All right.

So now that we've cloned the starting project, it's a good time to take a look through the screens. And in order to be able to view all the screens, we're first going to set up some routes so inside our main.dart.

We're going to add a routes property to our material app and I'm going to leave this as a challenge to you.

I want you to try and remember or research or Google how we can use name routes in our Flutter app to be able to do the following.

Can you create a map of four named routes? One route should go to the welcome screen, one to the login screen, one to the registration screen and one to the chat screen. Then set the initial route to go to the welcome screen and run your app. Verify that you see the welcome screen and after that, set the initial route to go to the login screen and run your app. This time you should see the login screen show up instead.

So once you're ready pause the video and try to complete this challenge.

All right.

So if you remember, there's a great cookbook recipe on how to build named routes.

And you can either follow it or you might remember how to do this.

So first things first in our routes property,let's check to see what it expects.

So it expects a map object that has a string as a key and a route builder as an object.

So we have to supply the current context and also which screen or which stateful widget should be built as a result of going to that route.

So just as they've shown here, we're going to create our own map of routes.

So let's head back into our routes property and I'm going to build a new map by opening up a set of curly braces.

And the first item in our routes is going to have a key of welcome screen. And then we can build the route by creating an anonymous function that takes a context, the current context as an input and returns the screen that we want which is the welcome screen.

So now this is our first route out of the way.

Let's go ahead and create the second one which is going to be our login screen. And we're going to do exactly the same thing here.

But instead of going to welcome screen, we're gonna go to login screen and let's just repeat this for the other screens.

So there's registration and chats.

All right.

So we have all of our routes now defined as key value pairs and we should now be able to specify an initial route.

So the one that we're going to pick out as the start of our app is going to be the welcome screen.

Now remember if you're going to use initial routes, you can't use the home property.

They do one and the same thing.

So they would conflict with each other if they were both present.

So now we're ready to run our app and see how it looks. All going well, you should be at to see the welcome screen that has a button to login or to register.

And if you want to check out any of the other screens then just swap in the name of the route.

So if we want to take a look at the login screen, we can of course just put that in here and let's hit hot restarts and we should be able to see or login screen.

So that's all very well and good but here's the problem. Because we're only using strings to refer to other strings, if we have a typo here, our IDE Android Studio is not actually going to help us.

So let me show you what I mean.

If instead of typing login screen I didn't have enough coffee and I typed in long in screen or some other sort of typo in here, well that actually refers to absolutely nothing because there are no routes inside our map of routes that is exactly like this.

There is something called login screen but the computer is not clever enough to know that's what we're referring to.

So that means if we tried to run our app then it would crash and it would be something that seems really really obscure, something about a global key or something else.

And it basically just can't bring up the first screen.

This is why programmers are cautious and they try to avoid using strings when it matters because there's no way for the IDE for Android Studio to help us.

So when we need to use strings such as here, we try to make sure there's only one place where it exists as a string.

And in all other places, we use some sort of code to minimize errors because code is checkable whereas strings are not. Strings can be anything.

So let's change this. Why don't we associate these names of routes with the actual screen that it's going to point to?

So for example we could go into the welcome screen and right at the top here inside the welcome screen stateful widget which is the one that we want to create when we navigate over here, what if we created a new property and it's a string right?

And let's just call it id right? The id of this particular screen.

And then we can give it a string and we can call it welcome\_screen.

And now instead of providing the text here or here, we can simply refer to the welcomeScreen.id and we can do that here as well right?

So if we wanted to instead of going to login screen, we want to go to the welcome screen again we could specify that id. And so what will happen is that it will create a new welcome screen, tap into that id and check to see which of these routes have the matching key.

So let's run this. And you can see that our route now works.

We get that initial welcome screen showing up on screen. But if you think about this, it's actually quite wasteful in terms of code.

We're creating a whole new welcome screen just to be able to grab that id both here and here.

And that's not very efficient with our resources.

So what else can we do?

Well if instead of just creating a string called id that has this value, what if we added the words static in front of it?

So this is a modifier and it's used in many different languages including Java and Dart and Swift. And we can use this to modify this variable so that it's now associated with the class.

So what I can do now is instead of having to create a welcome screen object, that id now exists on the class. So I can delete the parentheses which means I'm no longer creating any new objects which makes it more efficient.

Now even if I tried to mess up really hard say if I had a in and welcome screen, well Android Studio is going to tell me about that and it's going to tell me here's an error, I don't know what this is.

You probably haven't drunk enough coffee or you know, not had breakfast.

Please review this. So I'll take a look at that and be like oh yeah that should have been welcome screen and now always well in the world. But what exactly is this static keyword that I'm using and what does it mean when I add it to a variable like this? In the next lesson we're going to do a deep dive into static methods and variables.

So if you come from a programming language where you're already super familiar with static, then feel free to skip the next lesson.

But if you want to know what this is all about and how it works in Dart, then we'll go into that in more detail in the next lesson.

