

LAB MANUAL

Course: CSC336--- Web Engineering



Department of Computer Science

Learning Procedure

- 1) Stage **J** (Journey inside-out the concept)
- 2) Stage **a₁** (Apply the learned)
- 3) Stage **V** (Verify the accuracy)

COMSATS Institute of Information Technology (CIIT)
Islamabad

Table of Contents

| Lab # | Topics Covered | Page # |
|----------|--|--------|
| Lab # 01 | HTML Basics | |
| Lab # 02 | Working with HTML Links and tables | |
| Lab # 03 | HTML forms | |
| Lab # 04 | Css (Cascading Style Sheet) Basics | |
| Lab # 05 | Designing page lay-out using DIV,s and CSS | |
| Lab # 06 | Lab Sessional 1 | |
| Lab # 07 | JavaScript Basics | |
| Lab # 08 | JavaScript Timers | |
| Lab # 09 | JavaScript Form Validation | |
| Lab # 10 | PHP Basics | |
| Lab # 11 | Laravel Setup and Basics | |
| Lab # 12 | Lab Sessional 2 | |
| Lab # 13 | Laravel Blade Templating | |

| | | |
|----------------|-------------------------------------|--|
| Lab # 14 | Laravel CRUD Operations | |
| Lab # 15 | Laravel Migrations and Eloquent ORM | |
| | Terminal Examination | |

Statement Purpose:

To familiarize the students with

- HTML page structure
- Text formatting in HTML
- Lists in HTML
- Add images to web pages
- Use images as links
- Add video and audio files to webpages

Activity Outcomes:

After this lab the students should be able to understand HTML and its basic tags

- Students should be able to design basic web page using HTML Tags
- Student should be able to add text formatting tags
- Student should be able to add lists to web pages
- Student should be able to add images and videos to the web pages

1) Stage J (Journey)

Introduction

HTML

HyperText Markup Language (HTML) is the main markup language for displaying web pages and other information that can be displayed in a web browser. HTML is written in the form of HTML elements consisting of tags enclosed in angle brackets (like `<html>`), within the web page content. HTML tags most commonly come in pairs like `<h1>` and `</h1>`, although some tags, known as empty elements, are unpaired, for example ``. The first tag in a pair is the start tag, the second tag is the end tag (they are also called opening tags and closing tags). In between these tags web designers can add text, tags, comments and other types of text-based content.

The purpose of a web browser is to read HTML documents and compose them into visible or audible web pages. The browser does not display the HTML tags, but uses the tags to interpret the content of the page.

HTML Basic Structure

```
<html>
```

```
<head>
```

```
<title> Page Title Goes Here </title>
```

```
</head>
```

```
<body>
content goes here
</body>
</html>
```

Adding audio,image and video files:

```
<html>
<head>
<title>adding video</title>
</head>
<body>

<video src="abc.mp4">
</body>
</html>
```

How To Create basic web page

1. Open Notepad
2. Click on File -> Save as...
3. In the File name pull-down box, type in webpage.html
4. Click on Save
5. Type in content for your file
6. Once you finished the content, click on File -> Save

2) Stage a1 (apply)

Lab Activities:

Activity 1:

- Create basic page of COMSATS University as given below
- Add text about COMSATS and apply text formatting

Solution:

```
<html>
<head>
<title>Home</title>
</head>
<body bgcolor="#98E0F0">
<h1><font color="#1322D6"> COMSATS Institute of Information Technology </font>
</h1>
<hr width="100%" color="#030303" size="4" />

<Center><p><h2><b><pre> Home Department Admissions Academics
Exams</pre></b></h2> </p> </center>
<hr width="100%" color="#030303" size="4" />
<br />
```

```

<p> <h2><font color="#1322D6"> Historic Perspective:</font> </h2> </p>

<a href="lol.html"> click here </a><!-- Write your comments here -->

<p>WWF's goal is to: <q>Build a future where people live in harmony with
nature.</q></p>

</body>
</html>

```

Activity 2:

Add list of topics, images and videos to your website

Solution:

```

<html>
<head>
<title>Home</title>
</head>
<body bgcolor="#98E0F0">
<h1><font color="#1322D6"> COMSATS Institute of Information Technology </font>
</h1>
<hr width="100%" color="#030303" size="4" />
<Center><p><h2><b><pre> Home    Department    Admissions    Academics
Exams</pre></b></h2> </p> </center>
<hr width="100%" color="#030303" size="4" />
<ol type="I" start="4">
<li> computer </li>
<li> mouse </li>
<li> keyboard </li>
</ol>
<br />
<dl>
<dt>Coffee</dt>
<dd>- black hot drink</dd>
<dt>Milk</dt>

```

```
<dd>- white cold drink</dd>
</dl>

<iframe width="420" height="315"
src="https://www.youtube.com/embed/XGSy3_Czz8k?autoplay=1">
</iframe>
</body> </html>
```

3) Stage **v** (verify)

Home Activities:

Activity 2:

Learn and try different tags and formatting options on your webpage of Comsats.

4) Stage **a2** (assess)

Assignment:

Create a webpage for Comsats library. Add lists and apply text formatting to your page. Make videos and take images of the library and then add them to your page.

Statement Purpose:

To familiarize the students with

- Internal links
- External links
- In-page references
- use of tables in a web page

Activity Outcomes:

After this lab the students should be able to add linking information and tables in web pages.

1) Stage J (Journey)

Introduction

The crux of HTML is its capability to reference countless other pieces of information easily on the internet. When you link to another page in your own web site, the link is known as an internal link. When you link to a different site, it is known as an external link. Similarly, we can link different section with in a page.

The element `<a>` is used to link another document. Anything between the opening `<a>` tag and the closing `` tag becomes part of the link that users can click in a browser. To link another page, href attribute of opening tag of `<a>` is used. The value of the href attribute is the name of the file you are linking to.

Internal Link

An internal link can be created as

```
<a href="page_path"> Text to click </a>
```

Example: ` Click here `

External Link

An external link can be created as

```
<a href="full web address of the page"> Text to click </a>
```

Example: ` Click here `

In-page reference:

Can be created in two steps

Step 1: Mark locations

```
<a name="LOCATION1">
```

Step 2: link


```
<a href="#LOCATION1">.....</a>
```

4. Example

Internal link:

```
<html>
<head>
<title> Internal Linking </title>
</head>
<body>
It is the first page. To go to the next page,
please
<a href="second.html"> click here</a>
</body>
</html>
```

External link:

```
<html>
<head>
<title> External Linking </title>
</head>
<body>
This is the home page. To go to the google page, please
<a href="https://www.google.com.pk"> click here</a>
</body>
</html>
```

HTML TABLES:

HTML tables are defined with the <table> tag. A table is divided into rows (with the <tr> tag), and each row is divided into data cells (with the <td> tag). td stands for "table data," and holds the content of a data cell. A <td> tag can contain text, links, images, lists, forms, other tables, etc. In tables, different attributes can also be used like table border, cell padding, cell spacing etc. cell spacing is the pixel width between the individual data cells in the TABLE. (The thickness of the lines making the TABLE grid). The default is zero. If the BORDER is set at 0, the cell spacing lines will be invisible. Cell padding is the pixel space between the cell contents and the cell border.

4. Examples

Simple Table:

```
<table border="1">
<tr>
<td>row 1, cell 1</td>
<td>row 1, cell 2</td>
</tr>
<tr>
<td>row 2, cell 1</td>
<td>row 2, cell 2</td>
</tr>
</table>
```

Table Border:

```
<table border="1">
<tr>
<td>Row 1, cell 1</td>
```

```
<td>Row 1, cell 2</td>
</tr>

</table>
```

Cell spacing

```
<table border="1" cellspacing="5">
<tr>
<td>some text</td>
<td>some text</td>
</tr><tr>
<td>some text</td>
<td>some text</td>
</tr>
</table>
```

Cell Padding

```
<table border="1" cellpadding="10">
<tr>
<td>some text</td>
<td>some text</td>
</tr><tr>
<td>some text</td>
<td>some text</td>
</tr>
</table>
```

2) Stage a1 (apply)

Lab Activities:

Activity 1:

Use the page you created in the first lab and add the following linking information

- Create links to home, departments, admission and exams pages of CIIT Islamabad campus

Solution:

```
<html>
<head>
<title>Home</title>
</head>
<body bgcolor="#98E0F0">
<h1><font color="#1322D6"> COMSATS Institute of Information Technology </font>
</h1>
<hr width="100%" color="#030303" size="4" />
```

```
<a href="home.html"> click here</a>
```

```
<a href="departments.html"> click here</a>
```

```
<a href="admissions.html"> click here</a>
<a href="exam page.html"> click here</a>
```

```
<Center><p><h2><b><pre> Home   Department   Admissions   Academics
  Exams</pre></b></h2> </p> </center>
<hr width="100%" color="#030303" size="4" />
<br />
```

```
<p> <h2><font color="#1322D6"> Historic Perspective:</font> </h2> </p>
```

```
<a href="lol.html"> click here </a><!-- Write your comments here -->
```

```
<p>WWF's goal is to: <q>Build a future where people live in harmony with
nature.</q></p>
```

```
</body>
</html>
```

3) Stage v (verify)

Home Activities:

Activity 2:

- Create in-page reference to the list created in the body and marks the locations accordingly
- Add tables to department and admission pages.

4) Stage a2 (assess)

Use tables to create make a page layout as given below

Statement Purpose:

To familiarize the students with the HTML forms and different input tags.

Activity Outcomes:

After this lab the students should be able to understand HTML input tags

1) Stage J (Journey)

Introduction

HTML forms are used to pass data to a server. A form can contain input elements like text fields, checkboxes, radio-buttons, submit buttons and more. A form can also contain select lists, textarea, fieldset, legend, and label elements. The `<form>` tag is used to create an HTML form.

`<form>`

input elements

`</form>`

The `<form>` tag

The `<form arguments> ... </form>` tag encloses form elements (and probably other elements as well). The arguments to form tell what to do with the user input

`action="url"` (required)

Specifies where to send the data when the Submit button is clicked

`method="get"` (default)

Form data is sent as a URL with `?form_data` info appended to the end

`method="post"`

Form data is sent in the body of the URL request

`target="target"` . Target tells where to open the page sent as a result of the request.

Input Tags

There are many input tags in the forms.

Text input

A text field:

```
<input type="text" name="textfield" value="with an initial value" />
```

A multi-line text field

```
<textarea name="textarea" cols="24" rows="2">Hello</textarea>
```

A password field:

```
<input type="password" name="textfield3" value="secret" />
```

Buttons

A submit button:

```
<input type="submit" name="Submit" value="Submit" />
```

A reset button:

```
<input type="reset" name="Submit2" value="Reset" />
```

A plain button:

```
<input type="button" name="Submit3" value="Push Me" />
```

Radio Buttons

Radio buttons:

```
<input type="radio" name="radiobutton" value="myValue1" /> male<br>
```

```
<input type="radio" name="radiobutton" value="myValue2" checked="checked" />female
```

Checkboxes

```
<input type="checkbox" name="checkbox" value="checkbox" checked="checked">
```

Drop-down menu

```
<select name="select">  
  <option value="red">red</option>  
  <option value="green">green</option>  
  <option value="BLUE">blue</option>  
</select>
```

HTML 5 forms elements:

HTML5 is the current version of HTML and still under development. HTML5 enhances HTML form not only by providing new attributes to existing elements and but also provide new elements which can be added to the HTML forms.

HTML5 attributes for existing elements

Required: make an input field must to fill

Pattern: used to validate user's input

Readonly: makes an element read-only

Disabled: is used to make an input field disabled

Autocomplete: adds autocomplete functionality to input fields

HTML5 new form elements:

Datalist: an input field with predefined autocomplete options

My favorite color: <input type="text" list="color">

```
<datalist id="color">
```

```
<option>red</option>
```

```
<option>black</option>
```

```
</datalist>
```

Email input field: checks user's input for valid email address

```
<input type="email" name="email">
```

Date input field: shows a calendar to choose a data

```
<input type="date" name="dob">
```

Color input field: displays color window to choose a color

```
<input type="color" name="favcolor">
```

Number input field: makes an input field to accept only numeric values

```
<input type="number">
```

2)Stage a1 (apply)

Lab Activities:

Activity 1:

- Make a form with name, gender (use radio buttons) ,password and submit form option.Use text box for input fields and buttons for submit option.

