IT TECHNICAL TRAINING

# GET INTO TECH PROGRAMME

EXERCISE GUIDE

V3.0

Visit your learning portal:

QA.COM/myQA

## Contents

## Exercise 3 – Key Internet Technologies

### Objective

To understand the role and purpose of internet technologies.

### Task

In a small group, work through the following list between you and write down the purpose of the following pieces of technology, in particular

- Whether they are relevant to clients (web browsers)
- Servers
- Or both

Cover at least one item from every set and search the web for any you don't know.

The purpose of this exercise is to introduce you to some terms and technologies that will crop up on the course. You are not expected to have perfect answers just yet!

Apache, MySQL, nGinX, IIS

HTML, CSS, Javascript, PHP

Chrome, Firefox, Microsoft Edge, Lynx

DNS, HTTP, TCP, HTTP Request Methods

# Exercise 4 – PHP Introduction

## Objective

- To run PHP files within NetBeans
- To run the XAMPP Apache webserver
- To understand how PHP and HTML may be combined in a file
- To be comfortable modifying code and rerunning to monitor changes
- To understand the two different types of PHP project

## Tasks

### Task A: Create a NetBeans Project

1. Create a new NetBeans project called **Exercise4.**
2. Copy the following files from your labfiles source location into the **Exercise4** project folder under **htdocs.**

- BuildingHTML.php
- MixedWithHTML.php
- PurePHP.php

3. Run each file and view the outcome within Chrome. Remember to start your Apache web server.
4. For each file, view the source (in Chrome use keyboard shortcut CTRL+U).
5. In NetBeans, look at the code and compare it to the view source and rendered output.

You may not understand the syntax just yet, however, see if you can work out how the HTML outputted from the webserver relates to the code in the PHP files. Have a go at identifying the differences between the styles of mixing static content (text, html) and PHP.

## Task B: Modify PHP

Modify the $title, $heading and $content variables on each page and rerun them. Change only the text between the quotes. For example, change ‘My Web Page’ to ‘An Example!’

Run the files to view them in your browser to see your changes.

## Task C: Change the Run Configuration

Change the **run configuration** of the project to run as a **Script (run in command line)** application and run each file within NetBeans.

Note where the output is rendered.

Compare how the **PurePHP.php** file renders differently as a command line application.

Can you work out what **PHP_EOL** and **“\t”** are doing?

## Note

You would not normally mix files from a command line application with files for a website. You would create two separate projects for this scenario.

# Exercise 5 – HTML

## Objective

- To create HTML pages within NetBeans
- To use basic tags such as <h1> and <p> tags
- To practise with hyperlinks, images, lists and tables
- To use the HTML5 structural tags
- To annotate your page with HTML comments
- To use character entities

## Tasks

### Task A: Create a NetBeans Project

1. Create a new NetBeans project of type PHP called **AboutMe**.
2. Ensure you create this project under your **htdocs** folder.
3. Your project will include an **index.php** file. Ensure this page is an HTML5 page and include a title of ‘About Me’ in the <head> section.
4. Use the HTML5 structural tags (section, header, article, footer) plus heading and paragraph tags as appropriate, and write a small piece of content about yourself.
5. Add a folder to the project called **images**. Put some image files into this folder and include at least one image in the index.html page.
6. Test the file as you work.

You have not styled your page so it may not be aesthetically pleasing. You will style it in a later exercise. Right now, it is the content and its semantic meaning that is important.

### Task B: Create a HTML Table

1. Add a second page to your project called **Favourites.php**
2. Create an HTML **table** in this page and list your top five favourite things, such as your favourite book, band or food. Add at least one **link** to an external resource (e.g. to a Wikipedia page for one of your favourite items). Investigate how you can get this page to open in a separate browser tab.
3. Add a **footer** to this page and add some **copyright** information to it.
4. Put some **comments** in your HTML to explain what the table tags mean.

### Task C: Create navigation links

Use a **nav** tag combined with an unordered **list** and **anchor** tags to create a navigation section to link your two pages together.

