The Computing Challenge 2023

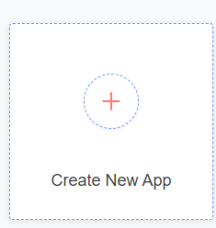
Lab Sheet 1: Introduction to Thunkable

This worksheet assumes you have created a Google account and installed the Thunkable App on your phone as previously instructed. If you have not done these things already, please do so now.

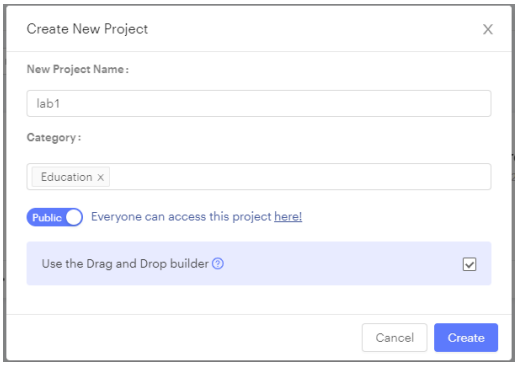
This worksheet will take you through the basics of creating and running your first app using Thunkable. You must work through this exercise INDIVIDUALLY, so that every member of your team is familiar with the concepts and use of Thunkable. A tutor will be in the lab for the whole session in case you get stuck – don't be frightened to ask them for help, it's what we are here for.

Remember that your assignment requires you to document this or tomorrow's lab sheet in your portfolio.

For your first lab we are going to keep things relatively simple and create an app with two character images which when clicked will speak to you in a lovely Lancashire accent. Login to Thunkable and within the **My Projects** section click on **Create New App.**



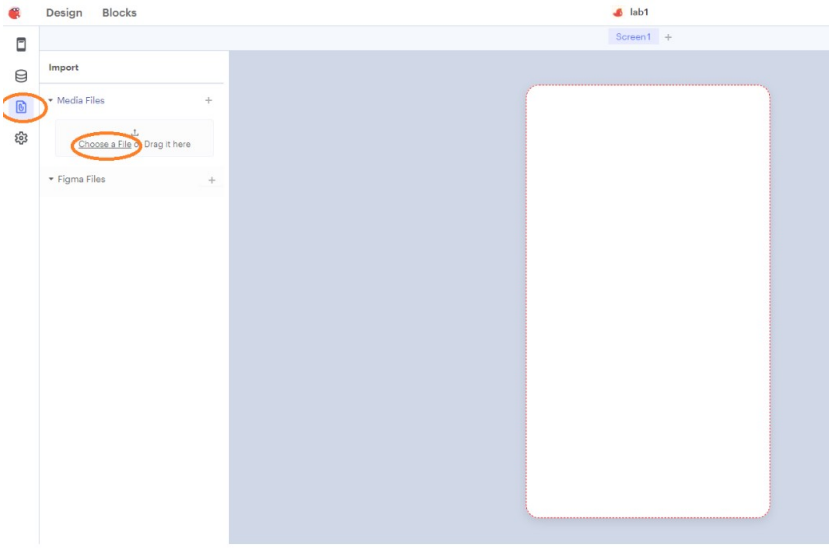
Give the project any name you like, but make sure that it is meaningful so that you can find it later e.g. CO1111 Lab 1. Select an appropriate category, we suggest Education, but it doesn't really matter. Leave everything else as default and click **Create.**