Solution:

```
<html>
<head>
<title>Get Identity</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>
<body>
<p><b>Who are you?</b></p>
<form method="post" action="">
  <p>Name:
    <input type="text" name="name">
  </p>
<p>Password:
  <input type="password" name="password">
</p>
<p>Gender:
  <label><input type="radio" name="gender" value="m" />Male</label>
  <label><input type="radio" name="gender" value="f" />Female</label>
</p>
<p>:
  <input type="submit" name="submit" value="Login" />
```

```

<input type="reset" name="reset" value="Cancel" />
</p>
</form>
</body>
</html>

```

3)Stage v (verify)

Home Activities:

Activity 2:

Write the HTML code for JOB application form of CIIT (find the complete job application form from CIIT web site)

The screenshot shows a web browser window with the following content:

- Browser title: NON_faculty_job_applicat...
- Address bar: file:///C:/Users/Administrator/Desktop/Web%20Engineering%206B-6C/Labs/Assignemnt%20Solutions/Lab_Assignment_1/assign.html
- Form Header:
 - COMSATS Logo
 - Non Faculty Job Application Form**
 - COMSATS Institute of Information Technology
 - Affix a recent photograph 3cm x 5cm
- Form Fields:
 1. **Post applied for**
(mention only one position) [Text Input]
 2. **Department**
(mention department, if required) [Text Input]
 3. **Campus**
(Please tick only one choice) Sahiwal Attock
 4. **Name in full**
(Use CAPITAL LETTERS) [Text Input]
 5. **Father Name**
(Use CAPITAL LETTERS) [Text Input]

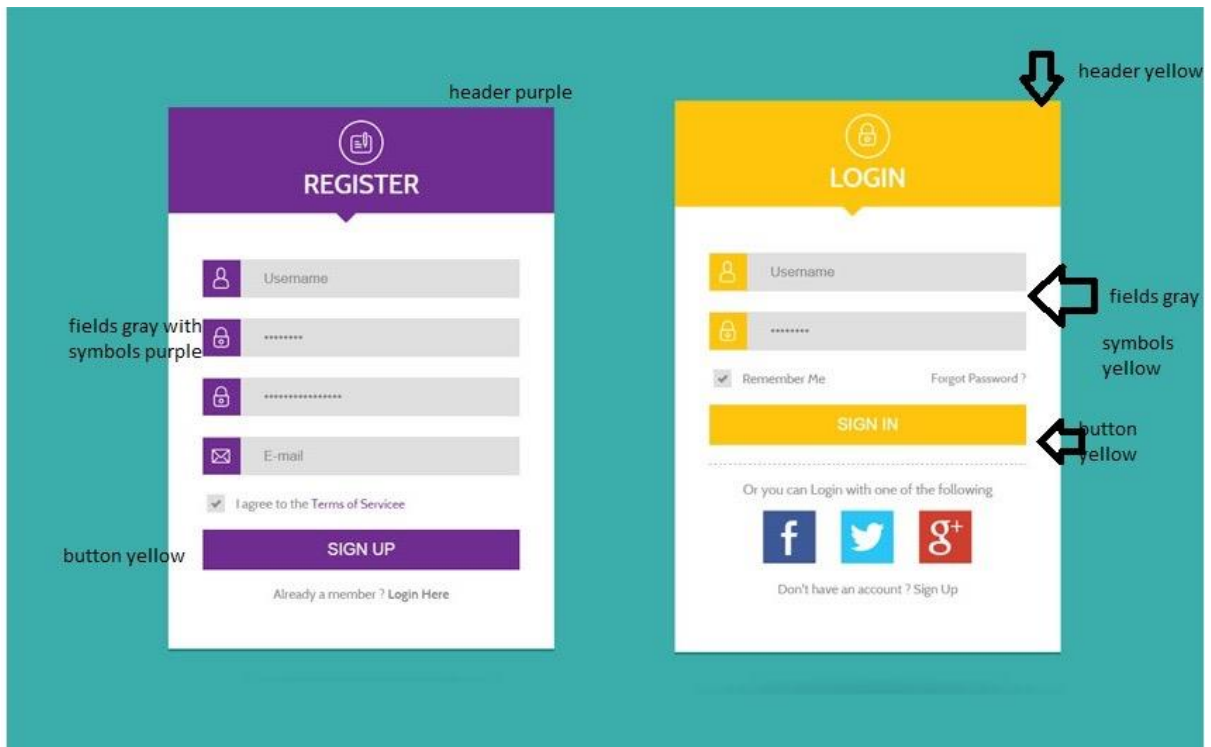
4)Stage a2 (assess)

Assignment 1:

Design the web template given below using Tables and forms concept in HTML.

Moreover add colors to the fields and tables as mentioned below.

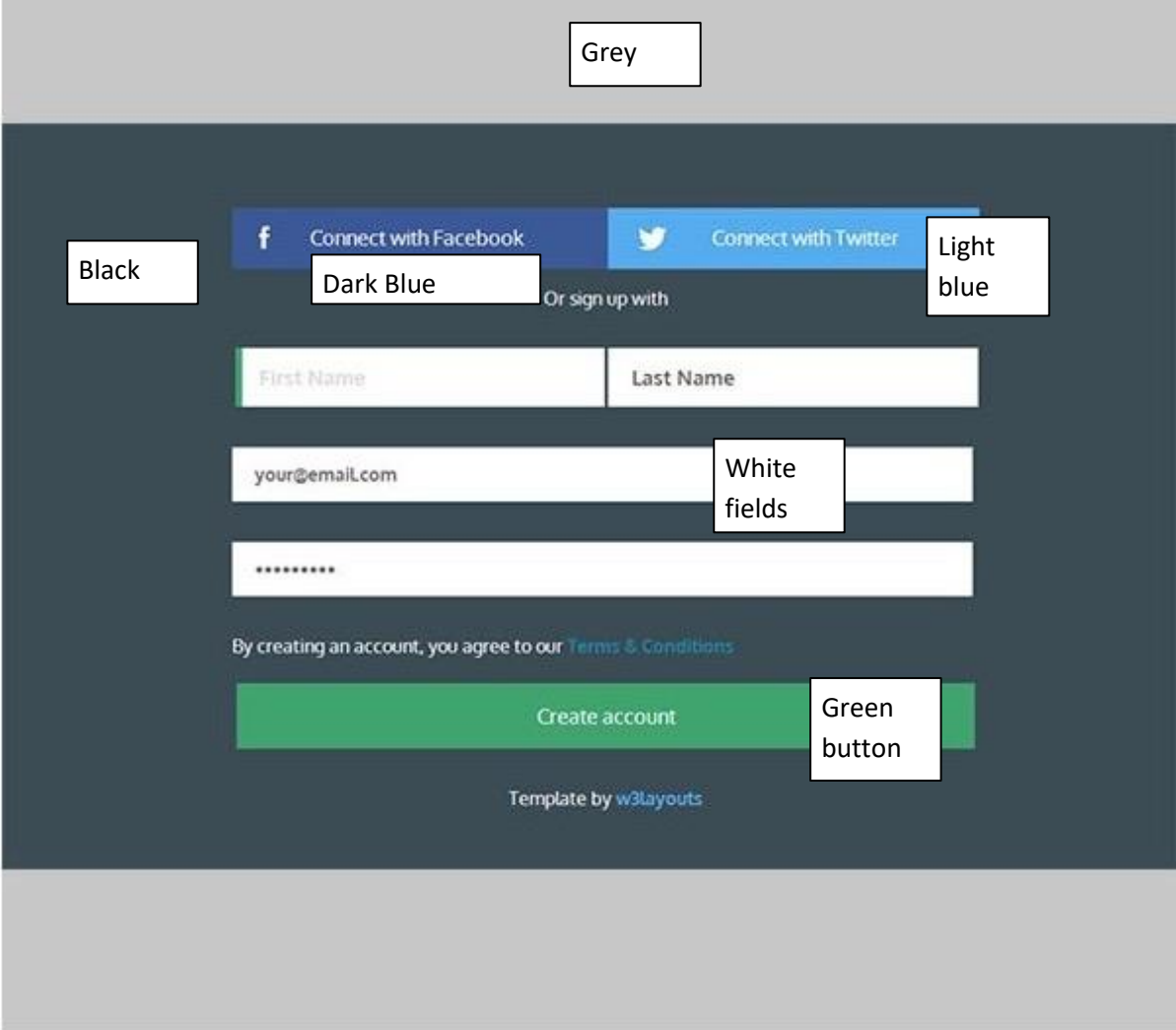
Add links to forgot password, signup, login here like key words



Assignment 2:

Design the web template given below using Tables and forms concept in HTML.

Moreover add colors to the fields and tables as mentioned below. Add links to connect face book, twitter, and terms & conditions



Statement Purpose:

To familiarize the students with the CSS.

Activity Outcomes:

After this lab the students should be able to apply CSS to the web pages.

1)Stage J (Journey)

Introduction

CSS(Cascading Style Sheet)

CSS stands for Cascading Style Sheets. It is a way to style HTML webpages. A style sheet is the presentation of an HTML document. CSS is a style language that defines layout of HTML documents. For example, CSS covers fonts, colours, margins, lines, height, width, background images, advanced positions and many other things. CSS can be written either inside the html tags (i.e. Inline), in the <head></head> section of an HTML document (i.e. Internal) or in a separate CSS File (i.e. External).

CSS Basic Structure

For inline styling CSS is written inside the style attribute of a tag. For example,

```
<tag style="property:value"></tag>
```

For Internal styling, CSS is written into the head section of the HTML document. For example,

```
<style>
```

```
.class
```

```
{
```

```
property:value;
```

```
}
```

```
</style>
```

For external styling, CSS is written as a separate file and the file is saved with an extension **.css**. In external style sheets the style tags are not necessary. Whereas, external CSS follow the same structure as the Internal CSS does.

Basic CSS Properties

Font Properties

Font Family

Font Style

Font Variant

Font Weight

Font Size

Font

Color and Background Properties

Color

Background Color

Background Image

Background Repeat

Background Position

Text Properties

Word Spacing

Letter Spacing

Text Decoration

Vertical Alignment

Line Height

Box Properties

Margin

Padding

Border Width

Border Color

Border Style

Width

Height

Float

4. Examples

Inline CSS:

```
<table style="width:200px; background-color: #000000;"> Table element with Inline Style</table>
```

Internal CSS:

```
<html>  
<head>
```

```
<title>Internal CSS</title>
<style>
h1 {color:#FF0000;
font-family:Calibri;
font-size:36px
}
</style>
</head>
<body>
<h1>This heading is styled with CSS</h1>
</body>
</html>
```

External CSS:

CSS file: (mystyle.css)

```
h1 {color:green;

font-size:36px;

font-family:calibri

}
```

2)Stage a1 (apply)

Lab Activities:

Activity 1:

Use the html page created in Lab 2 (page for COMSATS) and create an external style-sheet which style the elements of the page including style for

- apply font and text properties
- control the background color with CSS
- style different states of inks
- paragraphs and headings

Solution:

```
body {
background-color: lightblue;
}
h1 {
color: navy;
margin-left: 20px;
font-family: verdana;
```

```

font-size: 50px;
}
#para1 {
text-align: center;
color: red;
}
.center {
text-align: center;
color: yellow; }

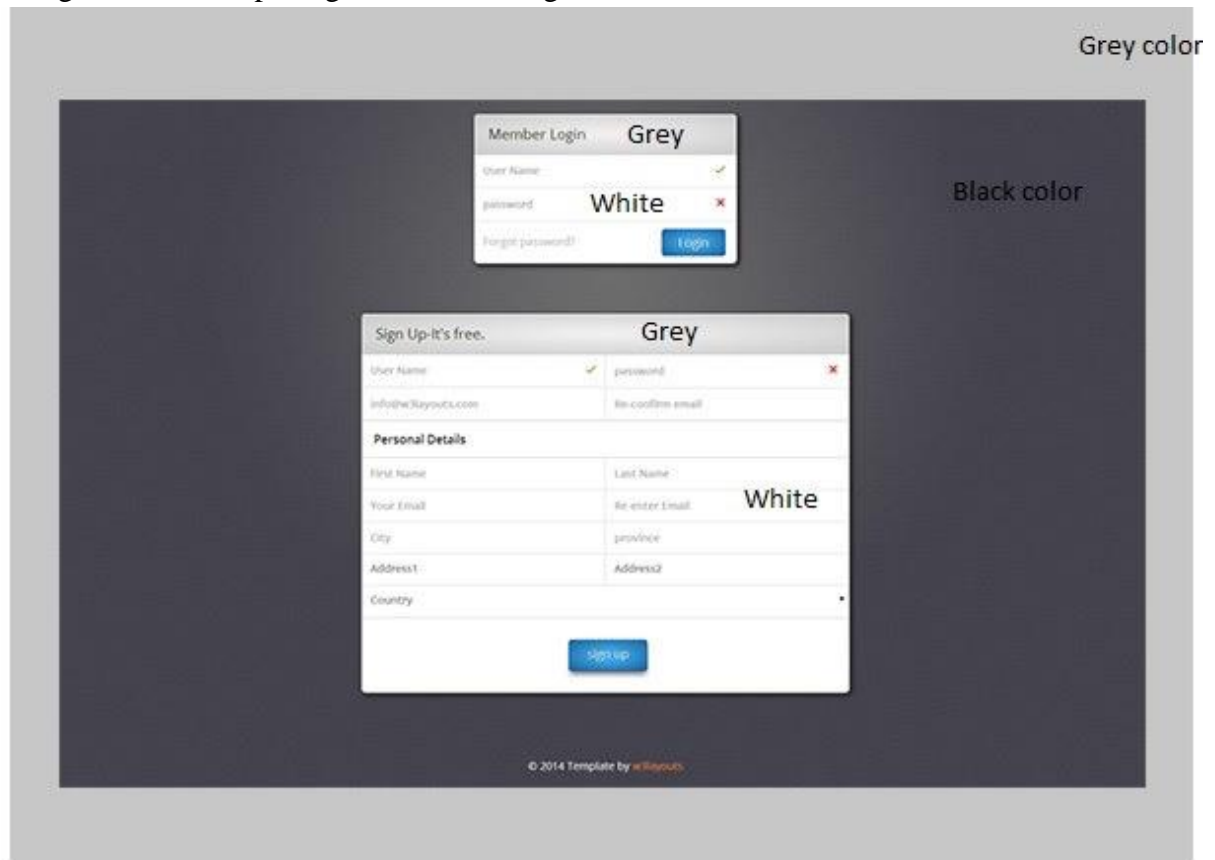
```

3)Stage ▼ (verify)

Home Activities:

Activity 2:

Design the web template given below using tables and external CSS



4)Stage a2 (assess)

Assignment:

Design the web template given below using tables and external CSS

The image shows a web form template. It features a grey header and footer. The main content area is white. On the left, there is a blue sidebar with white text. The sidebar contains the text 'Blue color white text' and 'Contact Now'. Below this, there are four input fields: 'Username', 'Password', 'Department', and 'Message'. At the bottom of the sidebar is a 'Send Message' button. To the right of the sidebar is a white form area. At the top of this area are four social media icons: Facebook, Twitter, LinkedIn, and a generic icon. Below the icons are three input fields: a text field with a user icon, a text field with an envelope icon, and a dropdown menu labeled 'Select a Department'. At the bottom of the form area is a large text area for the message.