# Exercise 6A – CSS

## Objective

- To style HTML pages with CSS
- To use an external style sheet plus internal and inline styles

## Tasks

### Task A: Create a NetBeans Project

1. Create a new NetBeans project of type PHP called **AboutLisa**.
2. Ensure you create this project under your **htdocs** folder.
3. In File Explorer / Finder, copy the starter file **Ex6A_Lisa_Starter.php** and the **images** folder into your **AboutLisa** project folder under htdocs.
4. Run the **Ex6A_Lisa_Starter.php** file and familiarise yourself with how it looks.
5. In NetBeans, add a Cascading Style Sheet file to your project called **styles.css**.
6. Specify the following style rule within your file:

```css
body {
    font: verdana, sans-serif;
    font-size: 20px;
    background-color: lightgrey;
    padding: 20px;
    color: navy;
    text-align: center;
}
```

7. Add a link to the <head> section of your HTML page to your styles.css file. **Tip**: drag the file from the Projects pane into the <head> section of the document.
8. Save all of your files and run your web page to test.

## Task B: Create more style rules

1. To improve the **navigation** section of the page, create a rule in **styles.css** to left align the text:

```
nav {text-align: left;}
```

2. To emphasize the **footer**, create a style rule that contains the following:

```
background-color: dimgray;
          color: white;
          font-size: 16px;
```

3. Save your files and refresh the page in your browser to test.

## Task C: Create internal and inline styles

1. You want the <h3> heading of "Who Am I?" to have its own specific style. Add an **id** attribute to the heading with a value of **MainTitle**.
2. Create an internal style block within the <head> section of the page. Create a style rule for the tag with an id of MainTitle that contains the following:

```
color: maroon;
line-height: 2;
font-size: 30px;
```

3. You want to make the building images on this page smaller than the viewport width so create an internal style rule for all <**img**> tags. This rule should specify that images are block-level display elements (they are normally inline), plus automatic margins so the browser balances the margins either side and finally, a width of 60%:

```
display: block;
margin: auto;
width: 60%;
```

4. Save your file and refresh your web page in the browser. The building images should look good, however, the image of Lisa is now too big.
5. As you will have some images that need a smaller percentage size you decide to create a style class rule called **smallimg**. Create this style rule and set the **width** property to **20%.**
6. Add the **class** attribute to the Lisa image and set its value to **smallimg**.
7. Save your files and refresh the page to test.

## Task D: Add CSS to your AboutMe Project

Practice creating styles for the pages in your **AboutMe** project.

# Exercise 6B – CSS

## Objective

- To style HTML pages with Bootstrap
- To import a Google Font
- To understand responsive web design

## Tasks

### Task A: Style a page using Bootstrap

1. Open your NetBeans project: **AboutMe.**
2. Add all the necessary declarations, meta tags and links for Bootstrap to your **Favourites** web page.
3. Research how to style a table using Bootstrap and implement what you learn into your favourites table. **Tip**: Don't forget the container.

### Task B: Import a Font

1. Browse Google Fonts and choose a font for your Favourites page.
2. Create an external style sheet and import the font.
3. Use the new font within your style rules e.g. the body{…} style rule.
4. Link to the style sheet from your web page.

### Task C: Examine a responsive application

1. In File Explorer / Finder add **Ex6B_Lisa_Bootstrap.php** and **stylesA.css** to your **AboutLisa** project folder.
2. Browse to the file in **Chrome** to test.
3. Open the Chrome Developer tools (**F12**).
4. Toggle the Device Bar (**Ctrl + Shift + M**). Adjust the size of the screen to see the effects on the layout of the web page. Pay particular attention to the navigation menu at the top and how the images are stacked or horizontal.
5. In NetBeans, explore the markup and Bootstrap classes that have been applied to the document. Notice the different **containers** in the document. Investigate the **navigation** section and notice there are two versions of the navigation area: one for when the viewport is wide enough to show the menu as text and another 'hamburger' button. Also investigate the **search** button and how it was styled with an **icon** plus the **rounded corners** of the image.
6. Edit **stylesA.css** to customise the page to your own individual style. If possible, use a colour palette chooser to help you find complimentary colours.

# Exercise 7 – PHP Syntax

## Objectives

- To declare and output variables
- To use different types of operators
- To become familiar with the PHP manual

## Tasks

### Part 1: Practice declaring variables

1. Create a new NetBeans project: **Exercise7**.
2. In Finder / Windows Explorer, copy the **index.php** file from the labfiles source folder to your project folder under htdocs and overwrite the file.
3. Locate the comment **`// Put the Part 1 PHP code here`**.
4. Declare a variable with the name **`$msg1`** and assign it the string value "**`Hello World<br/>`**".
5. Output this to the page with an echo statement.
6. Save your web page and run in a browser to test.