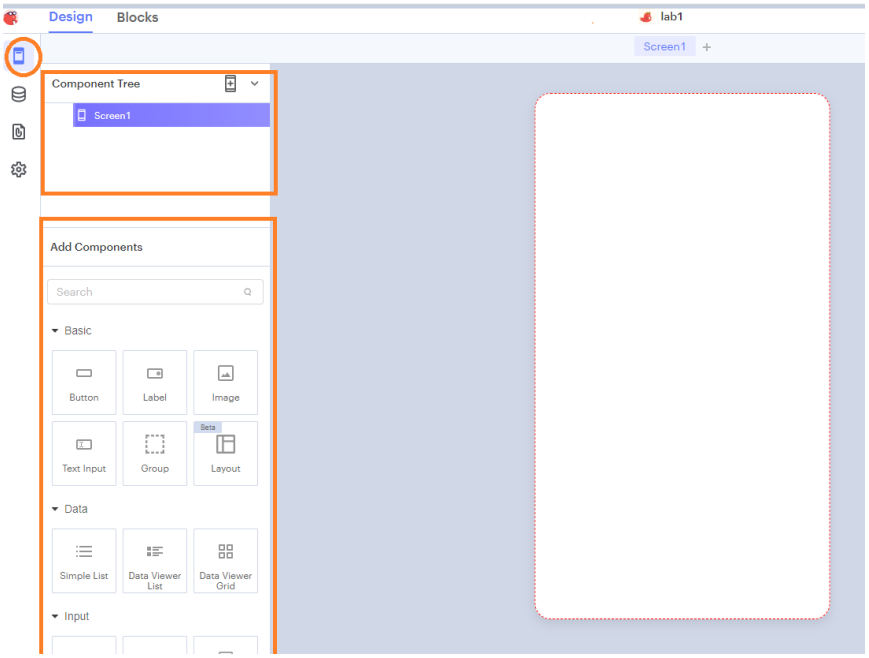
You will be taken straight to the Component Designer window to start building your app.

Firstly, let's import some media (the pictures and sounds) to use within our app. Download the **lab1media.zip** file from Blackboard and extract the files onto your computer.

Locate the **Assets** tab on the left-hand menu and click on the **Choose a File** link to upload the media. Browse to the folder where you have saved the media and select all of the extracted files to upload.

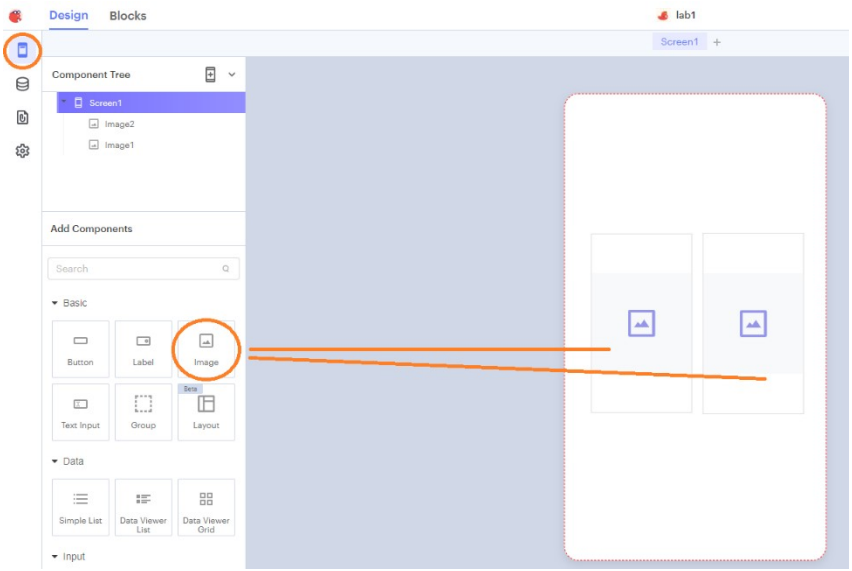


Switch back to the **Designer** tab. This pane has two areas... the **Component Tree** which shows the components we have selected for our screen in a hierarchical structure... and the **Component Palette (Add Components)** which holds all the components such as buttons, labels, etc.



At the moment, we only have one component (**Screen1**). This comes as default when we start a new project. We will start by adding a background image to **our screen**. All components have properties associated to them, such as size, colour, background, etc. However, not all components have the same properties. Select **Screen1** in the **Component Tree** to see it's associated properties in the property pane on the right-hand side. Click on the **Background** property and select the **Image** tab, and then click the file selection box and choose the image 'Eiffel.png'. Voila, your screen now has a beautiful Parisian background 😎

Next, we are going to add the two character images to the app screen. Click on the Designer icon on the left-hand side of the screen and locate the image component. Drag two of these onto the screen and place them side by side.