Statement Purpose:

To familiarize the students with the use of Div's and the use of CSS for designing page layout

Activity Outcomes:

After this lab the students should be able to use the DIV tag and the use of CSS for page layout designing

1)Stage J (Journey)

Introduction

The div tag is used to define a division or section in an HTML document. It visually, allows us to make containers that can be formatted. It can be declared as: `<div>.....</div>`. We use div and CSS to design a page layout. The div tag is used to represent sections in a page and CSS is used to style these sections. We can describe the process of designing a page as

- Determine the requirements of the site
- Group the required information
- Make a site map
- Identify key elements for each page
- Decide about the arrangement of information on each page
- Translate the design into code

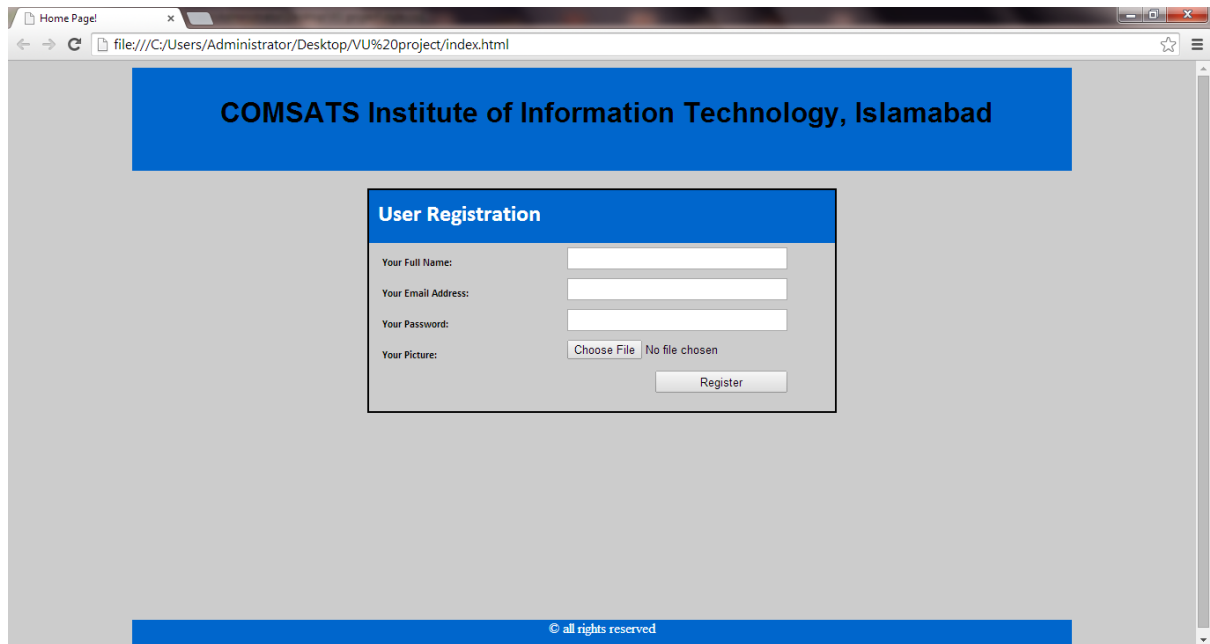
2)Stage a1 (apply)

Lab Activities:

Activity 1:

Use div and CSS properties to design a page layout which contains

- a header section to display the title, style this title with CSS
- a form container which contains a registration form, use CSS to style the elements of the form
- a footer section



Solution:

External CSS

```
body {
    background-color: lightblue;
}
h1 {
    color: navy;
    margin-left: 20px;
    font-family: verdana;
    font-size: 50px;
}
#para1 {
    text-align: center;
    color: red;
}
.center {
    text-align: center;
    color: yellow;
}
<html>
<head>
<title>Using divs</title>
</head>
<body>
```



```

<div>
<div style="width:100px;background-color:gray">First</div>
<div style="width:100px;background-color:red">second</div>
</div>
</body>
</html>

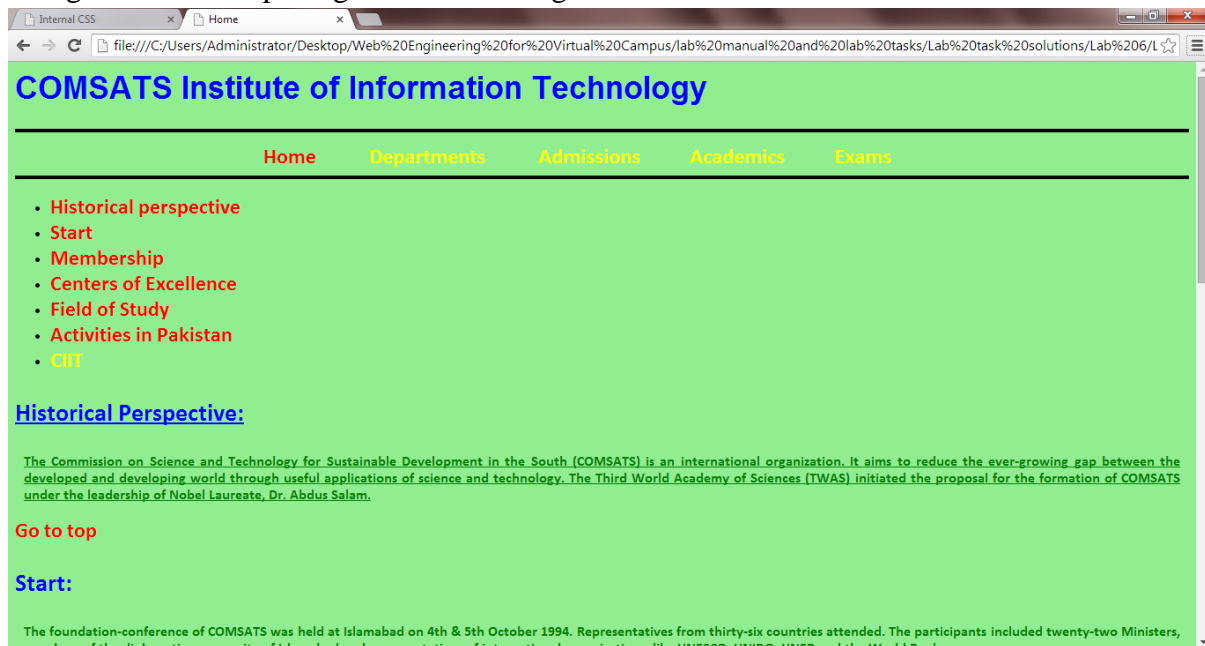
```

3) Stage v (verify)

Home Activities:

Activity 2:

Design the web template given below using tables and external CSS



4)Stage a2 (assess)

Assignment:

Design the web page for Comsats Alumni using External CSS.

Statement Purpose:

To familiarize the students with Java Script.

Activity Outcomes:

After this lab the students should be able to understand Java Script basics and to validate the form using Java Script.

1) Stage J (Journey)

Introduction

JavaScript is used in millions of Web pages to improve the design, validate forms, detect browsers, create cookies, and much more. JavaScript is the most popular scripting language on the internet, and works in all major browsers, such as Internet Explorer, Mozilla, Firefox, Netscape, Opera.

How to Put a JavaScript Into an HTML Page?

JavaScript can be embedded into HTML two ways

Inter script: `<script>`, `</script>` tags are used to start and end a javascript block

External script: JavaScript is written in a separate file and included in the HTML file using src attribute of the `<script>`. Example

```
<script src="myscripts.js"> </script>
```

Variables in JavaScript:

JavaScript provides numbers, string, boolean, and null variable types. JavaScript is a loosely typed language. var keyword is used to declare a variable. We can declare a variable in JavaScript as

```
var name='Asad';
```

Operats in JavaScript:

- Assignment Operator: =
- Arithmetic Operators: +, -, *, /, %, ++, --
- Logical Operators: &&, ||, !
- Comparison Operators: ==, ===, !=, !==, <, >, <=, >=

Input/output in JavaScript:

- write(); is used to write on browser
 - document.write("hello world")
 - document.write(a)
- prompt(); is used to take input from users
 - var num = prompt("Please Enter a Number", 0)
- alert(): used to show an alert box

Defining a function :

```
function functionName([parameters])
{
[statements]
}
```

Conditional statements:

If statement

```
if (condition)
    statement
```

or

```
if(condition)
    { statements }
```

If-else statement

```
if(condition)
    {statement}
else
    {statements }
```

Loops in JavaScript:

For loop

```
for(var i=1; i==10; i++)
{
Document.write("hello world")
}
```

While loop

```
While(condition)
{
}
```

Do-while loop

```
do
{
}
while(condition)
```

2) Stage a1 (apply)

Lab Activities:

Activity 1:

Input/output and variables:

```
<html>
<head>
<title>Untitled Document</title>
</head>
<body>
<script language="javascript">
var num=prompt("Pleae Enter a Number",0)
document.write("You Entered ",num)
</script>
</body>
</html>
```

Functions:

```
<html>
<head>
<title>Untitled Document</title>
<script language="javascript">
function getName()
{
var name=prompt("Pleae enter your name",'name')
document.write("Welcome Mr. ",name)
}
</script>
</head>
<body onload="getName()">
</body>
```

Conditional statement:

```
<html>
<head>
```

```

<title>Using If condition</title>
<script language="javascript">
function playGame()
{
var res=Math.round(Math.random()*10)
var num=prompt("Plea a number",0)
if(num==res)
document.write("You Won")
else
document.write("Your loss, correct Answer is"
,res)
}
</script>
</head>
<body>
<p onClick="playGame()">Play the Game</p>
</body>
</html>

```

3) Stage v (verify)

Home Activities:

Activity:

- Write the javascript code which asks the users to enter a number or zero to end taking the input. When user enters a 0 the program displays the sum.
- Write a javascript function which displays a question in prompt box and gets its answer. If the answer is correct then shows a success message otherwise displays a error message
- Write javascript code which asks users to enter a number, a message and displays that message for the number of times as entered by the user

4) Stage a2 (assess)

Assignment:

Use the contents of this lab in your project and present it before terminal exam

Statement Purpose:

To familiarize students with the concepts of JavaScript Animation using JavaScript Timers.

Activity Outcomes:

After this lab, the students should be able to understand the purpose and use of JavaScript Timers. Furthermore, they have to write timer functions in html for creating Website Carousel.

1) Stage J (Journey)

Introduction

A timer is a function that enables us to execute a function at a particular time.

Using timers, you can delay the execution of code so that it does not get done at the exact moment an event is triggered or the page is loaded. For example, you can use timers to change the advertisement banners on your website at regular intervals, or display a real-time clock, etc. There are two timer functions in JavaScript: `setTimeout()` and `setInterval()`.

setTimeout()

The `setTimeout()` method calls a function or evaluates an expression after a specified number of milliseconds. The function is only executed once. If you need to repeat execution, use the `setInterval()` method. We use the `clearTimeout()` method to prevent the function from running.

Syntax: `setTimeout(function, delay (in milliseconds))`

This function accepts two parameters: a *function*, which is the function to execute, and an optional *delay* parameter, which is the number of milliseconds representing the amount of time to wait before executing the function (1 second = 1000 milliseconds).

clearTimeout()

The `clearTimeout()` method clears a timer set with the `setTimeout()` method. The ID value returned by `setTimeout()` is used as the parameter for the `clearTimeout()` method.

```
id_of_settimeout = setTimeout("javascript function", milliseconds);
```

If the function has not already been executed, you will be able to stop the execution by calling the `clearTimeout()` method.

Syntax: `clearTimeout(id_of_settimeout)`

setInterval()

The `setInterval()` function executes a function or specified piece of code repeatedly at fixed time intervals. The `setInterval()` method will continue calling the function until the

`clearInterval()` is called, or the window is closed. The ID value returned by `setInterval()` is used as the parameter for the `clearInterval()` method.

Syntax: `setInterval(function, intervals(in milliseconds))`.

This function also accepts two parameters: a *function*, which is the function to execute, and *interval*, which is the number of milliseconds representing the amount of time to wait before executing the function (1 second = 1000 milliseconds).

clearInterval()

The `clearInterval()` method clears a timer set with the `setInterval()` method. The ID value returned by `setInterval()` is used as the parameter for the `clearInterval()` method.

```
id_of_setinterval = setInterval("javascript function", milliseconds);
```

Then you will be able to stop the execution by calling the `clearInterval()` method.

```
clearInterval(id_of_setinterval );
```

1) Stage **a1** (apply)

Lab Activities:

Activity 1:

In first activity of this lab, we will learn how to use `setTimeout()` method to display a text message after a delay of 3sec.

Solution:

```
<!DOCTYPE html>
<html>
<body>

<p>Click the button to wait 3 seconds, then Msg is displayed.</p>
<p id="demo"></p>

<button onclick="myFunction()">Try it</button>

<script>
var myVar;

function myFunction() {
  myVar = setTimeout(MsgFunc, 3000);
}

function MsgFunc() {
  document.getElementById("demo").innerHTML="This text is displayed after three
seconds";
}
</script>
</body>
```

Click the button to wait 3 seconds, then Msg is displayed.

This text is displayed after three seconds

Try it

Activity 2:

In second activity of this lab, we will learn how to use `clearTimeout()` method to stop the execution of a function before it is activated.

