Now that all of our design is complete, we're ready to actually write the functionality.

And as I mentioned in the beginning, the design is really the hardest part of this app and the functionality of actually calculating the BMI is in fact the easiest.

Let's get started by creating a new Dart file which is going to contain the functionality of calculating the BMI.

So let's call it maybe the calculator\_brain. And in this file we don't need any material components, but what we are going to create is a new class.

And it's going to be called the CalculatorBrain. And this class is going to have two properties.

We're going to pass it a height which is going to be an integer and we're also going to pass it a weight. Now the way that we're gonna pass over these properties is of course when we construct the calculator brain.

So let's create a constructor with the height and the weight.

So now that we have a constructor, we'll be able to supply the values for the height and weight when we create a new object from the calculator brain. And we can now start writing some of the functionality.

The first thing we'll need is to be able to calculate the actual BMI.

Now the BMI stands for the body mass index. And it's a way of normalising somebody's weight against their height.

So the formula for calculating the BMI is the mass or weight in kilograms divided by the height in meters squared.

But because our height is currently in centimeters, so we might have to do a little bit of adjustment here.

Let's go ahead and create our function which is going to return the BMI as a string and we're going to call it maybe calculateBMI. And this is going to take no inputs because we'll be able to really have access to everything we need height and weight when the calculator brain is initialized.

But it is going to do some work and the work that it's going to do is it's going to calculate the BMI which is going to be a double. And it's going to record the BMI and it's going to be equal to the weight which is already in kilograms divided by the height squared.

So as we saw earlier, the easiest way of squaring a number is by using the Dart math library.

So let's import that so that we can use the power function which takes a number and an exponent. So we're able to provide the height and then we're able to provide the power, which in this case, is to to be up to square the height.

Now remember I mentioned earlier that our height is in centimeters.

So in order to convert it to meters, we have to divide it by 100.

So now we have our BMI calculated from this super simple formula,we can now use it and convert it into a string.

So at this point in time, the BMI is going to be a super long value because we said it's a double. Now all that we want in our app is actually just a single decimal place.

So 18.something but not like a million lines long because it's so large it's gonna go off the screen and nobody needs that degree of accuracy. In order to convert our double into a single decimal place and also convert it into a string,there's a really convenient method. So we can write BMI.toStringAsFixed and this returns a decimal point string and we can specify how many decimal places we want as the input.

So let's use that method and say that we only want one decimal place and we can output this value using the return keyword.

So now our calculateBMI is able to calculate the BMI based on the weight and the height and it's also able to convert it to a single decimal point value and return it as a string.

The next thing that we want is to be able to provide a result based off that BMI.

So we're going to create another method that is going to return a string which is going to be the result and we're gonna call the method getResult.

So again it takes no inputs but it's going to check inside this method if the BMI is greater or equal to 25 in which case it will maybe return the words Overweight.

And then it's going to check else if it's not greater than 25, well then is it may be greater than 18.5.

Well in that case then the results should be normal.

And finally for all the results that are lower than 18.5, we're simply going to return

Underweight.

Now these interpretations are based off most of the common BMI charts which you can take a look at on

Wikipedia or elsewhere if you wish.

So at the moment, we don't have access to BMI though which is why I'm getting this error.

It doesn't know about BMI because the BMI that's calculated in here is limited to these curly braces.

It's only visible locally inside here.

But if we wanted to get access to it, we need to make it visible to this method.

So let's go ahead and create a private property up here.

So we're going to create it as a double and it's going to be a private variable. So we're going to add the underscore beforehand and it's going to be equal to nothing to begin with.

And then when we calculate our BMI, we're going to assign it to that value and then we can return it as the string as well.

And we're now able to use it in all the places where we need that BMI. But remember because this is a private value, we won't be able to access it from any other class other than the calculator brain because frankly it's nobody's business trying to get access to the BMI or trying to change it. If they need it they can get it through this method which we know only returns it in the way that we want it to, which is a single decimal point. But inside our CalculatorBrain we now have free access to it and we can use it to calculate the result but we can also use it to give an interpretation.