   "Hello World" should be displayed under the horizontal rule **`<hr>`** for Part.
7. Continuing in the same block of PHP code, declare variables called **`$username`** and **`$email`**. Assign them the values of "**`Student`**" and "**`student@email.com`**".
8. The purpose of the next three lines of code is to demonstrate the difference between using single and double quotes when using variables in strings. Add the following echo statements:

```
echo "$username can be contacted at $email<br/>";
echo '$username can be contacted at $email<br/>';
echo $username . ' can be contacted at ' . $email . "<br/>";
```

9. Save the page and refresh it in your browser.

   Notice that the second statement outputs as-is (barring the HTML line break tag **`<br/>`** which is rendered as such). The single quotes remove the ability to detect PHP variable names within a string. Concatenation in the third line shows how this can be overcome and that a combination of single and double quotes can also be used. Continuing in the same block of PHP code:
10. Declare variables called **`$msg2`** and **`$msg3`**.
11. Assign them the values "**`Print is similar to echo`**" and "**`Print only takes one argument`**".

12. The purpose of the next four lines of code is to demonstrate the differences between the echo and print statements. Add the following code:

```
print "<h3>$msg2</h3>";
print "<h3>$msg3</h3>";
echo $msg1, " ", $msg2, "<br/>";
print $msg1, " ", $msg2, "<br/>";
```

This code will give you your first error. The **echo** statement uses a set of string arguments, separated by a comma, however, the **print** statement accepts only a single argument. Therefore, the PHP Interpreter throws a Parse error (note the line number is included in the error message).

Go back to the code and comment out the offending **print** statement using **//**. Save the code and refresh the page to test.

13. The final section of Part 1 looks at outputting numeric values and booleans:

a) Declare four variables, **$w**, **$x**, **$y** and **$z**.

b) Assign these the values **7**, **5**, **true** and **false** respectively. Do not put quotes around **true** and **false** as they are booleans not strings.

c) The **echo** and **print** statements will output numbers as strings *without* the need to change (cast) them into a string.

d) Enter the following lines of code and then save your file. Refresh your page to see the results:

```
echo $w . " - the number is treated as text<br/>";
print $x . " - the number is treated as text<br/>";
```

e) Booleans are treated a little differently. Enter the following two lines of code directly beneath the last, saving and refreshing to see the results:

```
echo $y . " – representing true<br/>";
print $z . " – representing false<br/>";
```

f) Notice that the value of **$z** (which is false) is not outputted at all, whilst the value of **$y** (which is true) is outputted as 1. String conversion behind the scenes makes a Boolean **false** an empty string. Fix this by forcing an explicit cast to an integer by putting **(int)** before the **$z.** This ensures that an integer is returned.

14. Finally, see how arithmetic can be performed on numbers and the result returned by both **echo** and **print** statements. Add the following code, save and refresh the page to test.

```
echo ($w + $x) . "<br/>";
print ($w + $x) . "<br/>";
```

## Part 2: Investigating Types

You have seen that numbers and booleans can be automatically outputted as strings. This type conversion that happens in the background is called implicit casting. You will investigate to see if the new data type is preserved after the conversion.

1. In the PHP manual (PHP.net) search for help on the **gettype** function.
2. Within the PHP code block, under the comment **`// Put the Part 2 PHP code here`**, enter seven lines of code to:
    - Output the types of the seven previously defined variables (use **`echo`** and **`gettype($varName)`**)
    - Save the code and refresh the page to test
    - Has the data type of the variables been changed?

## Part 3: Operators

The final part of this exercise looks at operators and expressions.

1. Mathematical operators can be used to add, subtract, multiply, divide and find the modulus of variables.
    - In the following line of code, any of the symbols can replace the addition. Add this line of code within the section of PHP code under the comment **`// Put the Part 3 PHP code here`** before saving the web page and refreshing your browser.

```
echo $w + $x . "<br/>";
```

  - PHP performs the calculation, returns the result, and then converts it implicitly into a string for display.