Solution:

```
<!DOCTYPE html>
<html>
<body>

<p>Click the button to wait 3 seconds, then Msg is displayed.</p>
<p id="demo"></p>

<button onclick="start()">Start</button>
<button onclick="stop()">Stop</button>

<script>
var myVar;

function start() {
  myVar = setTimeout(MsgFunc, 3000);
}

function stop() {
  clearTimeout(myVar)
}

function MsgFunc() {
  document.getElementById("demo").innerHTML="This text is displayed after three
seconds";
}
</script>
```

Click the button to wait 3 seconds, then Msg is displayed.

Start Stop

Activity 3:

In third activity of this lab, we will learn how to use setInterval() method for creating a clock which displays current time by updating it against each second.

Solution:

```
<!DOCTYPE html>
<html>
<body>

<p>A script on this page starts this clock:</p>
<p id="demo"></p>

<button onclick="myStopFunction()">Stop time</button>

<script>
var myVar = setInterval(myTimer, 1000);

function myTimer() {
  var d = new Date();
  var t = d.toLocaleTimeString();
  document.getElementById("demo").innerHTML = t;
}

function myStopFunction() {
  clearInterval(myVar);
}
</script>

</body>
</html>
```

A script on this page starts this clock:

12:39:28 PM

Stop time

Activity 4:

In fourth activity of this lab, we will learn how to use clearInterval() method for stop the clock at any particular moment within its updating.

Solution:

```
<!DOCTYPE html>
<html>
<body>

<p>A script on this page starts this clock:</p>
<p id="demo"></p>

<button onclick="myStopFunction()">Stop time</button>

<script>
var myVar = setInterval(myTimer, 1000);

function myTimer() {
  var d = new Date();
  var t = d.toLocaleTimeString();
  document.getElementById("demo").innerHTML = t;
}

function myStopFunction() {
  clearInterval(myVar);
}
</script>

</body>
</html>
```

A script on this page starts this clock:

12:39:28 PM

Stop time

1) Stage V (verify)

Home Activities:

Activity 1:

Create a Website Carousel using JavaScript for a responsive animal history web page with its layout for different screen sizes as provided below. Following are the supplementary details about different sections of the page:

- 1- The task should use bootstrap classes to create layout for Animal History banner, Image Slider and Animal description sections of the web page. Students can only use *bootstrap.min.css* as a third-party CSS library.
- 2- Students are supposed to write a JavaScript code for imageSlider with *Play* and *Pause* buttons. Slider should stop when Pause button is pressed and start when Play button is clicked.
- 3- It is pertinent to note that in descriptive sections of different animals, location of image and description changes with different screen sizes.

1. Extra-small, Small Screen

Animal History



play pause



Zebras

Zebras (/ˈzɛbrə/ zeb-rə or /ˈzi brə/ zee-brə)[1] are several species of African equids (horse family) united by their distinctive black and white striped coats. Their stripes

2. Medium Screen



play pause



Zebras

Zebras (/ˈzɛbrə/ zeb-rə or /ˈzi brə/ zee-brə)[1] are several species of African equids (horse family) united by their distinctive black and white striped coats. Their stripes come in different patterns, unique to each individual. They are generally social animals that live in small harems to large herds. Unlike their closest relatives, horses and donkeys, zebras have never been truly domesticated.

There are three species of zebras: the plains zebra, the Grévy's zebra and the mountain zebra. The plains zebra and the mountain zebra belong to the subgenus *Hippotigris*, but Grévy's zebra is the sole species of subgenus *Dolichohippus*. The latter resembles an ass, to which it is closely

3. Large Screen

Animal History



Zebras

Zebras (*/ˈzɛbrə/ zeb-rah or /ˈzi brə/ zee-brə*)[1] are several species of African equids (horse family) united by their distinctive black and white striped coats. Their stripes come in different patterns, unique to each individual. They are generally social animals that live in small harems to large herds. Unlike their closest relatives, horses and donkeys, zebras have never been truly domesticated.

There are three species of zebras: the plains zebra, the Grévy's zebra and the mountain zebra. The plains zebra and the mountain zebra belong to the subgenus *Hippotigris*, but Grévy's zebra is the sole species of subgenus *Dolichohippus*. The latter resembles an ass, to which it is closely related, while the former two are more horse-like. All three belong to the genus *Equus*, along with other living equids.

The unique stripes of zebras make them one of the animals most familiar to people. They occur in a variety of habitats, such as grasslands, savannas, woodlands, thorny scrublands, mountains, and coastal hills. However, various anthropogenic factors have had a severe impact on zebra populations, in particular hunting for skins and habitat destruction. Grévy's zebra and the mountain zebra are endangered. While plains zebras are much more plentiful, one subspecies, the quagga, became extinct in the late 19th century – though there is currently a plan, called the Quagga Project, that aims to breed zebras that are phenotypically similar to the quagga in a process called breeding back.

Zebras evolved among the Old World horses within the last 4 million years. It has been suggested that zebras are polyphyletic and that striped equids evolved more than once. Extensive stripes are posited to have been of little use to equids that live in low densities in deserts (like asses and some horses) or ones that live in colder climates with shaggy coats and annual shading (like some horses). [4] However, molecular evidence supports zebras as a monophyletic lineage



Leopard

2) Stage **a2** (assess)

Assignment:

Create a HTML5 progress bar using JavaScript Timers which is used to indicate how much of a task has been completed, such as loading something on a page or registration process. It is typically displayed as a progress bar and often marked as a percentage from 0 to 100%.

Task Progress

Progress:

Statement Purpose:

This lab will introduce you to regular expressions (REGEX) and its practical implementation by demonstrating the use of regular expressions in different views.

Activity Outcomes:

This lab teaches you the following topics:

- Purpose and use of different regular expressions
- To write functions in html, validate using regular expressions
- Use of REGEX for validating different inputs (email, password etc).
- Use of REGEX for checking certain words in a string/comment.

Instructor Note:

As pre-lab activity, read documentation of REGEX in Javascript and PHP implement a demo example for email structure input that you are going to demonstrate in the lab and also follow the reading/instruction given by your theory instructor.

2) Stage J (Journey)

Introduction

A regular expression is an object that describes a pattern of characters. Regular expressions are nothing more than a sequence or pattern of characters itself. They provide the foundation for pattern-matching functionality. They are used to perform pattern-matching and "search-and-replace" functions on text.

Following link is helpful in creating regular expressions of your requirements.

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Regular_Expressions

3) Stage a1 (apply)

Lab Activities:

Activity 1:

Develop and demonstrate, using Javascript script, HTML document that collects the user input (the valid format is: A digit from 1 to 4 followed by two upper-case characters followed by two digits followed by two upper-case characters followed by three digits; no embedded spaces allowed) of the user. Event handler must be included for the form element that collects this information to validate the input. Messages in the alert windows must be produced when errors are detected.

Solution:

```

1
2 <!DOCTYPE html>
3 <html>
4 <head> <title> Input Validation </title> </head>
5 <body>
6 <form>
7 Note: A digit from 1 to 4, Two upper-case characters, Two digits, Two upper-case characters, Three digits.<br/>
8 User Input: <input type="text" name="user" /> <br/>
9 <input type="button" value="Validate" onclick="valid
10 (user)" />
11 </form>
12
13 <script type="text/javascript">
14 function valid(user)
15 {
16 var pattern1=/^[1-4][A-Z]{2}[0-9]{2}[A-Z]{2}[0-9]{3}$/
17 if(!user.value.match(pattern1)||user.value.length==0)
18 {
19 alert("Invalid User Input!\nEnter a valid input")
20 return false
21 }
22 else alert("Input is valid!")
23 }
24 </script>
25
26 </body>
27 </html>
28

```

Activity 2:

Using REGEX to perform character recognition in a string. Also use global(g) case sensitive(i) and multiline(m) search.

Solution:

```

1 <!DOCTYPE html>
2 <html>
3 <body>
4
5 <p>Click the button to do a global search for characters NOT inside the brackets [h] in a string.</p>
6
7 <button onclick="myFunction()">Try it</button>
8
9 <p id="demo"></p>
10
11 <script>
12 function myFunction() {
13     var str = "Is this all there is?";
14     var patt1 = /^[^h]/g;
15     var result = str.match(patt1);
16     document.getElementById("demo").innerHTML = result;
17 }
18 </script>
19
20 </body>
21 </html>

```

Activity 3:

Validate email field while creating a signup form. Also show the error messages if email is not valid. Use event handler for input field to validate email.

Solution:

```

1 <script src="https://code.jquery.com/jquery-3.1.1.min.js" integrity="sha256-hVVnYaiADRTO2PzUGmuLJr8BLUSjGIZsDYGMiJLv2b8="
  crossorigin="anonymous"></script>
2
3 <form method="post">
4     <label for="email" id="email-ariaLabel">Your email address:</label>
5     <input id="email" name="email" type="email" class="required"/>
6     <span id="error">Not a valid email address</span>
7     <span id="success">A valid email address!</span>
8 </form>
9
10 <script>
11 $(document).ready(function(){
12     $('#success').hide();
13     $('#error').hide();
14     $('#email').keyup(function () {
15         var email = $('#email').val();
16         // i Perform case-insensitive matching
17         // g Perform a global match (find all matches rather than stopping after the first match)
18         // m Perform multiline matching
19         var re = /[A-Z0-9_%+]+@[A-Z0-9.-]+\.[A-Z]{2,4}/igm;
20         if (re.test(email)) {
21             $('#success').show();
22             $('#error').hide();
23         } else {
24             $('#error').show();
25             $('#success').hide();
26         }
27     });
28 });
29 </script>

```

4) Stage v (verify)

Home Activities:

Activity: Validate password field while creating signup form. Ask the user to set a strong password by using at least 1 uppercase letter, 1 lowercase letter, 1 digit and 1 ASCII character show the error messages if password doesn't contain any of these expressions.

5) Stage a2 (assess)

Assignment:

Use REGEX in your projects as it enables you to Get useful and structure set of data. This will be assessed in your semester projects.

Statement Purpose:

This lab will give you introduction and practical implementation of different types of **API (Application Programming Interface)** and will guide you to perform actions against the data extracted from API.

Activity Outcomes:

This lab teaches you the following topics:

- Implement simple URL and API by looking at the difference between two.**
- Two ways of calling an API (function and key).**
- Describe **Events** and **Controls** of API

Describe **the arguments used to perform actions on data in different functions and Events.**

Instructor Note:

As pre-lab activity, read documentation of API you are going to implement in the lab and also as given by your theory instructor.

1) Stage **J** (Journey)

Introduction

An API, or Application Programming Interface, is a set of functions that one computer program makes available to other programs (or developers) so they can talk to it directly without having to give it access to the source code. The most popular APIs are from operating systems like Windows XP or Mac OS X. They allow third-party developers to write programs on top of Microsoft's and Apple's software.

If we type in a URL to a web site and that URL returns data in a structured format, then a basic API is already in place. To take this to the next level, you'll also want to allow developers to perform actions against the data. If there's no need to authenticate your users/developers with an API key, a GET request very well might be your method of choice and we can have the users query everything using just the URL alone. For example, selecting entry 3 from a public record could be as easy as typing in the URL: <http://site.com/api/select/3>.

However, when a lot of parameters need to be set (which would make the URL extremely long), or when security is of greater concern, a POST request is a safer and more common approach to asking data from a server. A POST request will usually require some sort of data to go along with the URL parameters so the server can check to make sure the user/developer has permission to do the type of request they're attempting.

2) Stage **a1** (apply)

Lab Activities:

Using Google Map API

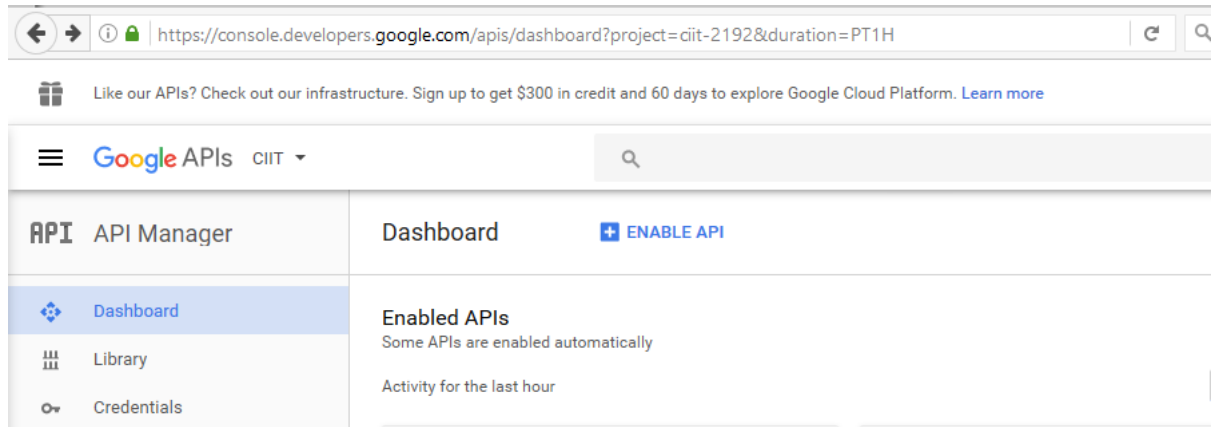
Activity 1:

Creating your own key using your google account.

