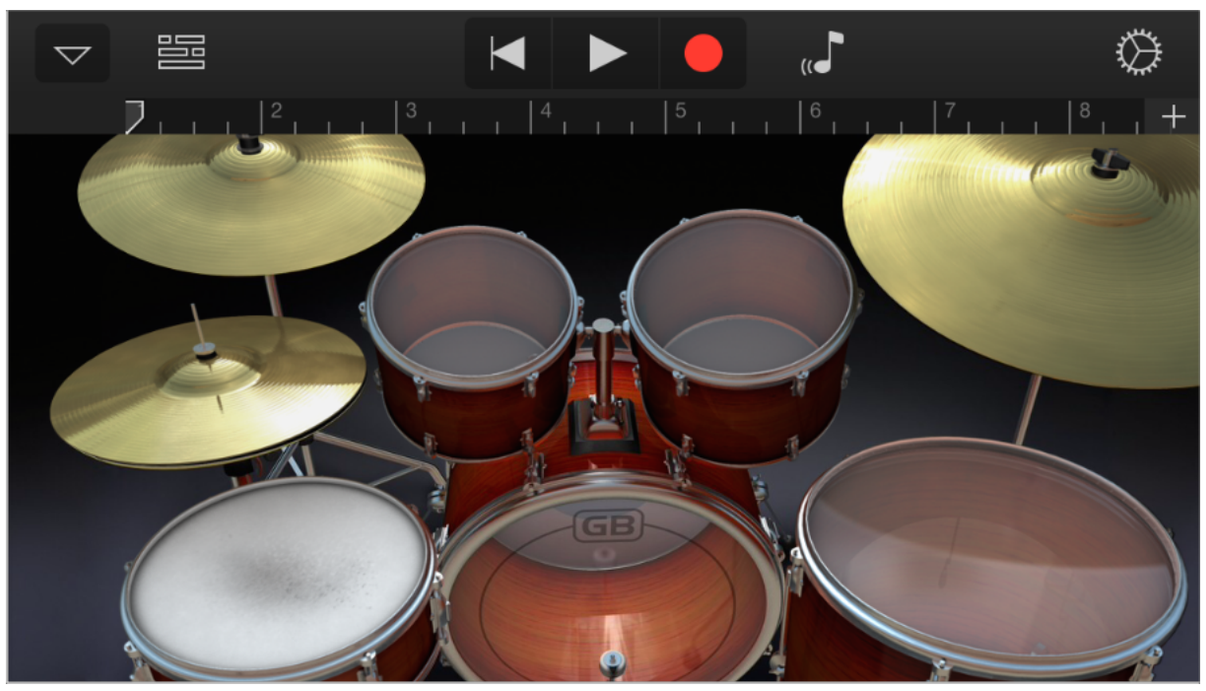
The best iOS apps balance UI customization with clarity of purpose and ease of use. To achieve this balance in your app, be sure to consider customization early in the design process. Because concerns about branding, originality, and marketability often influence customization decisions, it can be challenging to stay focused on how customization impacts the user experience.

Start by considering the tasks in your app: How often do users perform them and under what circumstances?

For example, imagine a calculator app that uses an elaborate, artistic style and imaginative layout to display familiar calculator elements. The meticulously rendered artwork and the imaginative layout don't prevent people from understanding how to tap the buttons and read the results. But for people who simply need to get their jobs done, the novelty of the experience wears off quickly and the beautiful custom UI becomes a hindrance.



In contrast, consider GarageBand. GarageBand could have helped people make music without displaying beautiful, realistic instruments, but this would have made the app less intuitive and less enjoyable to use. In GarageBand, the custom UI not only shows people how to use the app, it also makes the main task—that is, making music—easier to accomplish.



As you think about how customization might enhance or detract from the task your app enables, keep these guidelines in mind.

Always have a reason for customization. Ideally, UI customization facilitates the task people want to perform and enhances their experience. As much as possible, you need to let your app's task drive your customization decisions.

As much as possible, avoid increasing the user's cognitive burden. Users are familiar with the appearance and behavior of the standard UI elements, so they don't have to stop and think about how to use them. When faced with elements that don't look or behave at all like standard ones, users lose the advantage of their prior experience. Unless your unique elements make performing the task easier, users might dislike being forced to learn new procedures that don't transfer to any other apps.

Be internally consistent. The more custom your UI is, the more important it is for the appearance and behavior of your custom elements to be consistent within your app. If users take the time to learn how to use the unfamiliar controls you create, they expect to be able to rely on that knowledge throughout your app.

Always defer to the content. Because the standard elements are so familiar, they don't compete with the

content for people's attention. As you customize your UI, take care to ensure that it doesn't overshadow the content people care about. For example, if your app allows people to watch videos, you might choose to design custom playback controls. But whether you use custom or standard playback controls is less important than whether the controls fade out after the user begins watching the video and reappear with a tap.

Think twice before you redesign a standard control. If you plan on doing more than customizing a standard control, make sure your redesigned control provides as much information as the standard one. For example, if you create a switch control that doesn't indicate the presence of the opposite value, people might not realize that it's a two-state control.

Be sure to thoroughly user-test custom UI elements. During testing, closely observe users to see if they can predict what your elements do and if they can interact with them easily. If, for example, you create a control that has a hit target smaller than 44 x 44 points, people will have trouble activating it. Or if you create a view that responds differently to a tap than it does to a swipe, be sure the functionality the view provides is worth the extra care people have to take when interacting with it.

Prototype & Iterate

Before you invest significant engineering resources into the implementation of your design, it's a really good idea to create prototypes for user testing. Even if you can get only a few colleagues to test the prototypes, you'll benefit from their fresh perspectives on your app's functionality and user experience.

In the very early stages of your design you can use paper prototypes or wireframes to lay out the main views and controls, and to map the flow among screens. You can get some useful feedback from testing wireframes, but their sparseness may mislead testers. This is because it's difficult for people to imagine how the experience of an app will change when wireframes are filled in with real content.

You'll get more valuable feedback if you can put together a fleshed-out prototype that runs on a device. When people can interact with your prototype on a device, they're more likely to uncover places where the app doesn't function as they expect, or where the user experience is too complex.

The easiest way to create a credible prototype is to use a storyboard-based Xcode template to build a basic app, and populate it with some appropriate placeholder content. (A *storyboard* file captures the entire UI of your app, including the transitions among different screens.) Then, install the prototype on a device so that your testers can have as realistic an experience as possible.

You don't need to supply a large amount of content or enable every control in your prototype app, but you do need to provide enough context to suggest a realistic experience. Aim for a balance between the typical user experience and the more unusual edge cases. For example, if it's likely that your app will handle long lists of items, you should avoid creating a prototype that displays only one or two list items. And for testing user interactions, as long as testers can tap an area of the screen to advance to the next logical view or to perform the main task, they'll be able to provide constructive feedback.

When you base your prototype on an Xcode app template, you get lots of functionality for free and it's relatively easy to make design adjustments in response to feedback. With a short turnaround time, you should be able to test several iterations of your prototype before you solidify your design and commit resources to its implementation. To get started learning about Xcode, see *Xcode Overview*.

Copyright © 2016 Apple Inc. All rights reserved. Terms of Use | Privacy Policy | Updated: 2015-11-05