Now in the last lesson, we set up firebase for Android.

Now we can tackle setting up for iOS. Now　you'll need to be running Mac OS with Xcode already installed to do this.

So if you're on Windows, you can simply skip this lesson.

You should have already created your project on the Firebase website.

So now we're ready to add our iOS app and it's all going to be under the same project in Firebase because they're going to share the same data.

So now we click on Add app and pick our iOS app to add Firebase to, then we can register our app by providing an iOS bundle ID.

Now how do we find this?

Well if we head back into our project and we click on our iOS project and open up the folder, we right click on the runner folder and open it inside Finder.

And here we're going to open up this Runner.scodeproj.

So it's the one that has a blue icon rather than the one that has a white icon.

So it's this one and it's this one right here.

So let's double click on that to open it in Xcode.

And it goes without saying this obviously means you have to be running this on a Mac.

If you're creating an iOS Flutter app that requires Firebase, you'll need access to a Mac.

And more specifically, a Mac with the latest version of Xcode installed on it.

Now as I said before, if you don't own a Mac you could probably get away with renting one or borrowing one from friends for a couple of days while you test out the app and run it and try it out on a actual Mac and on an actual iPhone device.

But as you've seen, you'll be doing most of the development and most of the code writing in Android Studio so that can be run on Mac or Windows.

So once we're here, you're going to click on this blue icon runner and it's going to open up the general settings of our app. And here lies our bundle identifier and the same thing goes for what I mentioned about the Android package.

I recommend changing this bundle identifier to change the middle part or change the first part to your own name.

So it could be com.

angelayu or if you own a web domain then it could be something like com.google.FlashCharts.

But this has to be unique in order for you to be able to publish your app onto the app store.

So once you've created your unique bundle identifier, so not co.appbrewery.FlashChat, then you can copy it and we're going to now paste it into Firebase right here. And we're going to leave the other two fields blank and we're going to register iOS counterpart of our app.

So now once it's registered, we can download a GoogleService-Info.plist and remembering what I said before also applies here.

If you downloaded it twice, then be aware that the file name changes and we have to have just this file name.

It can't look like anything else for this to work.

So once you've got that file downloaded, then it's time to incorporate it into our Xcode project.

Now notice how I'm actually adding it to Xcode directly and it's because Xcode will add this as a target of our app and it'll do all the behind the scenes linking.

Now in the Flutter tutorials online, they often tell you to add this straight into Android Studio or Visual studio wherever you're creating your Flutter app, but it actually won't work and you will get very cryptic error messages later on.

So it's really important that you take this GoogleService-Info.plist file and drop it into your runner project.

So usually I like to put it inside the runner folder right here and be sure that when this screen pops up to make sure that 'Copy items if needed' is checked.

And also this 'Add to targets' is checked.

These are really important.

So if your screen looks like this then go ahead and click finish and that should be added to your iOS project and we can now close down Xcode.

So now the next step of setting up a Firebase involves CocoaPods. And because we're using Flutter, it's actually going to do this for us automatically.

So we don't actually need to perform any of these commands.

Instead what we can do is we can go back to our Android Studio for our Flash chat project, change our destination to an iPhone, a physical device or the simulator, and hit run to build our app incorporating those things that we added just now.

All right.

So far so good.

We are getting our app launched and we're not getting any major errors in the console.

If you do see an error message in here in red, then be sure to review all the steps that I mentioned before carefully because it's really really easy to forget a step or miss a step and everything will come crashing and burning.

So if you do have any problems there be sure to re-watch what we've done in this video so far.