Solution:

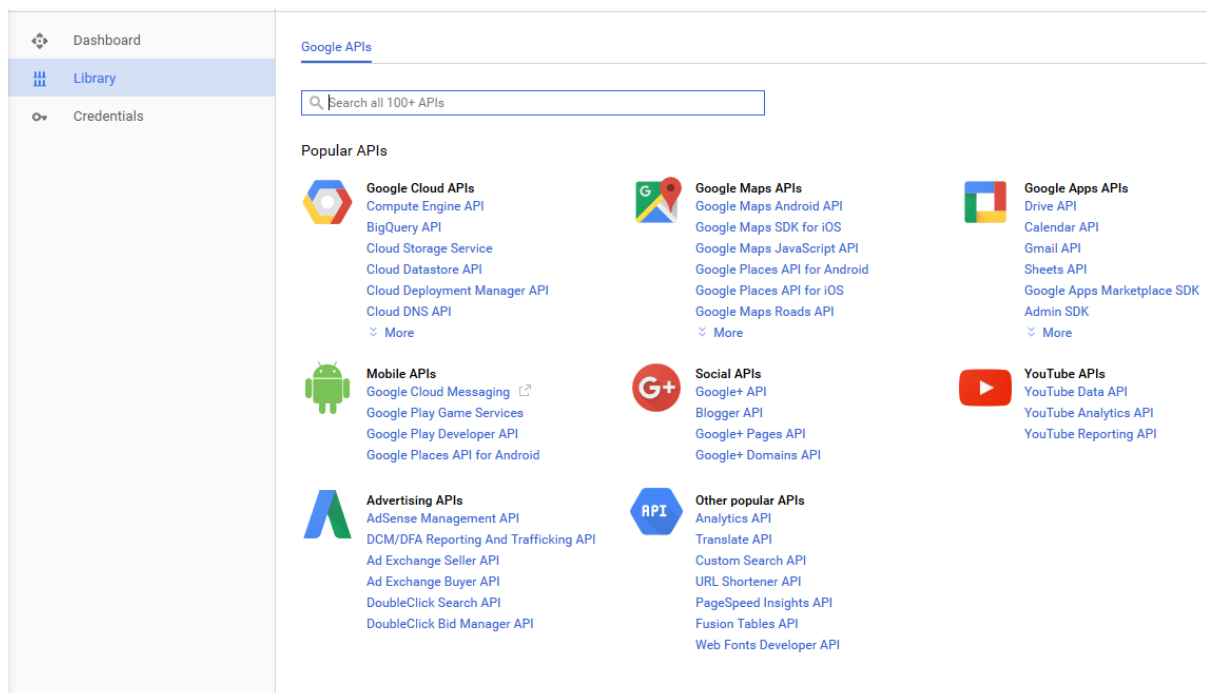
Step1:

You need to create a project for which you are creating your API key. After creating the project, you will be directed towards the following window which will allow you to enable google API.



Step 2:

Click on Enable API the following window will appear. This will allow you to choose API of your requirement.



Step 3:

In this lab we are using API for javascript so I will use Google Maps Javascript API. After you enable the API. Go to Credentials tab in the left panel of above window. That will have your Google Map API key for Javascript.

Activity 2:

Write code to use Google Map API

Solution:

```
1  <!DOCTYPE html>
2  <html>
3  <body>
4
5  <h1>My First Google Map</h1>
6
7  <div id="map" style="width:100%;height:500px"></div>
8
9  <script>
10 function myMap() {
11     var mapCanvas = document.getElementById("map");
12     var mapOptions = {
13         center: new google.maps.LatLng(51.5, -0.2),
14         zoom: 10
15     }
16     var map = new google.maps.Map(mapCanvas, mapOptions);
17 }
18 </script>
19
20 <script src="https://maps.googleapis.com/maps/api/js?key=Your_Key&callback=myMap"></script>
21
22 </body>
23 </html>
```

Activity 3:

Use of Events in Google Map API(Markers)

Solution:

```
1  <!DOCTYPE html>
2  <html>
3
4  <body>
5
6  <div id="map" style="width:100%;height:500px"></div>
7
8  <script>
9  function myMap() {
10     var myCenter = new google.maps.LatLng(51.508742,-0.120850);
11     var mapCanvas = document.getElementById("map");
12     var mapOptions = {center: myCenter, zoom: 5};
13     var map = new google.maps.Map(mapCanvas, mapOptions);
14     var marker = new google.maps.Marker({position:myCenter});
15     marker.setMap(map);
16 }
17 </script>
18
19 <script src="https://maps.googleapis.com/maps/api/js?key=your_key&callback=myMap"></script>
20
21 </body>
22 </html>
```

Activity 4:

Use of polygon function in Google Map API(Markers)

Solution:

```

<!DOCTYPE html>
<html>
<body>

<div id="map" style="width:100%;height:500px"></div>

<script>
function myMap() {
  var stavanger = new google.maps.LatLng(58.983991,5.734863);
  var amsterdam = new google.maps.LatLng(52.395715,4.888916);
  var london = new google.maps.LatLng(51.508742,-0.120850);

  var mapCanvas = document.getElementById("map");
  var mapOptions = {center: amsterdam, zoom: 4};
  var map = new google.maps.Map(mapCanvas,mapOptions);

  var flightPath = new google.maps.Polygon({
    path: [stavanger, amsterdam, london],
    strokeColor: "#0000FF",
    strokeOpacity: 0.8,
    strokeWeight: 2,
    fillColor: "#0000FF",
    fillOpacity: 0.4
  });
  flightPath.setMap(map);
}
</script>
<script src="https://maps.googleapis.com/maps/api/js?key=your_key&callback=myMap"></script>

</body>
</html>

```

3) Stage v (verify)

Home Activities:

The best way to learn about APIs is to use them. After you've gone through the process once, the concepts become a lot easier to understand.

Activity: Use different events from Google Map API documentation and implement those events and controls to get a control on this API

4) Stage a2 (assess)

Statement Purpose:

To familiarize students with the basic programming constructs in PHP

Activity Outcomes:

After this lab the students should be able to use PHP constructs to solve basic programming problems

1) Stage J (Journey)

Introduction

PHP is a powerful server side scripting language. It is used to create dynamic web pages. It is mainly used to manage database at server. PHP can be embedded with HTML in three ways

- We can add blocks of PHP code in HTML
- We can use HTML instructions in PHP
- We can write standalone PHP script

Following are some of the basic rules for writing PHP code

- Blocks of PHP code can be added in HTML code with opening and closing tags, as follows: `<?php and ?>`. Similarly, we can use simply `<? or ?>` or `<script language="PHP">.....</script>` to add PHP blocks in HTML
- PHP statements end with a semicolon
- Comments can be added as: `//` for one line comment and `/*` and `*/` for multiple lines comment

PHP provides several instructions to write output on the browser screen. Most commonly we use echo to write output on the browser screen. The syntax of the echo is

```
echo ("Welcome to PHP");
```

We can also use echo command as

```
echo "Welcome to PHP";
```

print command can also be used to write out put on the browser. The syntax of writing print command is

```
print("Welcome to PHP"); or print "Welcome to PHP";
```

echo is marginally faster as compared to print as echo does not return any value. **printf** function can also be used to output a formatted string.

We can also write HTML instructions in PHP. The echo statement outputs whatever it's told to the browser. It can output not only plain text but also HTML tags. We have to write HTML tags in quotation marks. In PHP single and double quotation marks are used interchangeably but their logical sequence is must to maintain.

A constant is a placeholder for a value that you reference within your code that is formally defined before using it. The name of a constant in PHP begins with a letter or an underscore. Names of constants are case sensitive. Typically they are named using all capital letters. PHP function **define()** is used to assign a value to a constant.

In PHP, variable names begin with \$ sign. First character must be a letter or underscore while remaining characters may be letters, numbers or underscores. Variable names are case sensitive. We don't need to declare or initialize variables. PHP is a loosely typed language it means that data types does not require to be declare explicitly.

An operator is a symbol used to perform a specific operation on variables or operands. PHP provides several types of operators to represent different operations. Following are the operators available in PHP

Arithmetic operators: + (addition), - (subtraction), / (division), * (multiplication) and % (remainder)

Assignment operator: = (assignment), += (Operator adds and assigns a value. For example \$a += \$b, here += adds the value of \$b in \$a and then assigns the result to \$a. -=, *= and /= operators can also be used), .= operator concatenates and assign the value. For example \$.=\$b, here .= concatenates the value of \$b with \$a and then assigns this value to \$a)

String operators: . (Concatenate two strings), .= (concatenates; and assign the value).

Increment and decrement operators: ++ (increment), -- (decrement). These operators can be used either in prefix or postfix form.

Logical operators: AND or && (logical and), OR or || (logical or), Not or ! (logical not) and XOR (logical exclusive-or).

Comparison operators: == (equality), === (checks both types and values of variables), != (inequality), > (greater), < (less), >= (greater or equal) and <= (less or equal).

Conditional statements: Conditional statements make it possible for your computer program to respond accordingly to a wide variety of inputs, using logic to discern between various conditions based on input value. In PHP following conditional statements are available.

If statement: if statements allow code to be executed when the condition specified is met; if the condition is true then the code in the curly braces is executed. Here is the syntax for an if statement:

```
if (condition)  
{  
  statement  
}
```

if...else statement: When you have two possible situations and you want to react differently for each, you can use an if...else statement. This means: "If the conditions specified are met, run the first block of code; otherwise run the second block." The syntax is as follows:

```
if (condition)  
{  
  code to be executed if condition is true  
}  
else  
{  
  code to be executed if condition is false  
}
```

Switch statement: You can think of the switch statement as a variant of the if-else combination, often used when you need to compare a variable against a limited number of values called cases in switch statement terminology. The syntax of the switch statement in PHP is

```
switch(variable)  
{  
  case option:  
  action  
  break;  
  .  
  .  
}
```

Looping statements in PHP: Looping statements are used to execute the same block of code a specified number of times. Following are the basic loops in PHP.

A while loop runs the same block of code while or until a condition is true. Syntax of for loop is given below

while loop:

```
while(condition)  
{  
Statements  
Increment/decrement  
}
```

A do while loop runs once before the condition is checked. If the condition is true, it will continue to run until the condition is false. (The difference between a do and a do while loop is that do while runs once whether or not the condition is met.).

```
do  
{  
Statements  
Increment/decrement  
}  
while(condition)
```

A for loop runs the same block of code a specified number of times (for example, five times).

```
for (init counter; test counter; increment counter) {  
    code to be executed;  
}
```

A foreach loop is used to read an array.

```
foreach (array_expression as $value)  
    { statement}
```

Arrays in PHP: An array is traditionally defined as a group of items that share certain characteristics. Each item consists of two components; the key and a value. Key is the index of the item in the array and value is the value of the item. PHP doesn't require that you assign a size or type to an array at creation time.

Declaring an array: PHP doesn't even require that you declare the array before using it, although you're free to do so. We just add items to the array. We can add an item to a list as

```
$array-name[index of the element]= value;
```

For example we can start and add an item in an array as given below

```
$players[0] = 'Muhammad Yousuf';
```

We can add more items in the similar way but use a different index

```
$players[1]="Ricky Ponting";
```

Accessing an array: we can read an array item just by entering the array name and the index of the item. For example if we want to read and display the value of second item in the \$players array we can do this as given below

```
echo $players[1];
```

Associative array: PHP allow us to create arrays where items are declared by name instead of index or key number. Such an array is called an associative array. An item in an associative array can be added as

```
$array-name[item name] = value
```

For example

```
$players['yousuf']="Muhammad Yousuf";
```

Items in associate array are accessed by name. For example, we can read the value of the above array as

```
echo $players['yousuf'];
```

array() function: we can also use array function to create an array. By using this function we can add multiple items in one line. The syntax of the array function is

```
$array_name=array(item_1,item_2,...item_n);
```

Example is

```
$players=array("M.Yoursuf","Imran Khan");
```

We can also use array function to create associative arrays such as

```
$players=array("Yousuf"=>"M.Yoursuf","imran"=>"Imran Khan");
```

Sorting an array: PHP provides us sort() to sort an array in ascending order while rsort() function to sort an array in descending order.

A function is a block of statements that can be used repeatedly in a program. In PHP, a function can be defined as;

```
function functionName(list of arguments) {  
    code to be executed;  
}
```

To call the function, just write its name along with the required arguments.

2) Stage a1 (apply)

Activity 1:

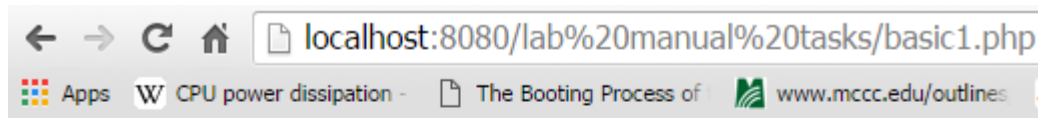
Write the PHP code that creates a table as given below

| Subject | Total Marks | Obtained Marks |
|------------------|-------------|----------------|
| Web Engineering | 100 | 80 |
| Database Systems | 100 | 90 |

Solution:

```
<?php  
echo "<table border=1 >  
<tr> <th><font color=blue>Subject</th> <th>Total Marks</th><th>Obtained Marks</font></td></tr>  
<tr> <td>Web Engineering</td> <td>100</td><td>80</td></tr>  
<tr> <td>Database Systems</td> <td>100</td> <td>90</td></tr>  
</table>";  
?>
```

Out-put:



| Subject | Total Marks | Obtained Marks |
|------------------|-------------|----------------|
| Web Engineering | 100 | 80 |
| Database Systems | 100 | 90 |

Activity 2:

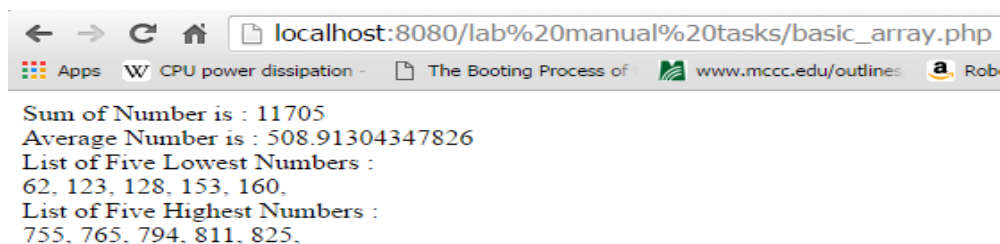
Write a PHP script to calculate and display the sum, average, and five lowest and highest numbers from the given list.

