Mobile optimization ideas: Inputs and interactions

Skip Ahead

Number of inputs
Form guiders
Input methods
Default Values
How to set up a user input test in Optimizely

Note:

This article is part of a series about optimization and testing ideas for mobile apps.

User input is a great way to get information about your users and qualifying leads. Often, if you have access to the right information from your users, you can better engage your users and retain them. For many apps, the trick is to balance getting information with providing a user-friendly interface.

Information entry is a source of friction in any UX (user experience), but cramped mobile keypads often make users even more averse to inputting information while navigating the app experience.

Test styles, labels, and data validation features to make this experience less painful and identify opportunities to access additional information to make your overall app experience more engaging. This article will go over some key areas to test in your app:

- Number of inputs
- Form guiders
- Input methods
- Default values
- Auto advance
Number of inputs

Optimize the number of inputs and screens you want to include in your form. You can add fields that ask additional optional information from your users to make sure that it doesn’t negatively affect your form submission conversions. In addition, you can decide if you want to ask all your questions on one screen, or have the user navigate through multiple screens.

Metrics to track: Form submissions

Idea #1: Test collecting additional information about your users. In the case of the Fitbit app, there’s an opportunity to test both the number of inputs and the number of screens to include in the form below. The first opportunity is to test whether adding additional fields affects user engagement and the form completion conversion rate. For example, requesting additional information about the user such as their favorite sports can identify more loyal users who are willing to enter more information into the app.

Idea #2: Test the number of screens a user sees as they are completing a form. Currently, the Fitbit app uses multiple screens to ask multiple questions. Fitbit can test whether more forms are completed if all information is requested on one screen.

Form guiders

There are two key ways to help a user fill out a form: with real-time validation and ghost text.

Real-time validation can allow your users to make changes as they’re filling out their forms. This makes it such that users don’t need to fill out all the fields in a form to find out at the end that they forgot to fill out the form properly. Here, you can test if people are able to complete this form faster when real-time validation is included in your app. Redfin uses this method below when you have a question for a Redfin agent.
Metrics to track:

- Form submissions
- Number of populated fields per user

Idea #1: Test whether real-time validation of input fields increases the number of form submissions by decreasing friction. When users submit a form and need to resubmit again due to invalid inputs, it causes friction. Real-time validation can allow your users to make changes as they’re filling out their forms.

This makes it such that users don’t need to fill out all the fields in a form to find out at the end that they forgot to fill out the form properly. Here, you can test if people are able to complete this form faster when real-time validation is included in your app. Redfin uses this method below when you have a question for a Redfin agent.

Idea #2: Test whether adding ghost text helps more users fill out a form correctly on the first try. Ghost text
allows you to give users a hint as to how the information on a form needs to be filled out. Shyp uses default text to let you know what information should be filled out when you first sign up on their form.

Input methods

With a native device, users can fill out forms in interesting ways using different gestures. If you look at an app like Tinder, you might not think of swiping left or right as filling out a form, but you are indeed saying whether you are accepting or rejecting a match. On a native device, you'll find many opportunities to interact with other hardware available on the device.

Metrics to track:

- Engagement rate
- Form submission
Idea #1: Test the way in which users input their user information. The keyboard is often a cause of friction, so why are you always using free-form inputs? Mint uses a graph to allow users to input budget limits rather than just having a number input box. This method is particularly interesting, because the graph input format provides context to help users make a decision.

Default Values

Just like decreasing the amount of keyboard usage alleviates friction, you can also help your users out by helping to fill in forms with default inputs.

Metrics to track:

- Engagement rate
Idea #1: Test the number of subscribers you have to your company’s newsletter will increase if the box is checked by default. In the case of WebMD, they might assume that most people don’t want more security, so they default the PIN creation to “N”.

How to set up a user input test in Optimizely

With the Visual Editor:

In Optimizely’s Visual Editor, you can test a number of different ideas from this article, including making changes to placeholder text (ghost text), adding or modifying default text, and moving around form items (as long as your app is not built using auto layouts).
With Code Blocks:

With Code Blocks, you can add code to collect additional information about your users, add different versions of your form, and create new user input experiences. The example shows how you might implement some code that tests different versions of a flow (which differ by length).

Tip:

In this section, we provide templated code, which you can use for guidance as you’re thinking about how you would like to implement code blocks with Optimizely.

For iOS:

```swift
OptimizelyCodeBlocksKeyWithBlockNames(myCheckoutBlocksKey,
    @"shortForm",
    @"longForm");

@implementation MyViewController
- (void) someFunction {
    // This line defines Code Blocks "shortForm", "longForm", and a default block that is executed in the case that the experiment is not activated.
    [Optimizely codeBlocksWithKey:myCheckoutBlocksKey
     blockOne:^{
        // This block is executed when myCheckoutTest -> shortCheckout
        [self performSegueWithIdentifier:@"shortForm" sender:self];
    }
    blockTwo:^{
        // This block is executed when myCheckoutTest -> longCheckout
        [self performSegueWithIdentifier:@"longForm" sender:self];
    }
    defaultBlock:^{
        // This block is executed by default
        [self performSegueWithIdentifier:@"defaultForm" sender:self];
    }];
```

For Android:

```java
public class UserInputActivity extends Activity {
    private static OptimizelyCodeBlock form = Optimizely.codeBlock("Form").
            withBranchNames("shortForm", "longForm");

    private void checkout {
        // This line defines Code Blocks "shortForm", "longForm", and a default block that is executed in the case that the experiment is
// not activated.
Form.execute(new DefaultCodeBranch() {
    @Override
    public void execute() {
        // This block is executed by default
        goToDefaultForm();
    }
}, new CodeBranch() {
    @Override
    public void execute() {
        // This block is executed when myFormTest -> shortForm
        goToShortForm();
    }
}, new CodeBranch() {
    @Override
    public void execute() {
        // This block is executed when myFormTest -> longForm
        goToLongForm();
    }
});

Tip:

If you use code blocks to insert the major features you want in your new onboarding, you can use the Visual Editor from there to make copy, color, and layout changes for views that have been added via code blocks. In order to make visual changes to those code blocks, simply select the code block that you want to show in your variation in the drop down, and then switch back to the Visual Editor to make the visual changes in your code block.

Targeting

When thinking about targeting conditions for your forms, you might want to test different audiences such as if a user is new vs. returning customer. You can use custom tags (iOS | Android) to target different forms to different users. This requires code to be added to the app prior to releasing to the app store and play store.

Goals

When you test different ways of getting input from your users, you'll want to see if the number of form completions increases. A great way to do this is to use a custom event (iOS | Android), which requires code to be added to the app prior to releasing the app to the app store or play store.