2. But what happens if you ask PHP to output an expression? Enter the following lines of code to investigate. Save and refresh to test.

```
echo ($w < $x) . "<br/>";
echo ($w > $x) . "<br/>";
echo ($w > $x == $y) . "<br/>";
echo ($w < $x == $y) . "<br/>";
```

   a) Here the PHP Interpreter evaluates the expression, returns the boolean result and then casts this into a string for output. Ordinarily you would use these comparison expressions to execute blocks of code conditionally rather than output the expressions via an echo statement.

   b) Ensure you understand the output from the above code.

3. You can also use logical operators (AND &&, OR ||, NOT !) in expressions. Enter the following code and examine the results. Save the file and refresh the browser to test.

```
echo ($w === 7 && $y == true) . "<br/>";
echo ($w == 8 || $z == false) . "<br/>";
```

4. Enter the following five lines of code. They demonstrate how the pre and post increment operators work. The same logic applies to the decrement operator also. Try to work out what the output will be before you refresh the page.

```
echo $x . ' - the current value of $x<br/>';
echo $x++ . ' - $x is returned and then incremented<br/>';
echo $x . ' - the new value of $x<br/>';
echo ++$x . ' - $x is incremented and then returned<br/>';
echo $x . ' - the new value of $x<br/>';
```

5. The assignment operators can be used to perform a mathematical calculation and then assign the result to the variable. This is a compound operator. In this example, numbers or other numeric variables could replace the variable used. Enter the following three lines of code to investigate this, saving and refreshing to test:

```
echo $w . ' - the current value of $w<br/>';
$w += $x;
echo $w . ' - the result of adding $x to $w';
```

6. Try substituting other mathematical operators in place of the addition(+) operator.

# Exercise 8 – Create a Calculator

## Objectives

- To obtain user input from the keyboard
- To use die() as a basic error handling mechanism
- To create a calculator application using conditional statements

## Background

PHP provides the stream_get_line() function which reads input from a "stream" (a generic name for a file-like resource).

There are several predefined streams, including STDOUT (where output goes; typically to the terminal or output window) and STDIN (where input comes from; typically, the keyboard).

By using stream_get_line() with STDIN we can create a prompt for a user to enter some input.

### Task A: Experiment with stream_get_line()

Create a new NetBeans project called Exercise8. Ensure it is a Script (run in command line) project type. Add the following code to the index.php file:

```
<?php
echo "Prompt?\n";
$result = stream_get_line(STDIN, 100, "\n");
echo $result;
?>
```

Run the project. In the Output window after the Prompt type "hello world" and hit the Enter key.

You should see the code reads in the text you entered and places it in the $result variable. The code then echoes the value of the variable to the output window.

Try changing the initial echo as well as entering more text at the prompt. What does 100 represent?

### Task B: die()

The most basic kind of error handling in PHP is the die() statement.

die() takes an argument which it echoes before stopping the execution of the script.

Try adding a die() statement to your script.

```php
<?php
echo "Prompt?\n";
die('TEST');
$result = stream_get_line(STDIN, 100, "\n");
echo $result;
?>
```

Run the script. You should see the die() statement emits the text passed as the parameter and then immediately halts the execution of the script.

Change the text of 'TEST' to read: 'There is an error. The application will shut down'.

Run the project to see the results.

## Task C: Create a Calculator

The task is to create a simple calculator application.

It should prompt the user for three inputs:

1. An operation (one of + * - /).
2. A first number.
3. A second number.

You should include some basic error checking:

- if the operation is / (i.e. division)
- AND the second argument is 0
- die() with an error

Then use a switch-case statement to decide which operation to perform. That is, switch on the operation you get from the user.

## Note:

The die() statement should not appear in production code since its solution to an error is to halt the script there and then. This might be useful during development, but should not appear in production code. There are a lot of tutorials online which use die(). You should see it as a placeholder for a more sensible mechanism. You will learn more about error handling later in the course.

# Exercise 9 – Arrays and Loops

## Objective

- To practise working with sequential arrays
- To practise working with associative arrays
- To practise iterating over arrays

### Task A: Experiment with sequential arrays

Create a new NetBeans command line project called Exercise9. Add the following code to index.php:

```php
<?php
echo 'What is your favourite colour? ';
//Add code here to create an array called $favourites
//Add code to read the response and store your favourite colour in the first position of the favourites array


echo 'What is your favourite food? ';
// Add code to read the response and store your favourite food in the second position of the favourites array


echo 'What is your favourite place? ';
// Add code to read the response and store your favourite place in the third position of the favourites array
//Add code to echo each element of the array individually


?>
```

Try creating the code detailed above. Run your script and check you see each of your favourite items.