123, 160, 62, 153, 345, 128, 387, 825, 666, 614, 723, 163, 811, 176, 732, 628, 722, 733, 755, 765, 794, 613, 627

Solution:

```
<?php
$num = "123, 160, 62, 153, 345, 128, 387, 825, 666, 614, 723, 163, 811, 176, 732, 628, 722, 733, 755, 765, 794, 613, 627";
$num_array = explode(',', $num);
$tot_num = 0;
$num_array_length = count($num_array);
foreach($num_array as $num)
{
    $tot_num += $num;
}
echo "Sum of Number is : ".$tot_num."
<br>";
$avg_num = $tot_num/$num_array_length;
echo "Average Number is : ".$avg_num."
<br>";
sort($num_array);
echo "
List of Five Lowest Numbers : <br>";
for ($i=0; $i< 5; $i++)
{
    echo $num_array[$i].", ";
}
echo "<br>";
List of Five Highest Numbers : <br>";
for ($i=(count($num_array)-5); $i< (count($num_array)); $i++)
{
    echo $num_array[$i].", ";
}
?>
```

Out-put:



Activity 3:

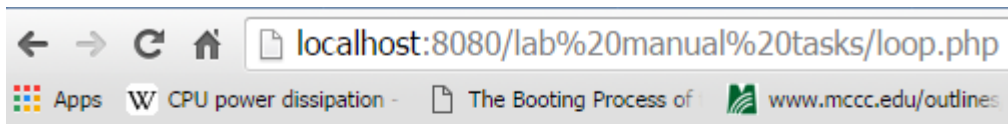
Write a PHP script to create the following table (using looping statement).

| | | | | |
|---------|---------|---------|---------|---------|
| 1+1=2 | 1+2=3 | 1+3=4 | 1+4=5 | 1+5=6 |
| 2+1=3 | 2+2=4 | 2+3=5 | 2+4=6 | 2+5=7 |
| 3+1=4 | 3+2=5 | 3+3=6 | 3+4=7 | 3+5=8 |
| 4+1=5 | 4+2=6 | 4+3=7 | 4+4=8 | 4+5=9 |
| 5+1=6 | 5+2=7 | 5+3=8 | 5+4=9 | 5+5=10 |
| 6+1=7 | 6+2=8 | 6+3=9 | 6+4=10 | 6+5=11 |
| 7+1=8 | 7+2=9 | 7+3=10 | 7+4=11 | 7+5=12 |
| 8+1=9 | 8+2=10 | 8+3=11 | 8+4=12 | 8+5=13 |
| 9+1=10 | 9+2=11 | 9+3=12 | 9+4=13 | 9+5=14 |
| 10+1=11 | 10+2=12 | 10+3=13 | 10+4=14 | 10+5=15 |

Solution:

```
<!DOCTYPE html>
<html>
<head>
<title> LOOP Example</title>
</head>
<body>
<table border="1">
<?php
for($row=1;$row<=10;$row++)
{
echo "<tr>";
for ($col=1;$col<=5;$col++)
{
$sum=$row+$col;
echo "<td>".$row. "+". $col . "=". $sum."</td>";
}
echo "</tr>";
}
?>
</table>
</body>
</html>
```

Out-put:



Activity 4:

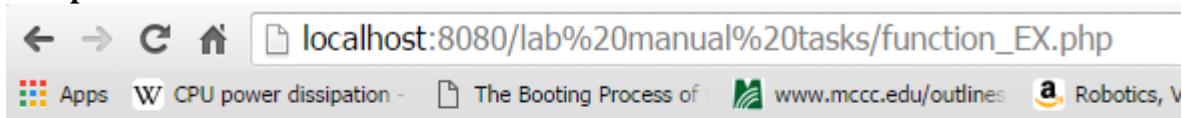
Write a PHP function that gets a number as an argument and calculates and displays its factorial.

Solution:

```
<?php
function findFact($n)
{
    if($n<0)
        echo "Please enter a positive number";

    elseif($n ==0)
    {
        return 1;
    }
    else
    {
        return $n * findFact($n-1);
    }
}
$num=6;
echo "Factorial of $num is: ".$factorial=findFact($num);
?>
```

Out-put:



3) Stage ▼ (verify)

Home Activities:

1. Write a PHP program that finds the sum of all prime number from an array

2. Write a nested `for` loop in the PPHP that prints the following output:

```
1
 1 2 1
 1 2 4 2 1
 1 2 4 8 4 2 1
 1 2 4 8 16 8 4 2 1
 1 2 4 8 16 32 16 8 4 2 1
 1 2 4 8 16 32 64 32 16 8 4 2 1
1 2 4 8 16 32 64 128 64 32 16 8 4 2 1
```

Statement Purpose:

To familiarize the students with the use of Laravel Framework.

Activity Outcomes:

After this lab, the students should be able to setup an environment for Laravel Framework and create a basic application using the Framework.

1) Stage J (Journey)

Introduction

Laravel is a free, open-source PHP web framework, created by Taylor Otwell and intended for the development of web applications following the model view controller (MVC) architectural pattern and based on Symfony. Some of the features of Laravel are a modular packaging system with a dedicated dependency manager, different ways for accessing relational databases, utilities that aid in application deployment and maintenance, and its orientation toward syntactic sugar.

Laravel 1, 2, and 3

The first beta of Laravel 1 was released in June 2011, and it consisted of entirely custom code. It featured a custom ORM (Eloquent), Closure routing (inspired by Ruby Sinatra), a module system for extension, and helpers for forms, validation, authentication, and more. Early Laravel development moved quickly, and Laravel 2 and 3 were released in November 2011 and February 2012 respectively. They introduced controllers, unit testing, a CLI tool, an IOC container, Eloquent relationships, and migrations.

Laravel 4

With Laravel 4, Taylor rewrote the entire framework from the ground up. By this point Composer was showing signs of becoming an industry standard and Taylor saw the value of rewriting the framework as a collection of components, distributed and bundled together with Composer. Taylor developed a set of components under the code-name Illuminate and, in May of 2013, released Laravel 4 as a fresh look at Laravel, based on pulling in Symfony and Illuminate packages in via Composer.

Laravel 4 also introduced queues, a mail component, Façades, and database seeding. And because Laravel was now relying on Symfony components, it was announced that Laravel would be mirroring (not exactly, but soon-after) the release-every-6-months release schedule Symfony follows.

Laravel 5

Laravel 4.3 was scheduled to release in November 2014, but as development progressed, it became clear that the significance of its changes merited a major release, and Laravel 5 was released in February 2015.

Laravel 5 introduced a revamped directory structure, removal of the form and HTML helpers, the introduction of the Contract interfaces, a spate of new views, Socialite for social media authentication, Elixir for asset compilation, Scheduler to simplify cron, dotenv for simplified environment management, Form Requests, and a brand new CLI.

Features

The following are some key features of the Laravel Framework:

- Bundles provide a modular packaging system since the release of Laravel 3, with bundled features already available for easy addition to applications. Furthermore, Laravel 4 uses Composer as a dependency manager to add framework-agnostic and Laravel-specific PHP packages available from the Packagist repository.
- Eloquent ORM (object-relational mapping) is an advanced PHP implementation of the active record pattern, providing at the same time internal methods for enforcing constraints on the relationships between database objects. Following the active record pattern, Eloquent ORM presents database tables as classes, with their object instances tied to single table rows.
- Query builder, available since Laravel 3, provides a more direct database access alternative to the Eloquent ORM. Instead of requiring SQL queries to be written directly, Laravel's query builder provides a set of classes and methods capable of building queries programmatically. It also allows selectable caching of the results of executed queries.
- Application logic is an integral part of developed applications, implemented either by using controllers or as part of the route declarations. The syntax used to define application logic is similar to the one used by Sinatra framework.
- Blade templating engine combines one or more templates with a data model to produce resulting views, doing that by transpiling the templates into cached PHP code for improved performance. Blade also provides a set of its own control structures such as conditional statements and loops, which are internally mapped to their PHP counterparts. Furthermore, Laravel services may be called from Blade templates, and the templating engine itself can be extended with custom directives.

2) Stage **a1** (apply)

Lab Activities:

Activity 1:

Install the Laravel Framework on your system. It will include following steps,

- Installing Composer

- Verify Compose Installation.
- Install Laravel.
- Verify Laravel Installation.

Solution:

Composer

Whatever machine you are developing on will need to have Composer installed globally. If you are not familiar with Composer, it is the foundation of most modern PHP development. Composer is a dependency manager for PHP, much like NPM for Node or Ruby Gems for Ruby. You will need Composer to install Laravel, update Laravel, and bring in external dependencies. Download the latest version of Compose installable archive file from <https://getcomposer.org/download/>

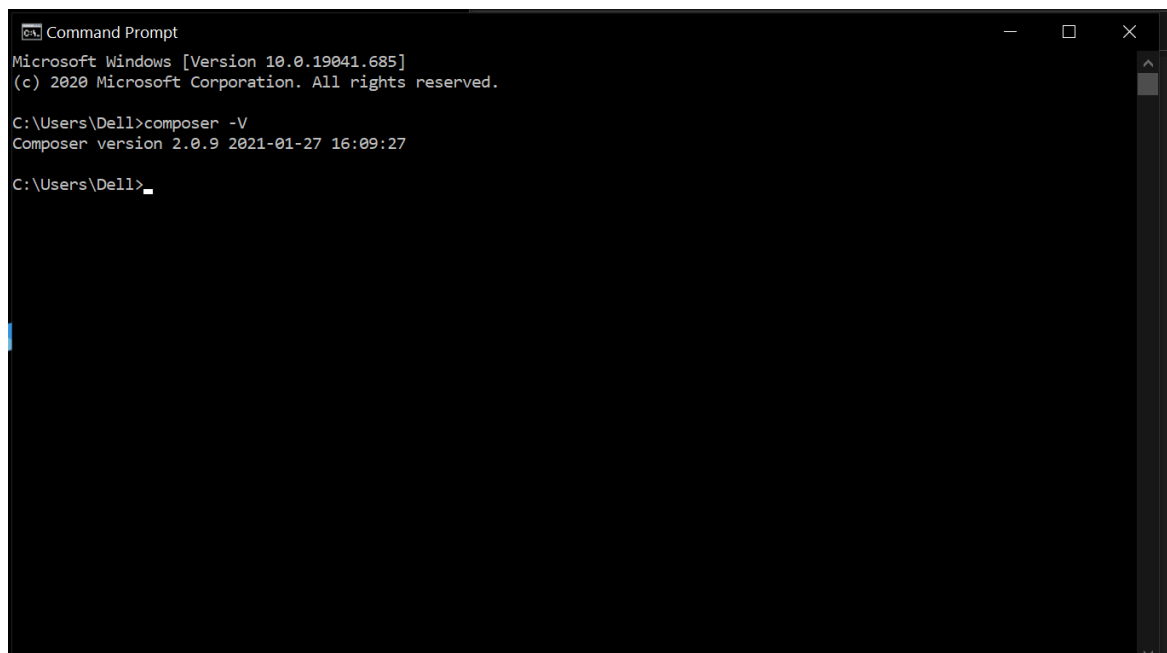
Windows Installer

Download and run `Composer-Setup.exe` - it will install the latest composer version whenever it is executed.

The installer - which requires that you have PHP already installed - will download Composer for you and set up your PATH environment variable so you can simply call composer from any directory.

Verify Compose Installation.

To verify installation, open your command prompt, and type `composer -V`. The system will return the installed version of the composer on your system.



```
Command Prompt
Microsoft Windows [Version 10.0.19041.685]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\Dell>composer -V
Composer version 2.0.9 2021-01-27 16:09:27

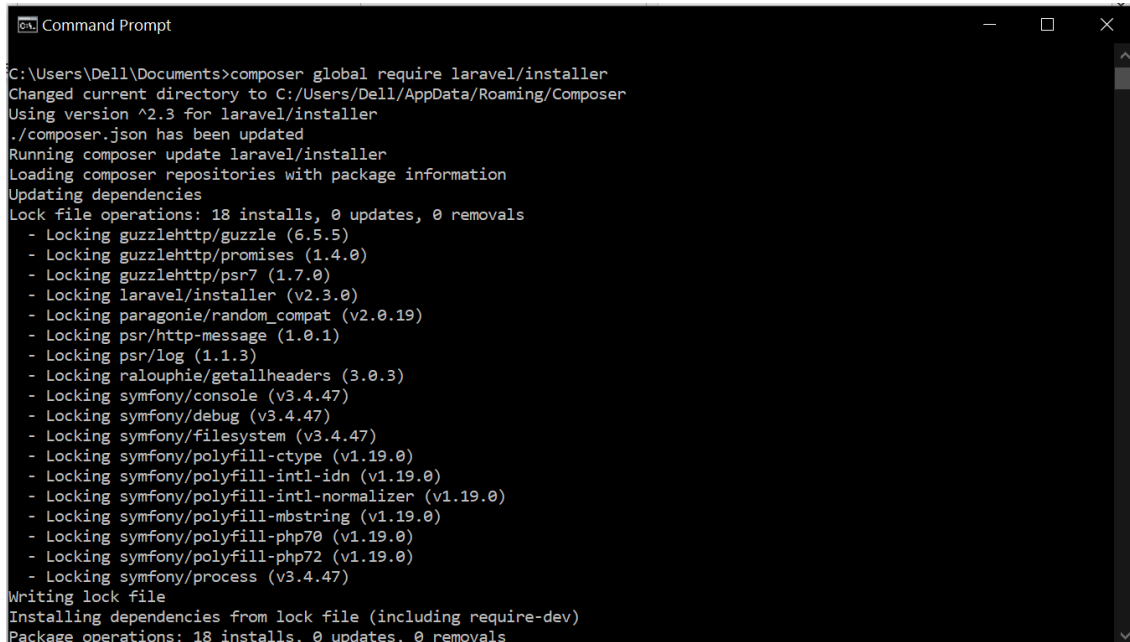
C:\Users\Dell>
```

The Laravel Installer

To install the Laravel as a global Composer dependency, type the following command in the command prompt:

```
Composer global require Laravel/installer
```

It will install Laravel Framework in your system. The screenshot below also describes the procedure.