So let's create another method called getInterpretation and this is going to perform the same three checks.

So I'm just going to copy and paste it in here.

But instead of just returning overweight, normal, underweight, we want to be able to give the user a little bit of feedback about what their weight means. And maybe a little bit of advice if you really wanted to. To save you from watching me type for an hour, I'm simply going to copy and paste some interpretations that I've written earlier.

You can either use these ones or you can make up your own. It's your app so do it your way.But now we have three methods inside our calculator brain, one that will give us the BMI, one that will give us the results of the BMI and one that will give us an interpretation of the BMI.

So we're now ready to go and we're ready to use that class CalculatorBrain.

So let's go ahead and import the calculator brain into our current file and we can collapse all of those imports after we're done. And the time when we need it is the moment when the user clicks on the calculate button.

So inside the onTap is where we're going to initialize a new calculator brain object. And I'm just gonna call it calc to keep it short. And it's going to be initialized using the constructor that we built earlier on.

So it's going to expect a height, which will be the height that we got from our slider here, and also the weight that we've got from our input page as well.

So let's set weight equal to the local weight.

So now that we've created our calculator brain and we can now get the calculation for the BMI, get the interpretation, get the result, we want to be able to pass that over to the next page which is the result page.

So if we go over to the result page, at the moment it has no properties that we can access from the outside.

But in order for us to pass some data over when we navigate over here, we have to create some of those properties.

So let's create a final property that is going to be called bmiResult and this is going to be a string.

And then let's create another one that's going to be called the resultText.

And finally we'll have one that's going to be called the interpretation. And again we have our final properties which will need initialization.

So let's go ahead and create our results page constructor and I'm gonna mark all of them as required because frankly in order to load up the results page and make it not look weird,we have to make sure it has both the BMI result,the result text as well as the interpretation.

So let's mark all three as required.

We've got this.interpretation, this.bmiResult and this.resultText.

So now when we create our results page in the moment when we want to navigate over, we can now pass over the values that we get from our calculator brain.

So we've got three properties that we can initialize.

One is the BMI result,another is the result text and finally we've got our interpretation.

So the BMI result is going to be equal to calc using the calculator brain to calculate the BMI.

And remember this will return a string which is going to go straight into that property BMI result over here and we're going to be able to use it inside our build method.

Now the next one is going to be calculated using the getResult method and the final one is going to rely on the getInterpretation method.

So now that we've got all those strings from our calculator brain, we can go ahead and use it in our build method.

So instead of having these hardcoded piece of text, we can go ahead and actually add the result text,the BMI results and also the interpretation.

Now the final thing I'm going to do just to keep our style consistent with the one here, is I'm going to capitalize all the words for the result text here. And as always there's always a simple way of doing this.

And the one in this case is called toupperCase and this simply turns all the characters in the string to uppercase which is what we need. Now, I'm going to stop my app and I'm going to run it from scratch so it starts from the beginning.

So now let's test out our BMI calculator.

We know from coding it up and also from Wikipedia that the gender doesn't really matter. But we might be using it in another part of our app, so we can select our agenda and it will activate our cards. And we can also select our height so my height is 180 and let's adjust our weight and let's adjust our age and go ahead and click calculate.

So it now calculates my BMI, gives me a result telling me that it's normal, and it tells me an interpretation of my BMI. My BMI is rounded to one decimal place and it's displayed as this giant number.

Now if we wanted to try it again with maybe a little bit lower height, a little bit more weight, let's click calculate and you can see we get a different result and a different interpretation.

In this module, we've learnt a lot of things and it might be worth reviewing some of those key concepts especially if you're new to Dart programming. But I hope you had fun building this beautiful BMI app with me and I hope you're inspired also to take your Flutter to designs to the next level using all the various ways of customizing widgets for your UI designs.

So that's all from me in this module.

I see on the next one.