Add code to display the items of the array by using firstly a for loop and then a foreach loop.

## Task B: Experiment with associative arrays

Add a file called Shopping.php to the project.

Create an associative array called $basket using the new square bracket [ ] array syntax.

Add the following items and their prices to your basket:

```
White Bread, 1.05
Brown Bread, 1.25
Milk, 1.00
Cheese, 2.75
```

Use a foreach loop to iterate over the items in your basket and print the item's name and price to the terminal.

## Task C: Experiment with multidimensional arrays

Add a file called MyFamily.php to the project.

Create a multidimensional array called $family that stores your family members' name, age and hair colour.

Fill the array with at least 3 family members and then choose a looping technique to display its contents in the output.

## Task D: Experiment with loops

Add a file called Loops.php to the project.

Write a loop that prints out pairs of numbers. The first number in the pair should be between 0 and 12 and the second number in the pair should display the first number squared.

```
0     0
1     1
2     4
3     9
4     16
…
12    144
```

# Exercise 10 – Functions

## Objective

- To practise creating functions
- To revisit conditional structures
- To use built-in functions

### Task A: Create a Game: Rock, Paper, Scissors!

You are going to create a command line version of the game: Rock…Paper…Scissors!

The user will play against the computer at this game. You should design a program that does the following:

- Prompts the user to enter a value: R, P or S
- The program should convert this value into Rock, Paper, or Scissors respectively
- Asks the computer to generate a random value between 0 and 2
- Convert the computer's choice. 0 becomes Rock; 1 becomes Paper; 2 becomes Scissors
- Compare the user's choice with the computer's choice to display a message indicating whether the user won, lost or drew against the computer
- Showcase what you have learned about conditional statements and create your own functions

### Note:

*Rock smashes Scissors, Paper wraps Rock, Scissors cut Paper!*

## Task B: Compare programs

Compare your solution to other class members'.

- Have you achieved this differently?
- Does anyone's solution stand out?
- How would you structure your code differently?
- Would you consider changing the names of your variables or functions?

# Exercise 11 – Functions, Namespaces & Includes

## Objective

- To structure your projects and practice creating functions
- To practice referencing other php files using include and require statements
- To create namespaces

### Task A: Create a doMaths Function

Create a new command line project called Exercise11

If one does not already exist, create an index.php file

Add a folder called library to the project

Within this folder add a further subfolder called maths

In the maths folder add a php file called mathsA.php

In mathsA.php create a function called doMathsA with the following signature:

```
function doMathsA($first, $second, $operator){
}
```

Write the logic to perform basic maths and return the answer using the supplied parameters.

In index.php, prompt the user 3 times to ask for their first, second and operator choices respectively and call the doMathsA function passing in these values. Be sure to reference the doMathsA file with an appropriate statement.

Print out the returned answer to the output window.

Test the application works as expected.

### Task B: Create a namespace

Add a new file to the library\maths folder called mathsB.php

Create a namespace at the top of this file called 'library\maths'

Copy the doMathsA function into this file and rename it to doMathsB

In index.php comment out the call to doMathA and reference the doMathsB function by including the file mathsB.php

Import the function using a ‘use function’ statement

Call doMathsB and ensure the application works as expected.

## Task C: Amend PHP’s Include Path

Use the set_include_path and get_include_path functions to instruct PHP to look in your library/maths folder to resolve any include / require statements.

Reference your mathsB.php file using only its filename and confirm the application still works as expected.

# Exercise 12 – Strings & Regular Expressions

## Objective

- To practice using string functions
- To practice using regular expressions

### Task A: Experiment with String Funtions

Create a new command line project called Exercise12.

Add a file called StringFunctions.php to the project.

Review the string function examples throughout chapter 12 and demonstrate their use within your project.

Also, ensure you demonstrate the difference between the printf and sprint functions.

### Task B: Experiment with Regular Expressions

Add a new file to your project called RegularExpressions.php

Review the regular expression examples throughout chapter 12 and demonstrate their use within this file.

