The Computing Challenge 2023

Lab sheet 3: Leaderboard API Integration into a Thunkable Application

This worksheet assumes you have completed lab sheet 1 and 2. If you have not done these things already, **please do so now.**

It is very important that you read this worksheet carefully and always ask the tutor where you're unsure. **DO NOT** simply copy the code blocks off the page or from a student sat next to you without understanding what you are doing. You will not learn this way.

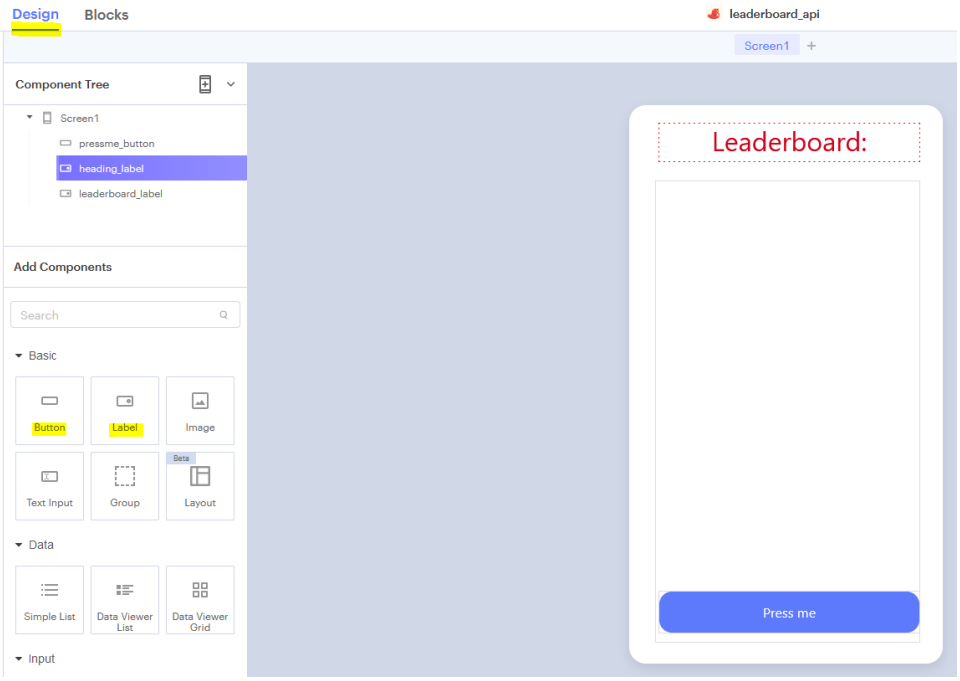
This is an INDIVIDUAL task not a team task, all team members are expected to complete this work as it is crucial to your understanding of how the app and server API work. A tutor will be in the lab for the whole session in case you get stuck – don't be frightened to ask them for help, it's what we are here for.

In the lecture you were introduced to the **leaderboard**. Today you are going to create a simple app that will get the leaderboard data from the server and display it on your phones screen.

Begin a new Thunkable project and call it 'Leaderboard' or something similar.

In the **Design** screen you should add:

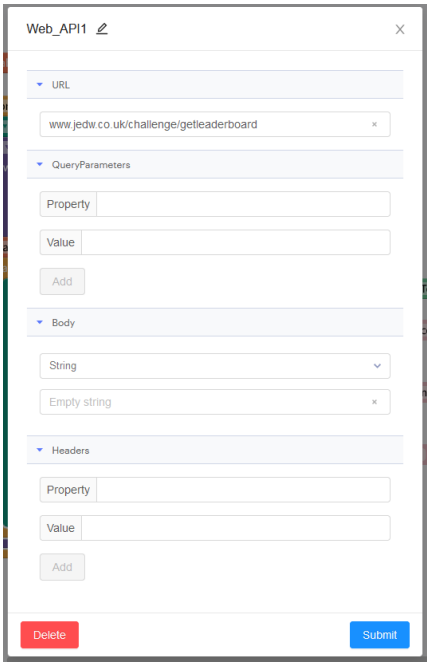
- A **label** component with the text 'Leaderboard'.
- A second **label** component with no text assigned, this will be where the leaderboard will be displayed. Drag out the second labels size to fill most of the screen.
- A **button** component at the bottom of the screen with the words 'Get Leaderboard' or something similar.



You can customise the colours as well as the overall look of the screen in any way that you wish. The example above is kept simple though.

