In this lesson, we'll continue making additional cosmetic improvements to our app to improve the styling.

In particular we will change both the colour and the alignment of our message bubbles dependent on the

sender.

But there are some other issues that we have to tackle too.

For example, we need to ensure that the hero image on our registration and login screen can handle different

screen sizes and aspect ratios.

So here I have our app running both on Android and iOS and the first thing that I want to show you that

we have to fix is, if you notice that on iOS especially in the later iPhones, the aspect ratio is a little

bit different to

Android. So we've got quite a lot of space in the vertical with less on the horizontal.

So this means that when I go into the register screen in the Android version and when the keyboard gets

popped up through that little icon and I enable the virtual keyboard or the soft keyboard, you can see

that much of our button is actually off the screen.

So how can we fix this?

Well we can actually wrap our hero widget inside a flexible widget and what this does is that it says,

'Well, this is the size that it should try to take up,

so about 200 pixels.

But if it can't, then it can be flexible about it and be smaller so that other parts of the screen is

visible.'

So it's a good idea to wrap our log in screen hero inside a flexible widget as well so that we cover

that basis too.

So now let's hit save and if we now take a look at our emulator, you'll notice that our hero widget which

contains our image will actually get smaller when the keyboard pops up and we pop down the keyboard

then it'll come back to its actual size.

So this way we don't have any of those terrible yellow and black stripes on screen telling us a part

of our UI has actually overflowed.

Now that we've done that, we will change both the color and the alignment of our message bubbles depending

on the sender.

This is quite important from a user experience point of view.

Let's take a look at what our app is like to use with more than one user.

I'm going to log in as angela@email.com on my iOS simulator. And then I'm going to go over to

the emulator and I'm going to register a new user. We'll call it jack@jackbauer.com and let's

type a password and hit register.

So now I've got both users shown on screen and you can tell that if I type a message over here, say 'Hello

Jack' and send it, then you'll see that that message appears over here but not mentioning the order, which

is out of order, which we'll fix.

But it's also really hard to tell who sent that message right?

Like it looks like it's all from the same person even though there's lots of people in this group chat.

A much better user experience would be something like this.

Here our message bubbles are both aligned and colored differently depending on the sender.

Also you can see that we've styled the corner of the message bubble to point towards the sender.

This is what we're aiming for

by the end of the lesson. Let's tackle the appearance of our message bubbles on our chat screen first.

Instead of having a bubble,

I want to have it have a little almost like a speech arrow right? So that it's a bubble

but then there's a little corner here showing which side it came from or showing which sender it came from.

To do that we have to split up our border radius for that blue material.

So instead of just having a all round circular border, we can actually change it to use something called

BorderRadius.only.

And this allows us to set a different border radius for each of the corners. So we can say that we want

a top left of probably circular 30,

and we want a bottom left of the same 30 circular radius. And we finally also want one on the bottom

right

that's also going to be the same 30 pixels.

So now, we should have a broader radius that has a round border for the top left, bottom left and bottom

right

and if we had save now, you'll see that our speech bubbles now have a little pointy part that points

to the user that sent it.

So now that we've got it like this, it would be nice if we could change the appearance for any messages

that came from other users other than ourselves.

So for example, the messages could have a different color and they could maybe be on the left rather

than on the right.

And the little pointy arrow could be pointing to the top left corner and all sorts of things that we

can change to differentiate which messages came from us versus from other people.

Now how can we do this?

Well we know that over here, we have this thing called loggedInUser

right?

The current user saved inside this Firebase user object.

So that means that if we move this loggedInUser out to the same place where we've got our Firestore,

then we can access it inside our messages stream. And what we can do here is once we've gotten access

to these senders email, well we can also create a currentUser and we can send that to the logged in

user's email address.

So we've got an email of the person who sent this particular message that we're looping through and

we've got the email of the user who's currently logged in.

So it's not difficult to compare whether if these to match to see whether if this message that we're

going through at the moment is in fact sent by the person who's currently logged into the app. We could

simply just make an IF statement checking if currentUser is equal to messageSender.

Well that means that they match,

right?

This means that the message is going to be from the logged in user.

So how can we send a variable over to our message bubble?

Maybe we can have something that's a boolean

called isMe and it could be set to True or false depending on whether if the message is from the current

user.

So we have this isMe variable and we initialize it when we create our message bubbles. And we can send

it over based off our IF statement.

But instead of using an IF statement, we can just use the condition. The double equals will test for equality

between the current user and the logged in user's email and give us either true or false.

If the currentUser is equal to the messageSender

and if that's the case, then isMe should be set to true.

And if that's false, then isMe should be set to false.

So now that means inside our build method for our message bubble, we have access to this boolean called

isMe.

So we know when we're building a bubble that consists of a message from me.

So how's a challenge? I want you to change the color of the message bubble based off that property.

So if it is me, the bubble should have that light blue accent. But if it isn't me, then it should be a

white bubble instead.

So pause the video and see if you can complete this challenge. All right.

So again we're going to need to use that ternary operator.

