So now that we've done all the fiddly bits of incorporating Firebase into our Flutter app, it's finally　time to actually use Firebase and tap into all of its potential. And its good point to mention that　make sure that you've actually run and tested the app before this stage because this is when things start　getting complicated and it can get really hairy if you haven't got a clean start namely everything is　actually working up to this point. But once you're ready let's go into our registration screen.

And if you've still got some of these gradle files etc open, then be sure to close them down. So we're　ready to go ahead and actually receive some of the data that the user typed into these text fields.

And the first text field which says 'Enter your email' is going to obviously contain the email　right?

And the second one is going to contain the password.

So let's store that value that the user typed in here　when that text file gets changed inside a variable.

So inside our registration screen state, let's create two new variables.

Both are going to be strings.

One is gonna be called email and　the other is going to be called password.

Now once we've created these variables, we're going to set them equal to the value that the user types　in here.

So the first one is going to be email　and the second one is going to be the password.

Now we're able to save what the users typed into here.

And the moment when they press on the register button, we should have a value for the password and a　value for the email.

So let's go ahead and test that out.

So inside this onPressed for our button, let's print out the email and also the password.

And now let's hit save to hot refresh our app and let's go in to register and try to register ourselves　as a new user.

So I'm going to call myself angela@gmail.com and I'm going to put in my password as 123456.

And I'm gonna hit register and now if we look inside our console, you can see those two values have been　captured and they're now available for use inside our onPressed.

And this is where we're going to be using those values to register our user with an email and a password.

Now while we here though, if we take a look at the current appearance of our text field, they look very　basic and there's a couple of things that we can do to make it just a little bit better.

One of the first things I'd like to do is to center align the text in here and you can do that really　easily by adding it as a property on the text field. So we can add a textAlign property and we can set　it to text align center and we can do the same for our password field.

So I'm just going to paste it into this other text field. So now if I perform a hot restart, you can see　that my place holder or my hintText is now centered.

And also when I start typing, it's also centred as well and it looks a lot better.

Now if you try to hot reload and you've got some funny behavior and some weird text in here, just be　sure to hot restart the app from scratch so that the text field can actually update.

Now once we've done that, the second thing that you'll notice is the password is kind of just out there　in the open right?

We can turn this into a standard password field by changing the second text field's property called obscure　text.

And if we set this to true, then you'll see that when I start typing inside this field then all my letters　or numbers become these little dots.

So if somebody is looking over my shoulder, they won't be to see what my super secret password is.

The final thing I want to show you is that whenever we use these text fields, we get a bog standard keyboard　showing up here.

And if you don't see that soft keyboard coming up, then on a Mac you can hit COMMAND + K and when you are　on Android and you click on one of these text fields, you should see a little keyboard icon show up here　and you should be able to toggle the keyboard on and off depending on that show virtual keyboard toggle.

Coming back to our code,I mentioned that the keyboard that you see here is the same for both fields and it makes it quite awkward　when you're typing an email if you have to go into the symbols in order to get that @ sign.

So what we can do is inside the e-mail text field which is this one, we can add one more property which　is the keyboard type. And we can specify that the keyboard type should be a text input type and the type　is going to be .emailAddress.

And when I hit save and you take a look at the difference.

Now when I activate the keyboard on the top one, you can see that it's actually the e-mail typing keyboard　with a space and the @ sign and also if you hold down the . you'll get all the common endings, the　com, net et cetera.

So this just makes it easier for our users　and these are really very minor adjustments. So now that we've added these to our text fields in the　registration screen, you'll probably realize that we'll need the same things in our log in screen.

So let's just copy these over to the log in screen.

So the first text field is going to get the keyboard type and the second text field is going to get　the obscure text and text align, so we can put those inside these two text fields.

And now let's head back to our registration screen and actually do the registering of our users.

Now in order to authenticate our users, we're going to be importing one of those packages that we installed　earlier on.

And that's the Firebase authentication package.

And in order to use authentication, let's import that into our registration screen.

So if you type auth you should be able to find the firebase\_auth.dart package.

Now the next step is to create a new authentication instance.

And we do that inside our registration screen state right here.

I'm going to create it as a final because I'm never going to change it once I've created it, and I'm　going to call it \_auth to keep it as a private property so that other classes can't accidentally　mess with this variable.

And I'm gonna set it to equal the FirebaseAuth.instance and you can see that this is a static instance　of this class.

So we're going to be using this auth object to actually authenticate the users.

We're going to be using the associated methods namely the sign in with email and password and also create　user with email and password methods.

So once we've created our auth instance, we're going to go down to the onPressed here where we have　our email and password.

We don't really need to print them out but we need to be able to use it to register our user.

