All right.

So first things first let's crack open Android Studio and start a new Flutter project.

We're going to be building this one completely from scratch.

Now once you've chosen Flutter application, click next and we're going to name our project to todoey\_flutter.

And once we're done here then let's just click next and finish.

All right.

So here's our brand new application with the usual bells and whistles.

So we've got our main.dart which contains the sample counter app which I'm going to basically delete

pretty much most of it.

So I'm going to get rid of my home page.

I'm also going to get rid of myApp and instead I'm going to create it pretty much from scratch.

So I'm going to use the shortcut stless to create a new stateless widget and I'm going to call it My

App so that this matches with the app that I'm going to run.

Now whenever you see a red box around here this is created by Android Studio to make it easy for you

to tab around and to change certain things.

But if you want it to disappear just hit tab and it will go away.

Now inside my app instead of building a container, I actually want a material app.

So I'm going to add a material app which is going to have a home route, so the first screen that we're

going to see. And I'm going to call this a TaskScreen and I'm going to create it right here.

Now this has got a red squiggly line because obviously we don't have a task screen yet

so let's go ahead and create it.

So inside our lib folder I'm going to create a new folder and I'm going to call it screens.

So this is where we're going to be organizing our screen files. And then we're going to create a new

Dart file which I'm gonna call task and what I want to show up in the first screen of our app is going

to be something called TasksScreen.

And I'm going to create it with my round brackets.

Now I've got my red squiggly lines because TasksScreen doesn't yet exist.

So let's go ahead and create it by creating a new folder where I'm going to create all my screen files.

That's where I'm going to organize all my screen files and I'm going to create the first one which is

a Dart file that's going to be called

tasks\_screen and I'm gonna hit enter.

And now we have our tasks\_screen.dart.

So here I'm going to import material which I pretty much will always need when I'm creating UI.

And then I'm going to create a new stateless widget which is going to be called TasksScreen.

Now notice that in this case I've got an S here

so that we've got a plural,

so a large number of tasks which is show up on screen but just check to make sure that they match exactly.

And if you want it always easy to just copy and paste as well.

So we've still got a red squiggly line because we just need to import our screens/tasks\_screen.dart

And now all the arrows are gone and we are no longer in trouble with Android Studio.

So finally let's go back to our top screen and let's start building out this screen.

We don't want to create just a container.

So I'm going to run my app as it is so that we can build while we see what we're building so that we

can paint our pixels onto the screen if you will.

So now that our build is done we've got this black screen which is not very exciting but we're going

to jazz it up very shortly.

So what is our app going to look like?

Well in the download section or the resources for this lesson you'll find a download to the to do a

mockups.

So I created these in Sketch which unfortunately is an application that's only available on Mac.

So I've actually exported my mockup to a PDF

so that you can download it and be able to zoom in and look around and pick out the colors and look

at what you need to design while you're doing it in Android Studio.

But I'm also going to be doing it step by step with you.

So let's take this design and let's transfer it into widgets.

So firstly we need to have a background color which is just a light blue accent color.

We don't want to have a container on screen but instead notice that we have our material app as the

top most widget.

Then very shortly it's followed by a task screen which takes us over here and we're going to build a

scaffold as usual.

So our scaffold is going to have a background color of light blue accent and as soon as I hit save we

see our color rendered on our simulator.

Now in this case, our scaffold is not going to have an app bar but it is of course going to have a body.

So the body of our scaffold is going to be a column widget because notice how in our mockup we've got

an icon,

we've got some text, some more text and then we've got this other component down here. So everything's

kind of more or less stacked vertically so I think a column will work quite nicely for this scenario.

And inside our column, we're going to have a bunch of children and the first child we'll have is this icon

here.

Now in order to grab the icon we're going to tap in to the icon widgets and the icon data is going to

be icons.list which is this one here, just looks kind of like a to do list.

So now that I've added my icon, you can sort of see it just right over here in the top left corner. And

to make it a little bit easier,

let's add our text here as well.

Why don't we put the Todoey title?

So let's add a text widget and the string that's going to be inside the text widget is of course Todoey.

And once I hit save, then you'll see it's sort of all mangled up here in the top left corner.

Now one option is we could wrap our column inside, of course, our safe area widget which we normally use

to stay safe from the parts of our phone where the user can't really interact with.

So on the iPhone, it's around the notch. On the Android phone it's going to be around the status bar so

where it says the battery and the time et cetera.

And also this part on the iPhone is kind of the new home button

so that's also going to be outside of the safe area.

But for me this kind of margin is not really enough.

I kind of want to push it down a little bit further because all our content is quite heavily padded

away from all the edges.

So instead of using a safe area, instead I'm actually just going to use a container because in our containers

remember there's always the padding property.

So let's pad our column and give it some edge insets.

Now instead of using the one that I normally use which is 'all' by giving it the same amount of padding

around all four sides, instead I'm going to choose 'only' in this case.

