Now previously we implemented the log in and registration functionality into our app and we now have　a log out and log in.

But notice when I click on log in, everything kind of just stops and nothing's really happening.

Even though we know as the developers, there's a lot of heavy lifting in the background where we're communicating　with Firebase's servers to try and validate our user and all of that, but the user doesn't know that　it just thinks that it's not really moving.

So let's go ahead and add a loading indicator to show the user that something's happening and that it's　not stuck.

And the quickest way to do that is to simply add a modal progress HUD package. And we can grab it using　it as a dependency and add it to our pubspec.yaml file right below all our other dependencies and　let's hit up package.get to grab it. Now

once that's done, we should be able to import it into our registration screen.

So it was called modal\_progress\_hud.dart and the way that it tells us to use it is we can simply wrap　it around everything that's inside our body of our scaffold and we can use this property code insyncCall to either let it spin or make it stop and disappear.

So when this property is equal to true, then it will spin and when it's equal to false then it will be　dismissed. In order to figure out when are progress spin should spin　we have to create a boolean which I'll call showSpinner. And I'm going to start it out by making it　equal to false because it shouldn't be spinning right in the beginning right?

And then we're going to wrap everything inside the body of our scaffold with that modal progress HUD　widget. And then we're going to add that required parameter which is the insyncCall　and this is going to be set equal to the value of showSpinner.

So it starts out being false but at some point namely when the user presses on the register button, then　we're going to get it to start spinning.

So we have to set it to true.

And because everything in Flutter is reactive, we have to use a set state and inside the set state we'll　set the showSpinner to true.

So we call our build method again and we start spinning our spinner.

Now if we head back to our registration screen, then you can see that as soon as I click on the button,　it's going to start spinning and it's not really going to stop.

So we have to tell it when to stop.

Now it should probably stop once we've actually gotten back the registered user, so probably around here.

We can use another set state and inside it we can set the showSpinner now to false.

So now if we hit save and we go back to our previous screen and we click on register and let's register　a new user right? Let's call him jack@email.com and we're going to add our trustee　123456 as the password and click register.

So we see that spinner for the amount of time that it takes for authentication to go through.

And once it's done, it gets dismissed and we get the next screen pushed onto the stack.

So that's a far better user experience　wouldn't you agree? As a challenge I want you to add the spinner for the login screen so that we get the　same experience over there as well.

So pause the video and try and give that a go. All right.

So we of course start out by importing our modal progress HUD and once we've done that we create a Boolean　value that's going to hold whether if we should show spinner and we again start it off as false.

And now we're going to wrap everything in our body inside the modal progress HUD and we're going to　add that property which is code inAsyncCall and we're going to set it to equal the showSpinner property,　so it starts off being false. And again in our button　as soon as we press on it, we're going to use a set state to set the showSpinner　now too true.

And we're going to set it back to false　once we've actually gotten back a user and we've successfully logged them in. Now　that's it.

So we've got a spinner for our log in screen as well as are registration screen. And we can test it and　see it in action by logging in with our previous registered users. In the next lesson,　we're going to actually start writing code for our chat screen and we're going to be saving the data　that the user types as their message into our cloud Firestore.

So all of that is coming up and I'll see you on the next lesson.