In the past few lessons we've set up already　most of our user interface. We've got the background layouts, we've got pretty much all of the column　structure setup and we've also got our list view. But now we need to look at the second part of our mockup　and we need to create this little pop up thing that comes up from the bottom to allow our user to add　any new to do list tasks.

Now if we head over to the Flutter docs and if we take a look at the very bottom, our widget index, and you　can scroll through and you might come across something called a bottom sheet.

So bottom sheets slide up from the bottom to reveal more content.

That sounds pretty much like exactly what we want right?

So let's click on it to read a little bit more.

And it tells us that there's two kinds of bottom sheets: one which is persistent, so something that shows　information that supplements the primary content.

So a little bit of extra content but it stays visible even when the user interacts with other parts　of the app.

So that's probably not what we want.

We don't want it to stay there all the time　right?

So instead, what we want is a modal bottom sheet which is something that prevents the user from interacting　with the rest of the app because it's going to be highlighted and it's going to come up as a little　pop up where the user can type in their new task and then once they're done, it can pop away　and for us to go back to the main user interface. So the way that we create these modal bottom sheets　is with something called a showModalBottomSheet function.

So let's click on it to find out a bit more.

You can see that this function has two required properties.

It needs to take in the current context which allows it to look up the navigator in the theme to see　how it should lay out this bottom sheet to be consistent with the rest of your app and also it needs　a builder,　so a widget builder. And a widget builder is simply a function that takes the current context and returns　a widget.

And in this case, it's going to be returning a bottom sheet widget.

So let's see how we can implement it.

So if we head back into our onPressed,　well this is the part that should trigger that bottom sheet to pop up.

So let's call our showModalBottomSheet.

Now notice, you have to be careful there's a showBottomSheet which shows the persistent type, but the　one that we want is actually the little pop up that comes up with an animation as well.

So the context that we're going to pass in is of course the current context of our widget from our task　screen.

So in our case, it's actually not all that much that's going to be passed over because we don't have　much in the theme　but we do have a navigation stack. So we can tell it where it is in that navigation stack.

But we've also got this builder method and you can see that this thing wants a function to be passed　in.

So let's create a function up here. And this function　if you remember has to return a widget and have a context as an input.

So notice how this is actually really similar to our build method that we've been using all the time.

So the output is a widget. And let's give the method a name　so buildBottomSheet and then it takes an input which is of type build context and we can call it context　and then it's got a return a widget.

So in this case, it's going to return whatever we want to show inside our bottom sheet.

So what do we want to show?

Well let's say that we just return a container showing just a really simple thing.

And of course make sure that you can actually spell unlike me, hope you got Widget instead of Wiget.

So now inside our builder, we can pass in this method that we've just created which is called Build　BottomSheet.

And if we go ahead and add a semicolon, then we've now set up our showModalBottomSheet.

And it's pretty short actually if you just look at it.

But now when we press on the button, that's the moment of truth, and we get a bottom sheet that pops up　from the bottom and builds a container widget to show us.

Now how can I be sure that we've got actually our container?

Well let's give it some text, shall we?

Let's give it a centered text which is going to be, 'This is a bottom sheet'. Let's hit save.

And you can see we've now got access to this bottom sheet area that we can put whatever content we want.

And you can see how it darkens the rest of the user interface so we can't interact with the rest of it.

But as soon as we click away, then it animates and drops down to the bottom. And then it animates up, animators　down.

I could play with this all day, but I won't for your sanity.

So now that we know how we can create these bottom sheets, that's delete the little bit of dummy content　that we've got in there and let's go ahead and create some real content.

Notice how we could have this very wordy method here,

buildBottomSheet with a name and also have a return.

But we know that when we just want to return one thing,　so a single line item in this case a container, we can actually delete the semicolon,　delete the return keyword as well as all of the curly brackets and replace it with our arrow function.

Now we have a single line code right here.

We can make this even shorter by instead of having a named function buildBottomSheet,　we can just make it an anonymous function.

So let's go ahead and take all of this part and we're going to cut it and replace the function here　with what we had before.

So now we can delete the name and the return type　and of course this context input also doesn't need the type here either.

It can infer it from the context that we pass in from over here which is a build context type.

So now our code is very very short, painfully short almost.

But the functionality is still exactly the same.

And instead of creating a container and building out a massive tree over here, I'm actually going to　take this and create a brand new file inside our screens.

So let's create a new Dart file called the add\_task\_screen.

And here we're going to create the widget tree for what we want to see inside this little bottom sheet.

So instead of returning a container, I'm going to return a AddTaskScreen and I'm also going to import　my screens folder and the add\_task\_screen.dart file.

So now let's go over here and actually create something called AddTaskScreen　and inside this screen what do we want to have?

Well if we take a look at our mockup, you can see that we want to have maybe a title, so a bit of a text　widget and then a text input, something where the user can write the actual task, and then we can　just have a flat button that says add. But before we create all of this interface, there's just one thing　that's very obviously different. Namely in our mockup　we wanted to have curved edges for this bottom sheet similar to how we've got our list to make our app　look really consistent in its design.

But at the moment we've got this very square looking bottom sheet. Even though it's very cool, it's still　not exactly what we wanted.

So let's look at how we can create this.

Well we've got our container　and what I'm going to do is I'm going to give it a color that is going to be exactly the same as this　background color. On a Mac　there's something called a digital color meter.

And what it allows me to do is basically it gives me a color dropper and you can install similar color　dropper tools on Windows as well.

And I'm going to hover over here and see what the value for this color is. So at the moment it seems　like it's something like 117 right?

So red is 117 , green is 117 and B is 117 .

So now with these values I'm gonna go into chrome and I'm going to search for 'rgb to hex'. And you end　up with this little color picker tool in Chrome where you can change the values of these things. So we　had 117, 117 and 117.