So, that's our interface built. Now let's add some functionality:

Move to the **Blocks** screen and locate the 'Advanced' section. Find the **Web API** component and click to add to your app. You will be presented with the following screen:



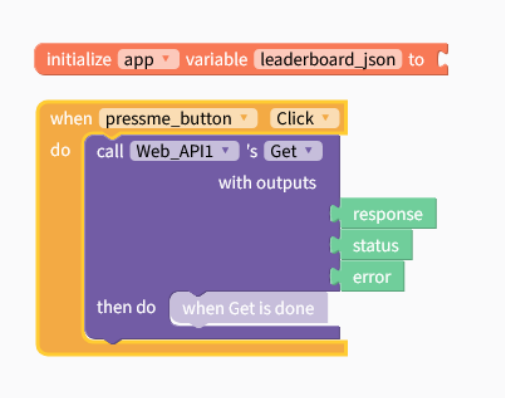
Set the URL to: **http://www.jedw.co.uk/challenge/getleaderboard**

The remainder of the options (QueryParameters, Body, Headers, etc..) are not relevant to what we are doing right now, so leave them blank and click **Submit**.

Go to the **Blocks** screen and add the following components:

- Initialize an **app variable** and call it something like ***leaderboard_json*,** the name doesn't matter (this can be found in the variables section under 'Core'**).** You do not need to provide a value for this variable.
- A **When-Do – Click** event for the button you have created. This can be found in the section related to your button under **UI components.**
- A **Call Web APIs Get** block, this can be found in the Web API section under 'Advanced'**.**

We want the leaderboard to be displayed when the user clicks the button, so drag the **Web API Get** block into the **Button click** event. It should look like this:



You will see that the **Web API Get** block comes with 3 local variables: ***response***, ***status***, and ***error.*** (A local variable is one which only exists within the confines of the block and not outside). The only variable we are interested in right now is ***response.***

When we call the **Web API** it will return the data that's held on the server for all of the teams. The data returned is in a format called **JSON** and it will be stored in the ***response*** variable. We will need to take the contents of ***response*,** (the raw JSON data), convert it into a more suitable format (an **object)** and copy it to our **app variable *leaderboard_json*** that we created earlier.

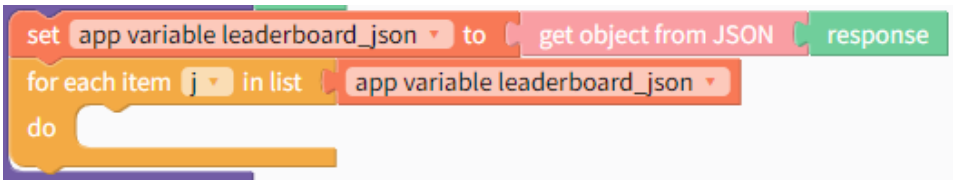
<u>Don't worry this is MUCH simpler than it sounds</u>

Firstly, get a **set app variable *leaderboard_json*** block from the 'variables' section and plug it into the **then-do** part of the **call Web_API** block. Next in the **Objects** section drag a **get object from JSON** block and plug this into the set variable block. Finally, make a copy of the ***response*** variable (right click-duplicate) and plug this into the get object from JSON block.

Now that we have our data in a suitable format, we can easily process it with a **loop** and display it in our app. To do this we will use a **for-each** loop, which is a common type of for loop used in many programming languages for iterating through data structures. Each time the loop goes around it will copy one piece of data (details for one team) from the data structure into a local variable which it calls ***j*.** We will then get the loop to add the contents

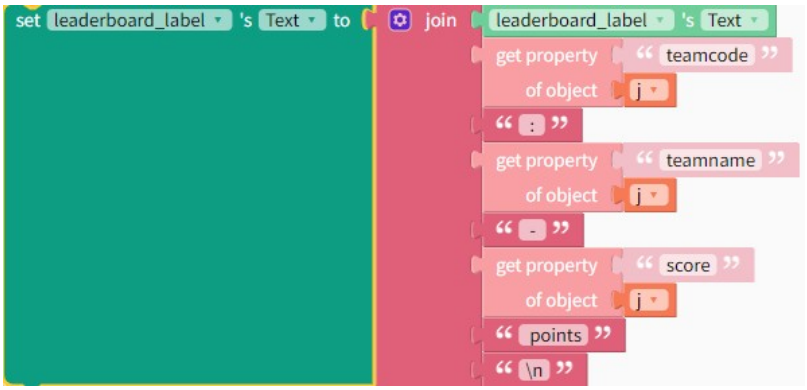
of ***j*** to the leaderboard label so that it will be displayed on the screen. After all of the data has been processed the screen should contain a list of all of the teams along with their scores.