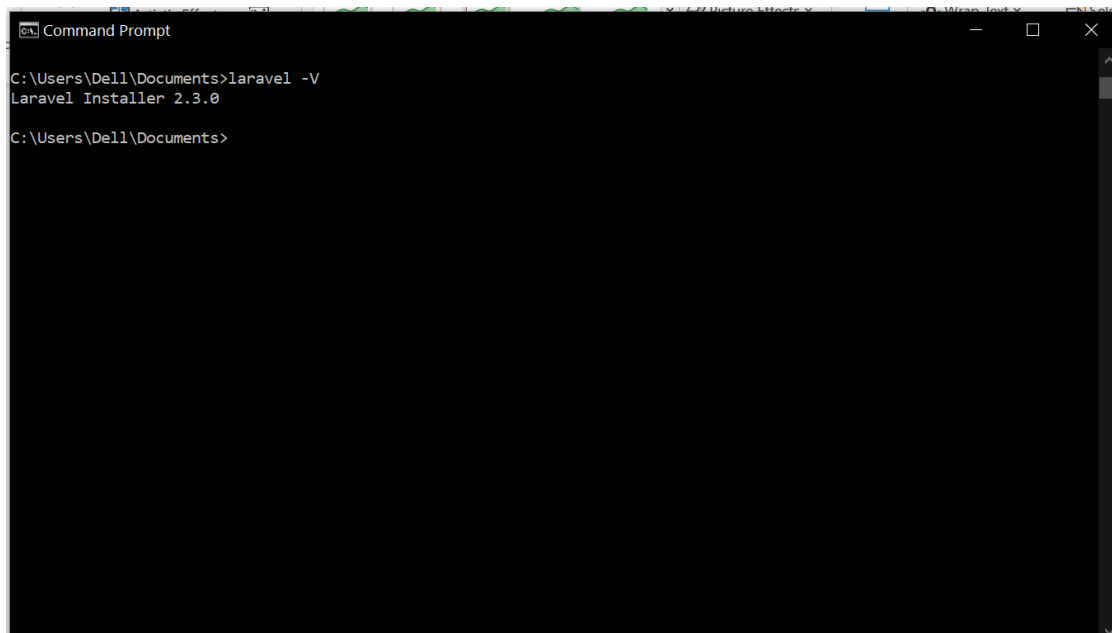
```
Command Prompt
C:\Users\Dell\Documents>composer global require laravel/installer
Changed current directory to C:/Users/Dell/AppData/Roaming/Composer
Using version ^2.3 for laravel/installer
./composer.json has been updated
Running composer update laravel/installer
Loading composer repositories with package information
Updating dependencies
Lock file operations: 18 installs, 0 updates, 0 removals
- Locking guzzlehttp/guzzle (6.5.5)
- Locking guzzlehttp/promises (1.4.0)
- Locking guzzlehttp/psr7 (1.7.0)
- Locking laravel/installer (v2.3.0)
- Locking paragonie/random_compat (v2.0.19)
- Locking psr/http-message (1.0.1)
- Locking psr/log (1.1.3)
- Locking ralouphie/getallheaders (3.0.3)
- Locking symfony/console (v3.4.47)
- Locking symfony/debug (v3.4.47)
- Locking symfony/filesystem (v3.4.47)
- Locking symfony/polyfill-ctype (v1.19.0)
- Locking symfony/polyfill-intl-idn (v1.19.0)
- Locking symfony/polyfill-intl-normalizer (v1.19.0)
- Locking symfony/polyfill-mbstring (v1.19.0)
- Locking symfony/polyfill-php70 (v1.19.0)
- Locking symfony/polyfill-php72 (v1.19.0)
- Locking symfony/process (v3.4.47)
Writing lock file
Installing dependencies from lock file (including require-dev)
Package operations: 18 installs, 0 updates, 0 removals
```

Verify Laravel Installation.

To verify installation of the Laravel framework on your system, type following command in the command prompt.

```
laravel -V
```

It will return the installed version of the Laravel on your system.



```
Command Prompt
C:\Users\Dell\Documents>laravel -V
Laravel Installer 2.3.0
C:\Users\Dell\Documents>
```

Activity 2:

Creating your first project with the Laravel Framework. It will include,

- Creating a new Laravel Project
- Preview the Project in the Browser.


Solution:

Creating a new Laravel Project

Once you have the Laravel installer tool installed, spinning up a new Laravel project is simple. Just run the command `laravel new <ProjectName>` from your command line.

This will create a new subdirectory of your current directory named `<ProjectName>` and install a bare Laravel project in it.

The following screenshot creates a new project with the name `studentsrecords`.



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19041.746]
(c) 2020 Microsoft Corporation. All rights reserved.

D:\Courses\Web Technologies-SPRING_2021\laravel>laravel new studentrecords

Laravel

Creating a "laravel/laravel" project at "./studentrecords"
Installing laravel/laravel (v8.5.9)
- Installing laravel/laravel (v8.5.9): Extracting archive
Created project in D:\Courses\Web Technologies-SPRING_2021\laravel\studentrecords
> @php -r "file_exists('.env') || copy('.env.example', '.env');"
Loading composer repositories with package information
Updating dependencies
Lock file operations: 105 installs, 0 updates, 0 removals
- Locking asm89/stack-cors (v2.0.2)
- Locking brick/math (0.9.2)
- Locking dnoegel/php-xdg-base-dir (v0.1.1)
- Locking doctrine/inflector (2.0.3)
- Locking doctrine/instantiator (1.4.0)
- Locking doctrine/lexer (1.2.1)
- Locking dragonmantank/cron-expression (v3.1.0)
- Locking egulias/email-validator (2.1.25)
- Locking facade/flare-client-php (1.3.7)
- Locking facade/ignition (2.5.12)
```

The directory structure of the ‘studentsrecords’ project can be seen in the screenshot provided below:

his PC > data (D:) > Courses > Web Technologies-SPRING_2021 > laravel > studentrecords >

| Name | Date modified | Type | Size |
|----------------|--------------------|--------------------|--------|
| app | 2/16/2021 11:21 AM | File folder | |
| bootstrap | 2/16/2021 11:21 AM | File folder | |
| config | 2/16/2021 11:21 AM | File folder | |
| database | 2/16/2021 11:21 AM | File folder | |
| public | 2/16/2021 11:21 AM | File folder | |
| resources | 2/16/2021 11:21 AM | File folder | |
| routes | 2/16/2021 11:21 AM | File folder | |
| storage | 2/16/2021 11:21 AM | File folder | |
| tests | 2/16/2021 11:21 AM | File folder | |
| vendor | 2/16/2021 11:23 AM | File folder | |
| .editorconfig | 2/16/2021 11:21 AM | EDITORCONFIG File | 1 KB |
| .env | 2/16/2021 11:23 AM | ENV File | 1 KB |
| .env.example | 2/16/2021 11:23 AM | EXAMPLE File | 1 KB |
| .gitattributes | 2/16/2021 11:21 AM | GITATTRIBUTES File | 1 KB |
| .gitignore | 2/16/2021 11:21 AM | GITIGNORE File | 1 KB |
| .styleci.yml | 2/16/2021 11:21 AM | YML File | 1 KB |
| artisan | 2/16/2021 11:21 AM | File | 2 KB |
| composer.json | 2/16/2021 11:21 AM | JSON File | 2 KB |
| composer.lock | 2/16/2021 11:21 AM | LOCK File | 263 KB |
| package.json | 2/16/2021 11:21 AM | JSON File | 1 KB |
| phpunit.xml | 2/16/2021 11:21 AM | XML Document | 2 KB |
| README.md | 2/16/2021 11:21 AM | MD File | 4 KB |
| server.php | 2/16/2021 11:21 AM | PHP File | 1 KB |
| webpack.mix.js | 2/16/2021 11:21 AM | JavaScript File | 1 KB |

Preview the Project.

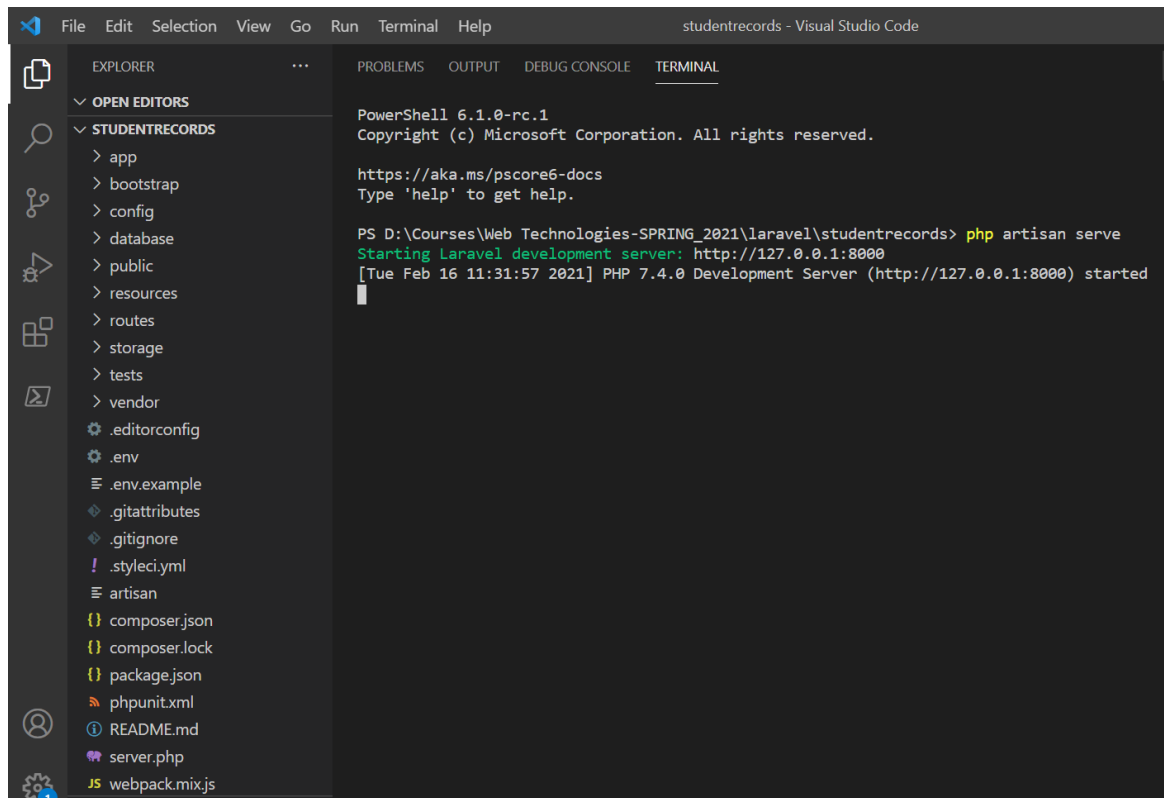
To view the website in the browser, we can use `artisan`. Artisan is the command line interface included with Laravel. Artisan exists at the root of your application as the `artisan` and provides a number of helpful commands that can assist you while you build your application. To view a list of all available `artisan` commands, you may use the `list` command:

```
php artisan list
```

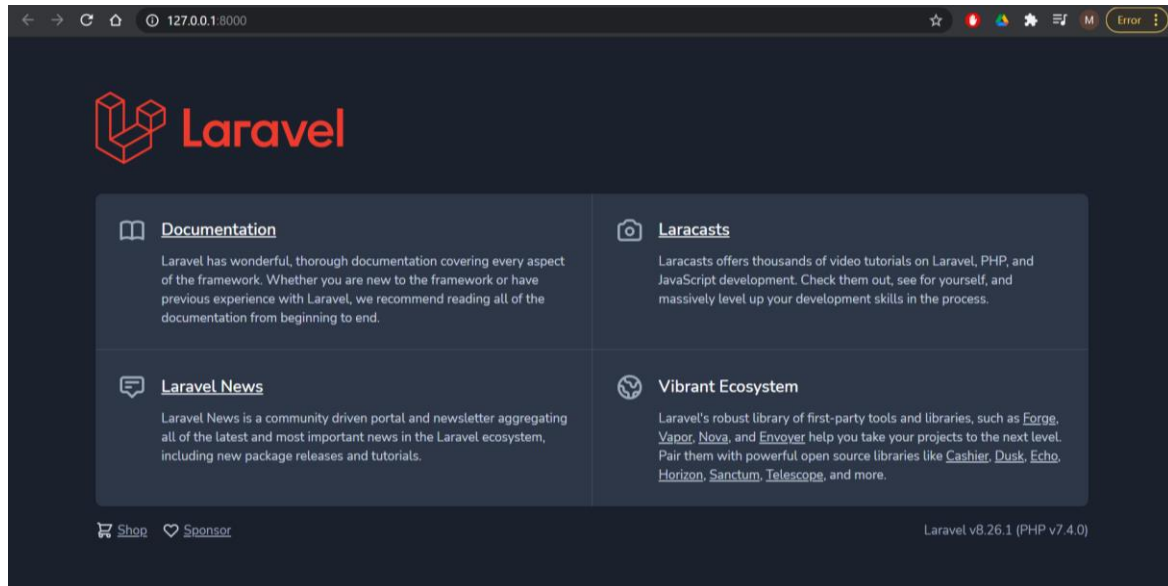
To preview the project, we can use the following command in the command prompt.