And now we're checking the value of isMe. If it's true, then my messages will be light blue and if it's

not sent by me then we're going to set it's background color to white.

So now if we hit save and once you've restarted the app, you'll see that all the messages that are sent

by the current user who's logged into the iPhone, which is angela@email.com, is blue. But all the

ones that were sent by other users like angela@gmail.com or jack@jackbauer.com, well they're

white.

The only problem is that we can't see the message in it because it's also white.

So let's go ahead and change the color of the text of the message bubbles which are sent by other users

and we'll change it to a black54.

So we can delete our IF statement now and inside where we have our TextStyle, instead of having it all

as white,

we're going to check to see if isMe is true, then it's white but otherwise the color is going to be

colors.black54.

Now the next thing that we want to change is to move all the messages that are sent by other people

over to the left so that we keep all the messages that we've sent on the right.

So pause the video and see if you can figure out how to do that.

All right.

So we've got this property for our messages which is called CrossAxisAlignment. And it's by default

set to end which means it's on the right side. And we can change this to change it to CrossAxisaAlignment.

start, if we want it to be lined up on the left.

So now if we restart our app and log in again, you'll see that the messages are sent by other people

are on the left, they have black text and white background and it's now looking pretty good.

The only odd thing out is this little corner here.

It would be great if this little straight corner was over here pointing at the center of the message

on the top left corner when it sent by somebody else. So pause the video and solve that one last challenge.

All right.

So we've got this thing called border radius which is determining where that point is.

And we're basically saying that it's circular on the top left, bottom left and bottom right.

Which means that the pointy bit, the only bit that's still left as square, is the top

right. Now we need to change that so that when it is me, this is the case. But when it's not me

well then in that case, we have to provide a different border radius.

So we're again going to go for a BorderRadius.only

and we need to have a bottom left and bottom right

both as rounded corners.

And the only one that has to be a non-rounded corner is the top left.

So the last one that we have to make rounded is the top right.

It's kind of confusing and there's a lot of corners.

But if you think about, it it should make sense.

So now when I type a message over here as jack@jackbauer.com and I hit send, then it comes in as a white

message.

But if I type over here as angela@email.com, then you can see it comes in as a blue message and

all the bubbles are looking exactly the way they should do.

Good stuff.

So what else is left?

What other issues should we fix?

Well let's think about our user experience again.

Typically in a chat app you would expect to see the most recent messages at the bottom of the list.

Also when a new message arrives, you would expect your screen to update and show you the newest message.

In other words, we would need our list view to scroll down to the bottom every time it updates.

Speaking of having the newest messages show up at the bottom, why are our chat messages not ordered chronologically?

Why all the orders are so strange, why is it coming in right down here?

Well this is because we initialized our Firebase database a long time ago and we've been doing various

changes and we've been messing around with it throughout this module.

So in order to fix the ordering, the first thing we have to do is to go into Firebase and actually delete

the entire collection.

And because this is so outrageous, they actually ask you to confirm by typing in the name of that collection.

We're going to leave this collection as blank because we're going to be filling in all the data over

from our app. And you can see that as soon as I've deleted the collection, our apps have responded by

wiping the messages too.

So now over here, I'm going to type the inaugural message from the Jack account and I'll type one from

the Angela account.

Now at this point there's one other problem, which is why is it that the messages are being ordered in

such a strange way?

Why are they all appearing right at the top?

And you can see that as I add more messages in here, that at some point it's going to go beyond what

is visible on the screen. So you can see that that message already went down all the way to there.

So if I add another message, you can see that it's not going to show up and it's not going to scroll

to it.

So how can we make it do that?

How can we make it scroll to a position?

Well the way that we can do this is by adding a property to our list view.

So at the moment we have a list view that only has two properties, padding and children.

One of the properties we can add in here is something called reverse.

And if we set this to true, then you can see now my list view is sticky towards the bottom of the view.

So that means if I type a message in here, it doesn't really matter what it is,

it always goes right to the top but we always have the bottom of the list view showing instead of previously

when it was always trying to stay at the top.

So now we have a different problem though namely that all our messages get added but they go right to

the top.

So how can we reverse our list of messages?

Well when we get our messages from our snapshot.data.documents, we know that this is a list of

document snapshots so we can use something called 'reversed' on any list.

And what this does is it will reverse the order of the list.

That means that whenever we create a new message now, it will go in into our list, our list will get reversed.

So now when we hit send, you'll see it appears at the very bottom which is exactly what we want.

So all the new messages come in at the very end and it shows at the very bottom of the list which has

now reversed in order but is also the bottom of the list view that's always shown on screen without

having to scroll.

Now all we need to do is tidy up the last bits of our code including the parts that we commented out

previously when we were testing our code, and also deleting the parts which we no longer need including

the getMessages method and the messagesStream because we've now replaced it with our messagesStream

widget. And we can delete it from our onPressed as well.

Brilliant.

So we've made quite a few styling and user experience improvements in this lesson by configuring our

message bubbles and list view. In the next lesson,

we're going to tackle the last part of our chat app namely setting the rules around authentication and

security.