Now in order to register our user, we're going to tap into that auth object that we created just now.

And we're going to use one of its associated methods which is create user with email and password. And　you can see it takes only two required field, email and a password both as strings. And it returns a future　because it can take any amount of time to authenticate and create our user.

So we don't want the user interface to be hanging while that's happening. Do let's hit enter and let's　put in our email variable as the email and password as the password.

And because this is going to return a future, let's keep a hold of that future and capture it in a final　variable.

So let's call it the newUser and we'll set it to equal the output of this method.

Now because this is an asynchronous method, we don't want to continue on without knowing whether if our　new user has been created.

So we're going to turn onPressed into an async method and we're going to add the keyword await in front　of our create user with email and password.

This way we can be sure that we definitely have finished authenticating and creating our user before　we proceed to the next step.

Because this can fail for a number of reasons, a user could have entered an email that's already been　registered, they could have entered an invalid email or an invalid password etc..

We want to be able to use our try and catch blocks to be able to cut any exceptions that might occur.

And for now, I'm simply just going to print the exception.

There is far more fancy ways of dealing with it　but this is good enough for now because we're focusing on the creating user part.

So if this async and await seems unfamiliar, then be sure to review the entire module where we focused　on talking about asynchronous methods and how we deal with it in Dart which is the Clima module.

But if all of this looks good to you then we can proceed onto the next step where we're going to check　to make sure that new user is not equal to null.

So we actually managed to get a user who is registered back and in that case we're going to navigate　the user to our chat screen.

So we're going to use pushNamed and we're going to use the route name which is our ChatScreen.id　and this is of course that static variable which comes from the chat class.

So we have to import that chat\_screen.dart.

Now that the user navigates to the chat screen, well we can pick it up from over here and we want to　be able to print out the current user's email address because we're going to need that when we start　sending messages in their name.

We're going to have to tag our messages with the sender and what better way of doing it than using the　signed in user's email?

So over here we're also going to import the auth package and we're also going to create another one　of those auth instances.

So it's going to be equal to the Firebase auth.instance and then we're going to create a method　called getCurrentUser.

And this method is not going to take any input or have any outputs but what it will do though is it　will check to see if there is a current user who is signed in. So previously if our registration was successful,then this user actually gets saved into the authentication object as a current user,so somebody who's signed in. Inside the chat screen, we can actually create a new final variable called　the user and we'll set it to equal the auth object .currentUser.

So this will be null if nobody is currently assigned in.

But if somebody has registered or if somebody is logged in, then this will correspond to the current　user and we'll be able to tap into that user's email or password.

Now because this is also an asynchronous method, it returns a future, in order to use this user,　we again have to mark this as async and make sure that we wait on this line to complete.

So now that that's complete we're going to check to see if user is not equal to null, then that means we　do have a currently signed in user.

Well in that case we're going to create a new variable,　so a new FirebaseUser and we're going to call this the loggedIn user and we're going to assign this　user that we got back from the current user to the loggedIn user　as long as it's not null.

And again because this can fail, we should wrap this inside a try and catch block so that we catch any　exceptions that might happen.

And finally we have to trigger this method somewhere.

And what a better place to trigger it than when our state is initialized? So let's delete that and let's　go ahead and call get current user. And once we call get current user, we're going to print the logged　in user.email property into our console just to confirm that we actually managed to get access to the　user who just recently registered so that the whole process worked. Now the very last thing we have to　do before we test our app is we actually have to enable this particular logging in method.

So we do that by going to our project on Firebase and then going into the Develop section and clicking　on authentication. So here we have currently no sign in methods　so let's go ahead and set one up under the sign in method tab. And you can see that you can use a whole　wide array of different social sign ins or anonymous sign in even,but the one that we want to enable is the Email/Password.

So let's click on this toggle to enable it and go ahead and hit save.

Now we're actually finally ready to test our app and I'm just gonna click Run to do a hot restart.

And once it's ready, it's going to drop me on the welcome screen.

We're going to click on register and I'm going to register my first user as [angela@email.com](mailto:angela@email.com)　and the password is 123456.

Now mind you when you're registering a user, make sure your password is also at least six characters.

Firebase passwords needs to be at least six characters long, short passwords will cause the registration　to fail. Let's hit register and you see that I progress to the chat screen automatically and that's because　of that navigator that we put in here.

And once I'm on the chat screen, I should be seeing my email,　so that user who just registered, being printed into the console.

And if I take a look on Firebase and I go into the users tab, I can see that new user who's just been　created right here. So perfect.

You've created your very first user and you've registered them using Firebase authentication.

Now in the next lesson, we're going to look at how we can log in users who have registered by updating　the code in the log in screen.

So for all of that and more, I'll see on the next lesson.