You will notice that these two image components are now listed in the **Component Tree** under Screen1. The components come with default names (image1, image2). When we start to program in the Blocks Editor, we will need more appropriate names so that we can identify our components. Change the names now to **JulieImage** and **JonathanImage**.

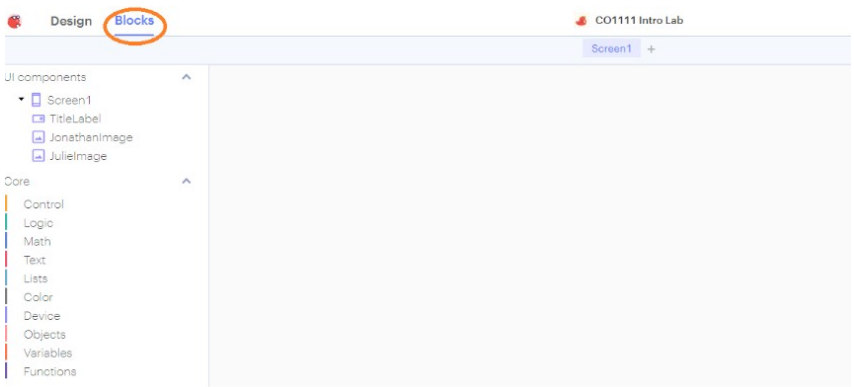
Click on each of the image components to see its associated properties. Within the properties editor change the **Picture Resize Mode** from **stretch** to **contain** for each image. This fits the image into its container whilst maintaining the aspect ratio. We now need to select the image file for each image component. Do this by clicking on 'No file source' and selecting the appropriate image from the dropdown. Place the images to make them look like they are standing on the grass, and not floating in the air.

Now let's add some text to the screen. Back in the Designer component palette, locate the **Label** component. Drag a Label onto the app screen and place above the two images.

In the property editor look at the properties for the label. Change the **Name** to something meaningful, e.g. **TitleLabel**. Then change the **Text** to 'Click on a Tutor to start the conversation!' Experiment with some of the other label properties.

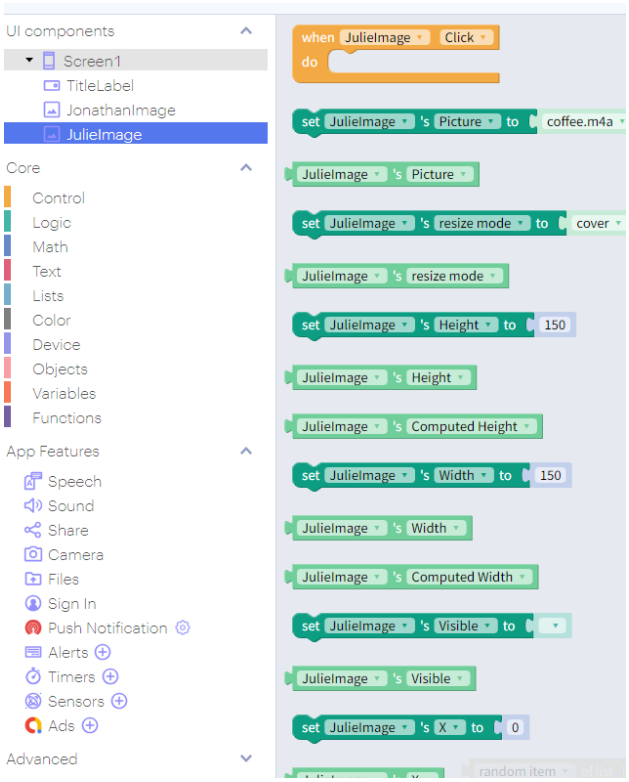
Your app should look something like this:

Right, now it is time to add our sounds so that when we click on a character, they will say something. For this we are going to need to do some programming. Click on the **Blocks** tab (at the top left of the screen) to switch to the Blocks editor. In the Blocks pane, you will see the components we added to the app (UI components) along with a list of built-in programming components.



