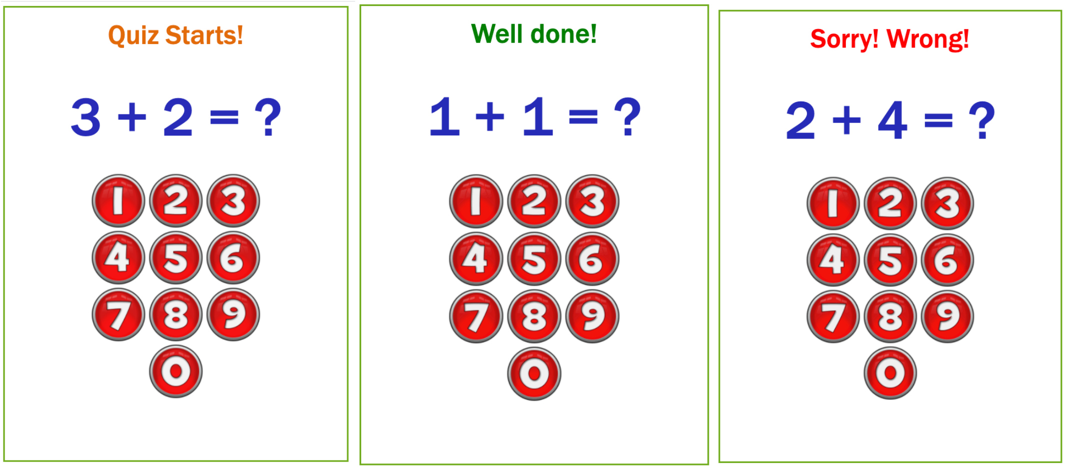
Lab 2. Math Game Version 2

Here are the user stories for version 2:

- The buttons should respond to user’s action.
- A random quiz should be given. The two operands should be within the range of 1 to 4.
- If the user chooses the correct answer, give a proper greeting message; otherwise display an error message.

2.1. Demo Video

At the end of this lab, you need to achieve an outcome as shown by the screenshot in the demo video (available at QMPlus). You should aim to produce an interface that is identical to the given showcase. In summary, there will be three types of display, as shown in Figure 1.



Quiz Starts!

3 + 2 = ?

Well done!

1 + 1 = ?

Sorry! Wrong!

2 + 4 = ?

Figure 1. Three Possible Messages During Game Play

2.2. Design Thinking

JavaScript (JS) will need to be used to implement the user interaction. A function check_answer() should be written in JS which can be called by the button objects.

You may wish to follow the pseudo code below to write the JS code:

```
Call new_Quiz() function;

Define check_answer() function:
    Retrieve button ID as a string;
    Calculate result=left+right;
    If result turns to string = button ID, then:
        Display the right message in the TopBar box;
        and all new_Quiz() function;
    Otherwise display the wrong message in the TopBar box;

Define new_Quiz() function:
    Declare two global variables: left, right. Both are random
    numbers in the range of 0 and 4.
    Print the math question in the quiz box.
```

It would be easier in coding if the buttons are named directly after the numbers: 1, 2, 3, …., 0.

Also note that there are requirements on where to place this piece of JS code:

- It should be placed after the **TopBar** and **Quiz** boxes are declared in HTML, otherwise the JS code will not recognise the assoc
- It should also be placed before the buttons are declared in HTML. Otherwise when the JS code is executed, the system will not recognise the TopBar box.

2.3. Implementation

1. Use this code to generate a random number in the range of 1 – 4:
   `Math.floor((Math.random() * 4) + 1)`
2. Name the button IDs as numbers: 1, 2, 3, ….., 0.
3. Use `this.id` to pass the object ID from HTML code to JS code.

   In HTML: `<button onclick="checkAnswer(this.id)" id="1"……`

   In JS: `function checkAnswer(buttonID) { …`
4. Use this code to convert integer to string: `result.toString()`
5. Give the **TopBar** box an ID called **message**, because objects can only be associated by ID inside JS code.
6. To print a message in a `<div class="top_bar" id="message">` box, use this code:
   `document.getElementById("message").innerHTML = "Good!";`
7. If you wish to specify font colours for the above code, add a `<span>` to the message:
   `document.getElementById("message").innerHTML = "<span style='color: green;'>Good!</span>";`
8. Strings can be added directly with the + operator: `"Hello" + "World"`

2.4. Submission

Please submit your work on QMPlus/EBU6305/Assessment/**Lab 2**.

Please note – if you are not attending the lab in person, your work will NOT be assessed, and you will receive zero mark for your submission.

Make sure all related files (including images) are within one project folder. Zip the project folder as one file to submit on QMPlus.

The following marking criteria will be used. QMPlus marking rubrics can only use integer points system. The equivalence of points and marks are as shown below.

Attempt in the lab	0.5 mark	1 point
Random Quiz	1 mark	2 points
Checking Answers	1 mark	2 points
Total	2.5 marks	5 points