Locate the **for each item in list** block and the ***app variable leaderboard_json*** block and add them as shown below:



To display the leaderboard, we are going to need to use a **join** block as seen below. During an iteration of the loop the variable ***j*** contains more than one value, it is an object containing one team code, one team name, and one score. In Thunkable we can access each of these values easily using the **get property of object** block

Have a go at building the block structure below and plug it into the **do** part of the **for each item** block:

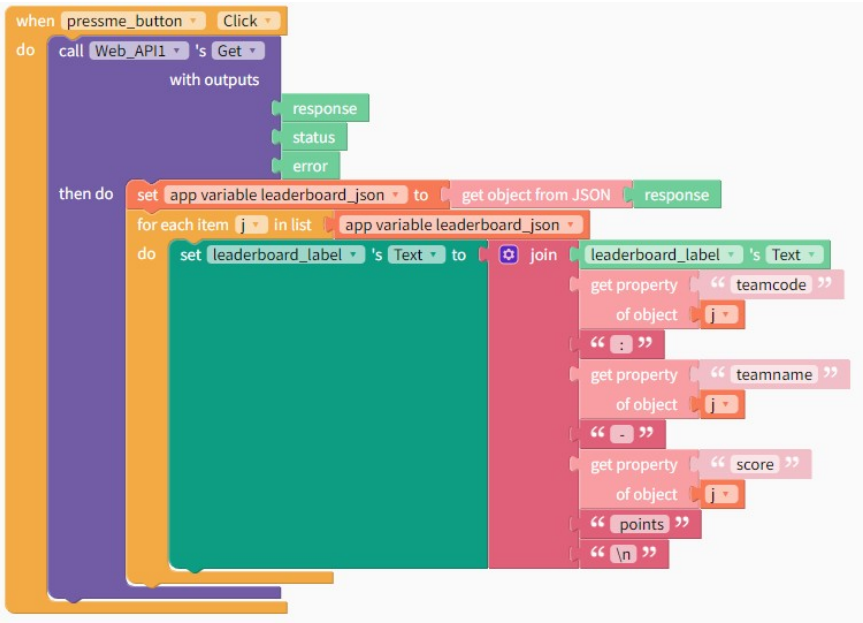


You may notice in the block code above that the first thing we join is the label to itself. Why do you think this is needed?

Answer: Without it all we would get would be the last team on the screen. The reason for this is quite simple. Each time the loop iterates the **set leaderboard_label text** block would overwrite the label rather than add more text onto the end. Therefore, the first thing we must join is the labels current content each time the loop iterates.

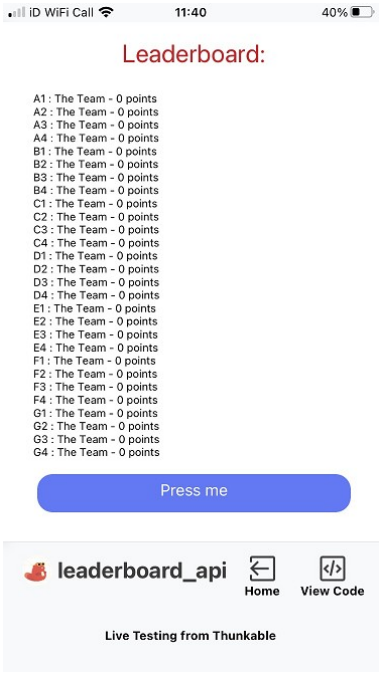
Lastly you may notice the text string ***n*** at the end of the join. This will NOT be displayed on the screen; it is a hidden character, the **newline** character which causes a **line break** in the text.

When you have completed the above steps your final block code should look like the image below:



Check your code carefully and test. To test this, you are going to need to use the Thunkable app on your phone. The in-browser web preview does **not** work when using the **Web API** functionality.

If all is well, you should have something like the app below. If you get an error, you should go back and check the blocks. Ask your tutor for help if you get stuck.



If your screen looks like the one above Well Done!

Now have a read of the API documentation here: http://jedw.co.uk/challenge

Taking Things Further

Some of the more eagle-eyed of you will notice that the list of teams is not sorted. So, if for example a team has more points than another the list will still appear in the same order as it came off the server regardless of scores.

Sorting the list is quite difficult, for now at least. You would need to build a function to sort all of the data within the app variable ***leaderboard_json*** into ascending order of each objects 'score' property.

Search the web for how to implement a ***bubble sort***. Our leaderboard is more complex because we are sorting a complex data structure and not something one-dimensional like a list of numbers.

REMINDER

The Friday Showcase will take place in tomorrow's lab. Spend the remainder of this lab preparing for your showcase.