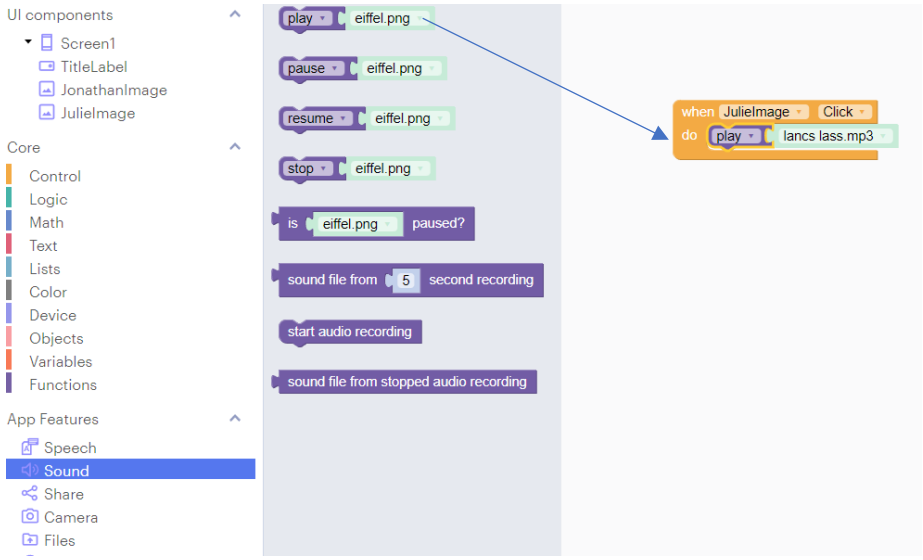
Click on the **JulieImage** component, and a list of Blocks will pop out to the side. Here you will see blocks to *read* and *change* properties related to the image (the **green** ones), and

also blocks to respond to the various possible events for the image (the yellow ones). The yellow blocks are called event handlers. You will notice there is only one event handler block for the image… a 'Click' event. Drag the **when JulieImage.Click** event handler block onto the canvas.



Note: If you ever drag something onto the canvas by mistake, you can simply drag it over the recycle bin in the bottom right corner of the screen to delete it.

We now want to plug some other blocks into it to make something happen when the image is clicked. Click on the **Sound** component in the **App Features** section of the component palette and drag out the **Play** block. Notice that sounds don't have any event handlers, but they have actions which they can perform, like play and pause, as well as their properties. Drag it into the space marked **do** in the **JulieImage.Click** block, and the two should snap together with a click. You will notice it has placed the name of one of your imported sound files into the block as a default. Check that this is '**lancs lass.mp3** ', if not then select it from the drop-down list.

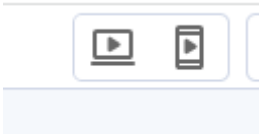


Repeat the above steps for the **JonathanImage** and pick the '**lancs lad.mp3**' sound file from the list.

Whilst we are adding sounds, let's also add some authentic French background music to the app. We want the music to start playing when the app starts up.

Click on **Screen1** in the components section and drag a **when Screen1.Starts** event handler out. Next in the **App Features** section click on **Sound** and drag a **play** block out and plug into the Screen1 event handler. Select '**FrenchMusic.mp3**' from the drop down.

Now, let's test to see if everything works! Locate the play icons in the top right-hand corner of the window…



You have two options… you can either test your app in the browser (**web preview**) or in the phone app (**Live Test on Device**). For the latter option you will need to open the Thunkable app installed on your phone and use the test code to connect the app to the project.

If all works correctly when the app starts running the French music will start playing and when you click on a character, they will speak to you. If something has gone wrong and you are struggling to figure out the problem, ask your tutor for help.

Taking Things Further

If you are feeling confident and would like an extra challenge, try this…

At the moment, you can click on any of the characters in any order and any number of times to hear the voice repeated over and over. If it were a real conversation one person (in this case Jonathan) would speak first and then the other person (Julie) would reply. Have a go at programming this for our characters.

Hint: Create a variable for each character which will hold a Boolean value (True or False) and when a character is clicked swap the value of each variable. You can then check the value of the variable before playing the sound.

Taking Things Even Further (and maybe earn some points)

Add an **accelerometer** component, and try and figure out how to make the characters say “Ahhhhhhh, an earthquake” when you shake your phone (the sound file for this in the media). Of course you will need to test this one on your phone.