So it's this gray color and the hex code is right here which is kind of #757575.

So let's go ahead and copy this over to our code.

So the color is going to be constructed using the color widget and it takes an integer value.

So if you remember from before, we had to do the 0xff to replace the hash tag with a pound sign and　then we end up with the integer value of our color.

So now if we hit save, you'll notice that this thing that pops up is now pretty much indistinguishable　from the background.

So we're going to use that as the first container and then inside this container, I'm going to add another　container.

Now bear with me, this seems a little bit weird　what I'm doing but it's kind of become clear very soon.

Now in this container, I'm going to add some decoration.

It's gonna be a box decoration and the color is just going to be a white color.

And then of course we have our border radius.

So now we can add a border radius with the top left and top right with a 20 circular radius for both.

And now if I hit save, you'll see that when this bottom sheet pops up, there's actually one bottom sheet　at the back which has that gray color that matches with the background.

And then there is another one inside that container which has the rounded edges.

So this allows us to have this little curved edge bottom sheet　and it looks really neat because it fits in with the rest of our design.

So now all we need to do is to add in the rest of the things that we see in our mockup.

So we've got a large-ish title that says 'Add task', a text field and a flat button.

So here's another challenge for you. Pause the video and see if you can recreate this exact layout using what　you've learned about widgets and laying out UI using Flutter widgets.

And once you're done, once you've given it a go, then come back and I'll show you my solution.

So pause video now.

All right.

So how did that go?

Well here's my version of how you could do this.

And there's a million ways of doing the same thing as always with Flutter.

So just because yours is different from my code doesn't mean that you got it wrong. As long as you get　this end result then you've pretty much achieved the objective. Inside our container

I'm going to add a child which is going to be a column because again everything is laid out vertically　one and top of the other.

And in this column it's going to have some children the first of which is going to be a text widget,　the second is going to be a text field widget　and the final one is going to be a flat button widget.

So now I've got three things and I'm going to give my text widget the name of 'Add task'.

So now when I hit save you can see my three widgets show up.

They all look really ugly but we're going to change it right now.

So, firstly the text widget. Let's add some style to it.

So let's add a style property which is going to be text style and I'm going to change the font size　to make it a lot bigger, to make it maybe 30. And then I'm going to change the color to make it our favorite　color　in this module anyways, the colors.lightBlueAccent and now it looks a bit better.

So onwards.

Next is our text field. And our text field is actually encompassing the whole length of this because when　I click on it, you can see the blue underline show up and the cursor is almost invisible at the very　left.

Now the other thing I want to change is when I pop up this little bottom screen, I want this text field　to be automatically already enabled because you can see that if I hold COMMAND + K to bring up my keyboard　or on Android you would toggle this little button right here, and I want it to automatically show up　this keyboard whenever it pops up so that I don't have to click on it. Given that there's only one text　field　it makes sense for us to automatically enable this text field so that they can start typing straight　away without having to have the extra step of tapping on this. And to do this　I'm going to add the autofocus property and I'm going to set it to true.

So now if I hit save and we open up our bottom sheet, you'll see that it's automatically enabled, this　bottom line is blue and our keyboard pops up every single time.

The other thing I want to do is to address the fact that are cursor is pretty much off the screen. And I　can bring it to the middle by simply changing the text align property to TextAlign.center.

And now you can see when I pressed the add button, the cursor shows up right in the middle and it's already　enabled and ready to go.

The final one is of course our flat button.

So let's give it some text by giving it a child which is a text widget and it's just going to read 'Add'.

And then I want to give it a color which is going to be the light blue accent.

But when I hit save and hot reload, you don't actually see it show up.

And the reason is because our flat button is currently giving us a warning namely that the parameter　onPressed is required.

And when the onPressed is not supplied for a button, then it default to a disabled button.

So the color isn't showing up because our button currently is disabled.

So let's go ahead and add the onPressed that it requires and here at a later date, we'll probably add　some code　so that we'll add our task to the list.

But right now I'm just going to leave it as it is. But as soon as I hit save and once we've got our on　Pressed and our flat button is not disabled anymore, you can see our background show up. And now I want　to fix that text.

So let's change the style to make it white,　and so text style and the color is colors.white.

All right so now let's hit save and the final thing that you'll notice is different between that one　and the mockup is this button goes edge to edge right?

Or pretty much the same width as the text field.

So how do we do that in our code?

Well we can change our column cross axis alignment to make it stretch from side to side and then let's　go ahead and change the text widget's　textAlign property to center as well.

Now notice the first property of text has to be 'Add Task'.

So if you try and add another property in front of it, it's not going to like that very much because　this doesn't have a parameter name. It doesn't have something like text widget with text something colon...

So it needs that positional data and its position must be the first in the widget.

So we have to let it be first.

And we have to treat it with extra special care.

So now that we've pretty much created everything that we have here, all we have to do is add some padding　around our column.

So let's go ahead and add some padding into our container right here which wraps our column.

And I would say that we could probably get away with simply just adding maybe all sides and let's add　good old 20 pixels and it looks pretty much like what we see over here.

Now you can of course add some padding between this field and add button if you wish but I think at　the moment that actually looks fine.

But of course feel free to modify the appearance as you like.

So now we have pretty much replicated our entire mockup.

We've got our area at the top which describes our list,　we've got our actual list view with some list items,　we've got a pop up an animated bottom sheet and we've now got everything inside that screen show laid　out as well.

So now we're pretty much ready to move on to the next stage where we look at how we make it all work　and make it all come together.

And the first thing I want to address is why is it that our checkboxes don't work?

So have a think about that and to find out the solution,

I'll see you on the next lesson.