And this is because I want to have different amounts of padding for the top and the rest of the three

sides.

So first I'm going to start off with my top value which is actually going to be a whopping 6

because I want to push it down a lot. And then I've got my left which is going to be 30, my right which

is going to be 30

and the bottom which is going to be 30 as well.

So now let's hit save

and we see this push down far beyond all of this sort of busyness and it's now inside the sort of body

of our actual app.

Now all we need to do is make this look more like this.

So you think it's really far away but actually it doesn't take all that much code at all because everything

the widget

we just need to wrap some things inside widgets.

So the first thing is our icon is inside this little round circle thing.

And the easiest way to generate these profile icons or avatar icons or you know in this case just a

circle, is to wrap it inside a circle avatar.

So we've seen this before so I won't go into it in too much detail. But if you want to have a quick revision

of what circle Avatar does or in fact any of the widgets I talk about,

be sure to have a look at the Flutter docs which, as always, is constantly being updated.

And it has more information than you'll ever want on any of these widgets.

So now that I've created my circle avatar, it has a icon inside it.

But I also want to change some other properties about the circle avatar namely maybe the background

color.

So let's change it to colors.white.

And now we've got a nice white icon

so looking pretty similar right? Now all we have to do is make it a little bit bigger.

So let's change the radius of our circle to let's say 30 pixels.

So that ends up being probably a diameter of about 60.

So it looks a lot bigger.

And now in contrast, our icon looks a little bit too small

so let's change our icon size as well.

But there's a property called size in icon and let's change that to 30 to match with the parent circle

avatar.

Now remember whenever you are stuck or can't remember what properties a widget has, the easiest way is

to hold down COMMAND on Mac or

CONTROL on Windows and then just hover over the widget and you'll be able to see all the properties

that you can change.

Now if you want more information on any of these properties, then simply having your cursor on the widget

and then going to our Quick documentation, which on Mac in my case is F1, but on your system it might

be different.

So have a quick look under View,

Quick documentation to see the shortcut. And that will bring up of course the docs as well as all the

properties and a way of being able to not just see what you can do but also understand the widget a

little bit better.

And finally the final step is of course you can hold down COMMAND on Mac or CONTROL Windows and just

click on it to see the source because Flutter's open source

right?

If you ever want to figure out how anything works or how to modify it,

that's the final all encompassing way of going about that.

All right.

So icon is looking pretty good but if you have a keen eye for design, you might have noticed that this

blue is a little bit different from this blue and that's because by default all of our material apps

get a blue color, which is colors.blue, as the main theme color. But in our case we want to switch

that out.

So let's change the icon color to also colors.lightBlueAccent to make it match. See, looks much better.

Now

the next thing we want to do is add maybe a little bit of space between the icon and the Todoey title.

So here I've pretty much got just a 10 pixel gap in between.

So because we're working inside a column, it's very easy to add that into the column as a sizedBox.

You don't have to mess with wrapping things deeper and deeper inside padding widgets.

So let's add a sized box with a height because that's the only dimension that we're concerned about

inside a column which is vertical.

And now we have a little bit of space, a little bit of breathing room, in between our user interface icons.

So now let's work on our text widget.

So we've got some writing, some string, in there which says Todoey. But let's change the style of our

text.

So let's change the text style widget to have a color of colors.white and then what else have we got?

We've got quite a large font size and it's also a bit heavier on the font weight,

right?

So let's change the font size to... let's make it a really big text maybe 50. And let's change the font

weight as well,

and I'm going to change this to a .w700

so this is a 700 weight which is quite heavy and it's a little bit more heavy than bold.

So you know that we could use bold, normal, light et cetera.

But you can also have more fine grained control by using the 100, 200 all the way up to 900. So let's choose

a 700 which should add a little bit of thickness to our title and make it look very important.

So now you can see that everything is kind of shifted in our column though right?

So what's going on?

Well let's take a look inside Flutter inspector and let's show debug paint to see where everything is.

Now you can also of course just change the background color of the column or wrap it inside a container

or do funky things.

But this is the easiest way of seeing what's happening with our layout and those of you guys coming from

Web Dev.

this is pretty much second nature to you, going into the Chrome developer tools or the Firefox developer

tools.

This is what we're expecting to see.

So you can see that here is our column and inside it we've got two items.

One is the circle avatar and the other is our Todoey title.

So inside within the column, both of these things are centered by default.

But the column only takes up as much width as the children need.

So if this to do a title it was gigantic

let's say I change it to, I don't know 200, then you'll see it forces the column to expand as far as it

can within its parent which is within the scaffold.

But in our case we don't actually want the child to change.

We don't want to change the dimensions of this title quite happy with that.

But instead what we want is to make everything inside our column to be aligned to the left of the column

which is also known as the start of the column.

So let's switch that off and let's go ahead and change a property on our column which is of course our cross

AxisAlignment and I'm going to choose start.

So now let's hit save and you'll see that everything now moves to the left of the column

even though the size of the column has not changed.