Also, ensure you demonstrate the differences between the preg_split, explode and str_split functions.

# Exercise 13 – File IO

## Objective

- To practice reading and writing to text files
- To practice reading from XML files

### Task A: Read from a Text File

Create a new command line project called Exercise13.

Add a file called ReadTextFile.php to the project.

Using Finder / File Explorer add the people.txt file to the project folder.

Practice using fopen, fgets and fclose to read from the people.txt file.

Also, add code to read the file by "slurping".

### Task B: Write to a Text File

Add a file called WriteTextFile.php to the project.

Add code to this file to open the people.txt file for appending.

Next, define the following variables:

```
$dataName = "Marge Simpson";
$dataEmail = "marge@springfield.com";
$dataPhone = "555-332-221";
```

Practice using fwrite to write the variable values to the file as a new line separated by tabs. Don't forget to close the file when you have finished writing to it.

Run your file and confirm that a new row for Marge Simpson has been appended to the bottom of the people.txt file.

### Task C: Read from an XML File

Using Finder / File Explorer add the books.xml file to the project folder.

Practice using the simplexml_load_file function to load the file into memory. Use the object oriented syntax with the "skinny" arrow to access the child nodes of the document:

```
$xmlNodes->children()
```

Loop through the document’s nodes and print out each book’s details:

```
echo $books->title
```

# Exercise 14 – Array Functions

## Objective

- To practice sorting arrays
- To practice slicing and merging arrays
- To use callbacks and closures with array filters

### Task A: Sort Arrays

Create a new command line project called Exercise14.

Using Finder / File Explorer add the file SortingArraysExercise.php to the project.

Follow the TODO instructions in the file to practice sorting arrays in different orders.

### Task B: Slice and Merge Arrays

Using Finder / File Explorer add the file SliceMergeExercise.php to your project.

Review the code and run the file to test. Answer the following questions:

What does the explode function do?

How would you slice the string 'example.com/blog/post/1' to return just 'post/1'?

What is the difference between array_merge and array addition?

### Task C: Filter an Array and use Callback Functions

Using Finder / File Explorer add the file ArrayFilterExercise.php to your project.

Review the code and run the file to test. Answer the following questions:

What does the array_filter function do?

How does the $min captured variable used with the criteria_greater_than function give you more flexibility than the filterArray callback?

What is the purpose of the ($basket['Item1']) code within the if statement at the bottom of the file?

# Exercise 15 – HTTP SuperGlobals

## Objective

- To practice using HTTP Get
- To practice using HTTP Post
- To use the new HTML5 attributes

### Task A: Use HTTP Get

Create a new web project called Exercise15.

Using Finder / File Explorer add the file GetExercise.php to the project.

Review the code and browse the file. Append two query parameters to the end of the URL in the following format:

```
?name=Lisa&hobby=Playing the Saxophone
```

Confirm the name and hobby information is displayed in the webpage.

Amend the URL and change the values of the parameters but ensure the names of the parameters remain unchanged as name and hobby.

Try adding some additional parameters to the URL and test the outcome.

```
?name=Aesha&hobby=X-Box Gaming&food=pizza&pet=Rabbit
```

Notice that these parameters are displayed using a loop. Remove this loop and replace with hard-coded access to the above named parameters.

### Task B: Use HTTP Post

Using Finder / File Explorer add the file PostExercise.php to the project.

Review the code and browse the file. Notice that the page contains an html form that contains several form fields but only two fields are displayed on the page after the form has been submitted.

Edit the code to display all of the submitted values after postback.

## Task C: Use HTML Attributes and Bootstrap

Edit the PostExercise.php file to use some of the new HTML 5 attributes such as:

- Email
- Date
- Placeholder
- Required
- Autofocus

Test your page in at least two different browsers to compare their behaviour for the HTML 5 attributes.

Style the page with Bootstrap. For more information refer to Chapter 6: CSS, in the Delegate Guide.

# Exercise 16 – JavaScript & AJAX

## Objective

- To practice calling PHP functions from a web page
- To practice writing JavaScript
- To make an asynchronous request with AJAX
- To style a page with Bootstrap

### Task A: Create PHP Functions

Create a new **web** project called **Exercise16.**

Add a php file called **GameFunctions.php**

Create (or copy from an earlier project) any functions required to play a game of Rock, Paper, Scissors. This is the back-end logic and functionality of your game.