All right, cool.

So now if we compare, we're getting pretty close

right? And it looks pretty similar.

Now let's go ahead and add the next piece of text. So below this text widget, let's add another text widget

and this one is going to show us how many tasks we have left to do.

So I'm going to just add a default sort of hardcoded string of 12 tasks and then I'm gonna change some

of the styling of this text widget by changing its color firstly.

So I also want this to be white

and I'm going to change the font size to 18.

So just a little bit bigger.

And that should do perfectly for now.

So now we've got the top part of our app laid out.

The next thing to look at is this bottom half where we've got this white area with rounded corners and

it looks really neat because it looks almost like we've got some sort of a little to do list card that

is sitting on top of to do list name or to do this title.

So how can we create this?

Well we could create maybe a container.

So in our column we've got currently four things.

And right at the end of that text widget, let's try adding a container. And our container is going to

get a color.

And it's going to be a white color,

so colors.white.

But notice that nothing actually shows up when I hot reload

and that's because containers only show up when they have children with a dimension.

In this case this container doesn't have any children at all.

But we can force it to show up by giving it some dimensions.

So let's give it a height of 300.

And we can see our container now right here with a height

and it's also taking up the full width of our column.

So how can we turn this white square into this nice little rounded corner card? So one of the properties

that a container has is something called a decoration.

And so we can set it to a box decoration widget which gives us a whole bunch of ways of manipulating

the appearance of a box.

So firstly, we can move the color property over here because actually if we tried to set a color on both

the box decoration property as well as the container color property,

notice that if I hit save

now it's actually going to crash.

And it tells me that there is an issue because either the color property has to be null or the decoration

has to be null.

You can't have both a color and a decoration because the color argument, this one, is just a shorthand

for decoration, BoxDecoration color.

So this is basically a shorthand for creating a new box decoration with a color.

So it's exclusive or. So you can only have one or the other.

So now let's hit save and we'll see it turn into a red box,

but we want a white box.

So now that we've got our box decoration, then we can go ahead and give it a border radius.

So a border radius will turn our box from a square box into a rounded corner box.

But I don't want rounded corners on all four sides because then it'll look like I've got some sort of

large icon living on my screen.

Instead I only want to curve the edges on the top.

So the way I do this is by creating a new border radius widget and I'm going to use the 'only' method

to only set up the top left and the top right.

So this is going to take a radius, so radius.circular.

And I'm going to give it 20 pixel border radius on the top left and the top right,

so this is going to be another circular radius of 20.

So now let's hit save.

And you can see we've got our little curved edges at the top and we've got a completely square or 90

degree bottom to our little container.

It's looking pretty cool right?

But how can we get it to extend all the way down to the bottom because at the moment it looks like it's

just a bit cut off?

Well one way is by increasing the height.

But let's say that I made the height too large, then it's going gonna go off the screen and I've got

my overflow problems.

So instead, what we want to do is delete that hard coded height

and instead we can wrap our container inside an expanded widget which is going to expand its child, the

container, so that it can take up as much space as there exists.

But there's still one problem because at the moment in our card, there's a padding on all these sides

over here, over here and over here. But in our mockup, you can see that it goes all the way to all the edges.

Now why is that?

Why does it have padding?

Well it's because that container lives inside our column and our column lives inside a container which

has some padding, namely 30 pixels on the left, 30 on the right

And 30 at the bottom.

So in order to free our container from this padding, we actually have to take it out of this column and

instead I'm going to create another column - this column with the Todoey, the task, the icon - is going

to be nested inside another column.

So I'm going to wrap my container inside another brand new column.

So now inside this column, I've got a container which contains all of this. But I've also got my expanded

widget which has the white curved card.

And now if I go ahead and take off the debug paint, then you can see I've achieved what I've been looking

to do which is that curved edge card like what I've got over here.

But there's one problem though because this column seems to have centered everything by default, because

this new column has its cross origin set as center.

So in order to move everything back to the left again in this new column I also have to set the cross

axis alignment to start. Brilliant.

We're getting pretty close to our final desired mockup.

All we have to do now is add a floating action button which is pretty easy. In our scaffold

we've got our background color, our body and we're now going to add a FAB or a floating action button.

And it's really easy to just type FAB for Android Studio to pick it up for you because I'm just too

lazy to write floating action button.

I mean maybe they should just call it a floating button or floating circle.

But anyways now inside this floating action button, I'm going to give it a color, a background color property,

and it's going to be our favorite color which is light blue accent and it's also going to have a child

which is going to be that add sign,

so it's gonna be another icon.

It's called icons.add.

Simple.

Now we've got our floating action button and pretty much most of our user interface laid out.

All we have to do now is to figure out how to create a scrolling list that we can embed inside this

container here. And in that list is going to be a whole bunch of the list items with a name for our task

and a little checkbox that we can check off. For all of that and more,

I'll see on the next lesson.