Use the **$_REQUEST** superglobal to read a parameter called **choice** from the incoming HTTP Request and assign this into a variable called **$userChoice**:

```
$userChoice = $_REQUEST["choice"];
```

Pass this $userChoice variable to your back-end functions as required to start a game.

Use the echo statement to pass the result of the game to the front-end.

### Task B: Use JavaScript and AJAX

Add an html file called **RPS.html**. This is the front-end logic and presentation layer of your game. Design this page with 3 images representing Rock, Paper and Scissors.

Each of these images should sit within an anchor tag that calls a JavaScript function called **Play** passing in either R, P or S depending on which image is clicked.

This function should have the following signature:

```
function play(strChoice) {…}
```

Inside this function use **AJAX** to make an asynchronous request to the GameFunctions.php script passing in a query parameter named choice with the value of strChoice that represents the R, P or S chosen by the game player:

```
'gamefunctions.php?choice=' + strChoice
```

Complete all the code (PHP, JavaScript / jQuery, HTML) necessary to successfully play Rock, Paper, Scissors from your web page.

Finally, add Bootstrap styling to your page.

# Exercise 17 – Upload & State

## Objective

- To practice uploading files to a web page
- To practice creating sessions

### Task A: Create an Upload Page

Create a new **web** project called **Exercise17**.

Add a php file called **UploadPage.php.**

Create a folder within your project called **Uploads**.

Using the module notes as a guide add the required HTML form and attributes to this page to enable you to upload a file to your site.

Write all the necessary PHP code to upload the chosen file to the Uploads folder.

You should include all necessary error checking including to allow the upload of only XML files and you should ensure a file is selected before attempting to upload to your site.

Use the provided sample files to test your site (**Categories.xml, Subcategories.xml** and **Products.xml**).

### Task B: Create a Session

In your **Exercise17** project add 3 new php web pages called **SessionPostPage1.php**, **SessionPostPage2.php** and **SessionPostPage3.php.**

At the top of **SessionPostPage1.php** add the code to start or retrieve a session:

```
session_start();
```

In the body section of the web page add the following form:

```
<form action="" method="post" >
  Username: <input type="text" name="username" />
  Password: <input type="password" name="password" />
  Colour: <input type="text" name="colour" />
  Animal: <input type="text" name="animal" />
  <input type="submit" value="Login" />
</form>
```

Underneath the form add some PHP code to set the session variables if the $_POST superglobal is not empty:

```
if(!empty($_POST)){
$_SESSION["username"] = $_POST['username'];
```

Add code for colour and animal but not password.

Add another conditional statement underneath the above code to display the user's name and favourite colour and animal if the $_SESSION superglobal contains values:

```
if(!empty($_SESSION)){
    echo "Welcome " . $_SESSION['username'] . '<br>';
    echo "You favourite colour is " . $_SESSION['color'] . '<br>';
    echo "You favourite animal is " . $_SESSION['animal'] . '<br>';
    echo "<a href='SessionPostPage2.php'>Go to Page 2</a><br>";
}
```

This code also includes a link to **SessionPostPage2.php** which enables you to test whether you can read the session variables.

At the top of **SessionPostPage2.php** add code to start or retrieve a session.

In the body of this page add a conditional statement to test whether the $_SESSION superglobal is not empty. Add the following code within the conditional block to read the session variables:

```
    echo "Hello " . $_SESSION['username'] . '<br>';
    echo "You favourite colour is " . $_SESSION['colour'] . '<br>';
    echo "You favourite animal is " . $_SESSION['animal'] . '<br>';
    echo "<a href='SessionPostPage3.php'>Logout</a><br>";
```

This code also includes a link to **SessionPostPage3.php** which acts as a logout page.

At the top of **SessionPostPage3.php** add code to start or retrieve a session, clear the session variables and then destroy the session.

In the body of this page add code to test if the $_SESSION superglobal is empty. If it is empty add code to display the message **"Welcome Guest".**

Add a link back to the Login page: **SessionPostPage1.php**

To continue your learning journey, visit:

QA.COM/TECHIT

company/qa-ltd

QALtd

QALtd

qatraining