```
php artisan serve
```

The screenshot below shows after we run the command. It starts Laravel development server at `http://127.0.0.1:8000`



Opening the URL <http://127.0.0.1:8000> in the browser will generate the following default landing page of the Laravel Project.



Activity 3:

Creating routes and views in your project with the Laravel Framework. It will include,

- Defining Routes in project 'studentrecords'
- Defining Views in project 'studentrecords'
- Creating Controllers for project 'studentrecords'

Solution:

Routes:

The essential function of any web application framework is taking requests from a user and delivering responses, usually via HTTP(S). This means defining an application's routes is the first and most important concept to approach when learning a web framework; without routes, you have no ability to interact with the end user.

In Laravel application, web routes are defined in `routes/web.php`. The simplest way to define a route is to match a path (for example, `/`) with a closure, as described in following example,

```
Route::get('/', function () {  
    return 'Hello, World!';  
});
```

We have now defined that, if anyone visits `/` (the root of your domain), Laravel's router should run the Closure defined there and return the result. Note that we return our content and don't echo or print it.

Many simple websites could be defined within the web routes file. With a few simple GET routes combined with some templates, we can serve a classic website easily.

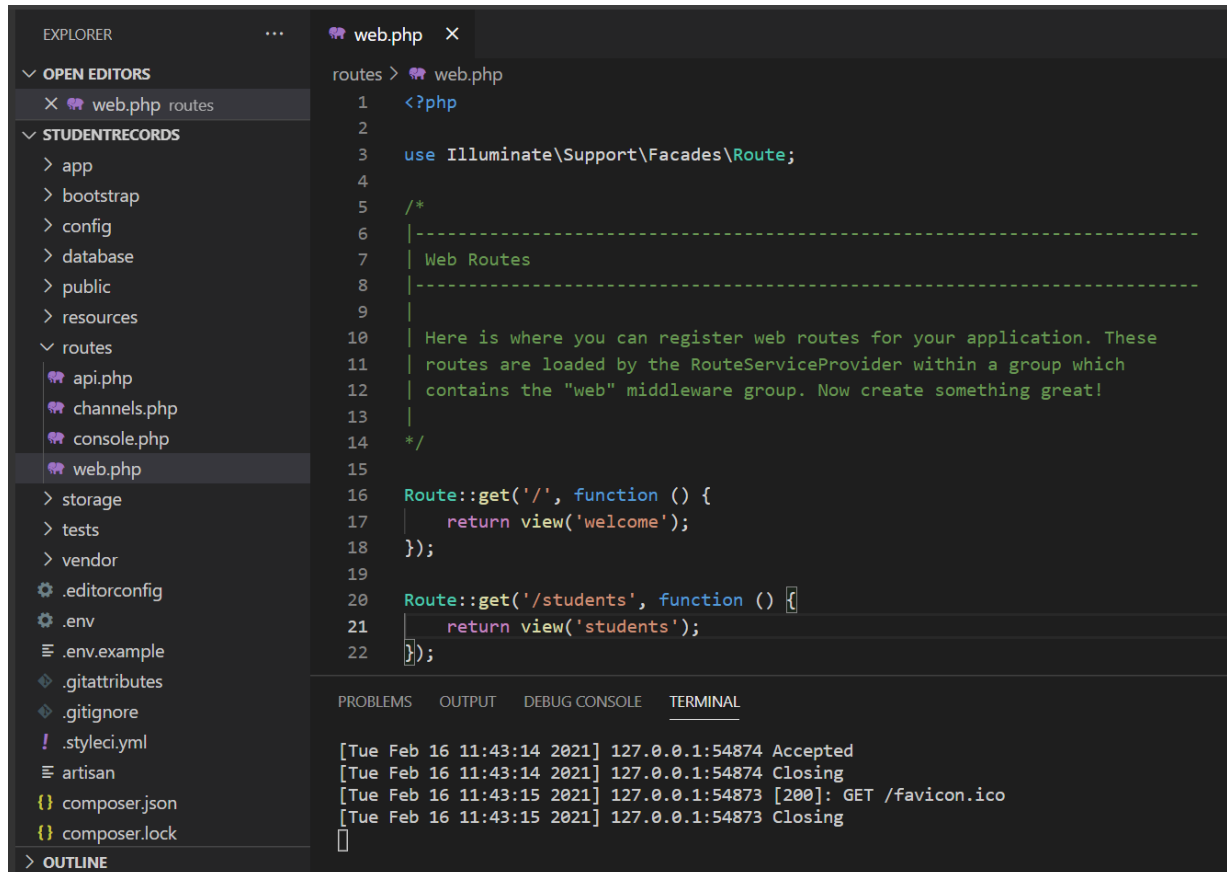
```
Route::get('/', function () {  
    return view('welcome');  
});  
  
Route::get('about', function () {  
    return view('about');  
});  
  
Route::get('products', function () {  
    return view('products');  
});  
  
Route::get('services', function () {  
    return view('services');  
});
```

Defining Routes in project 'studentrecords'

For defining a new route `/students` in addition to the existing route `/`, open the file `routes/web.php` and add following statement at the end of the file :

```
Route::get('/students', function () {
    return view('students');
});
```

The screenshot after adding the route is provided below:



Views:

Views (or Templates) are files that describe how some particular output should look like. You might have views for JSON or XML or emails, but the most common views in a web framework output HTML.

In Laravel, there are two formats of view you can use out of the box: Blade or PHP. The difference is in the filename: `about.php` will be rendered with the PHP engine, and `about.blade.php` will be rendered with the Blade engine.

```
Route::get('/', function () {
    return view('home');
});
```

This code above looks for a view in `resources/views/home.blade.php` or `resources/views/home.php` and loads its contents and parses any inline PHP or control structures until you have just the view's output. Once you return it, it is passed on to the rest of the application and eventually returned to the user.

Defining Views in project 'studentrecords'

For defining a view `students`, add the file `resources/views/students.blade.php` and add following statements inside the `<body>` tag:

```
<div class="flex justify-center min-h-screen sm:items-center sm:pt-0">
  <table align="center">
    <tr><th>Students List</th></tr>
    <tr>
      <td>This Laravel Application will display Students informat
ion here.</td>
    </tr>
  </table>
</div>
```

Visit URL <http://127.0.0.1:8000/students> , it will open the following Web Page.



Controllers:

In the MVC framework, the letter 'C' stands for Controller. It acts as a directing traffic between Views and Models.

Instead of defining all of your request handling logic as closures in your route files, you may wish to organize this behavior using "controller" classes. Controllers can group related request handling logic into a single class. For example, a `UserController` class might handle all incoming requests related to users, including showing, creating, updating, and deleting users. By default, controllers are stored in the `app/Http/Controllers` directory.

Creating a Controller

Open the command prompt or terminal based on the operating system you are using and type the following command to create controller using the Artisan CLI (Command Line Interface).

```
php artisan make:controller <ControllerName>
```

Replace the <controller-name> with the name of your controller. The created constructor can be seen at **app/Http/Controllers**.

A basic controller code-snippet will look something like this, and you have to create in the directory like **app/Http/Controller/AdminController.php**:

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class AdminController extends Controller

{

    //

}
```

The controller that you have created can be invoked from within the **routes.php** file using this syntax below-

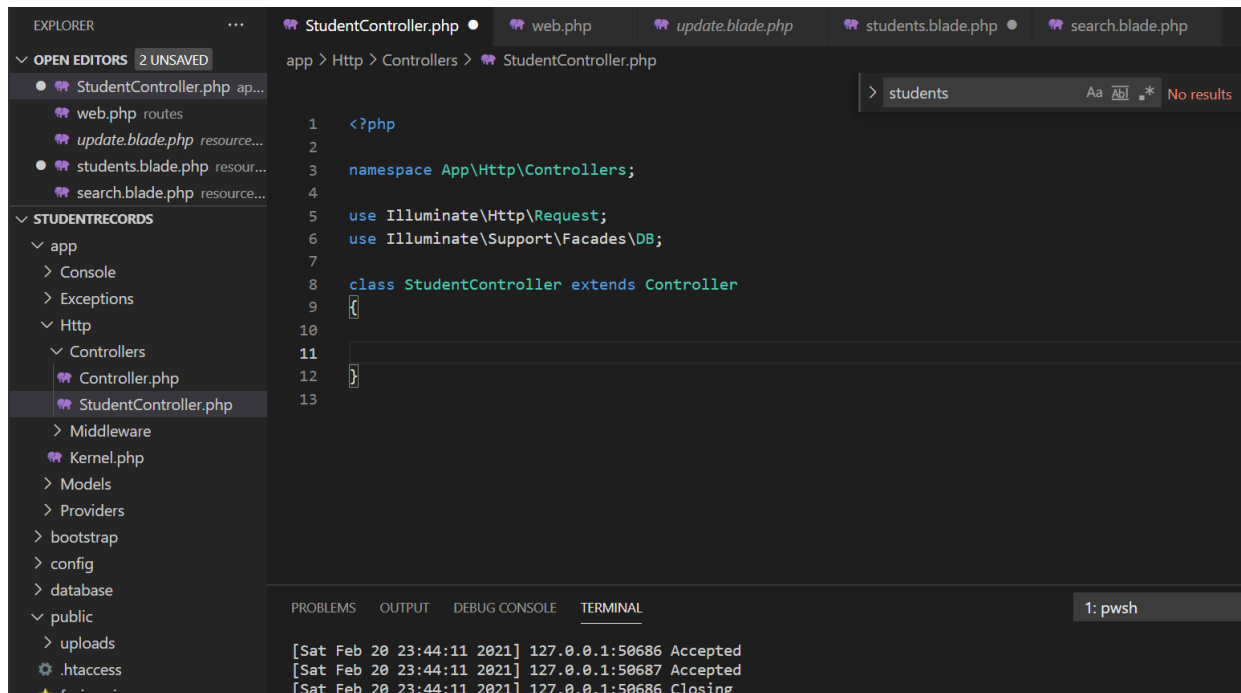
```
Route::get('base URI', 'controller@method');
```

Creating Controller in project 'studentrecords'

For defining the controller 'StudentsController' in the project 'studentrecords', open the command prompt and type the following command:

```
php artisan make:controller StudentController
```

It will create a controller file with the name StudentController at **app/Http/Controller/StudentController.php**.



Add following PHP function in the class `StudentController.php`

```
public function index () {
    return view('students');
}
```

To invoke controller from the routes file, add the following statement in `routes/web.php` file.

```
Route::get('/students', 'App\Http\Controllers\StudentController@index');
```

Open the URL <http://127.0.0.1:8000/students>, it will display the following Web Page.



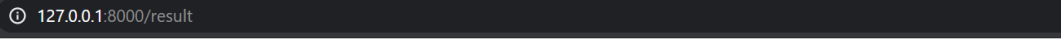
Students List
This Laravel Web Application will display Students Information here

3) Stage **V** (verify)

Home Activities:

Activity 1:

Modify the activities completed during the lab to create a static HTML view which returns the following Web Page.



127.0.0.1:8000/result

| Students List | | | | |
|---------------|--------------|-----------------|-------------|----------------|
| S.No | Name | Subject | Total Marks | Marks Obtained |
| 1 | Adeel Ahmed | Web Engineering | 100 | 79 |
| 2 | Asad Mehmood | Web Engineering | 100 | 79 |

4) Stage **a2** (assess)

Assignment:

Define routes and views for the following Web pages of a company using Laravel Framework.

1. Home
2. Profile
3. Clients
4. Products
5. Contact Us

Statement Purpose:

To familiarize the students with the use of Blade Templating in the Laravel Framework.

Activity Outcomes:

After this lab, the students should be able to use Blade template in Laravel for creating different views in the Web Application.

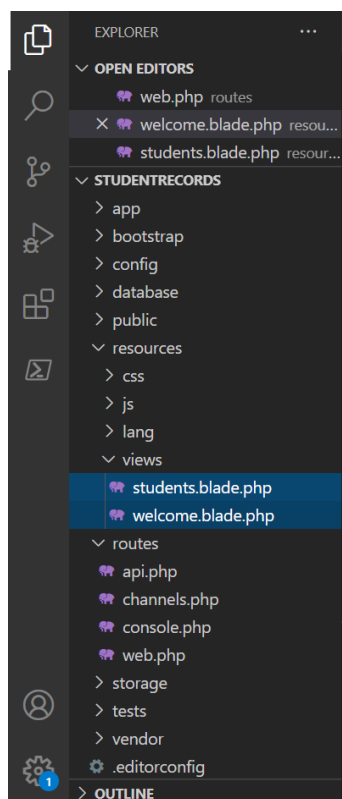
2) Stage J (Journey)

Introduction

Laravel 5.1 introduces the concept of using **Blade**, a templating engine to design a unique layout. The layout thus designed can be used by other views and includes a consistent design and structure.

Blade is the simple, yet powerful templating engine that is included with Laravel. Unlike some PHP templating engines, Blade does not restrict you from using plain PHP code in your templates. In fact, all Blade templates are compiled into plain PHP code and cached until they are modified, meaning Blade adds essentially zero overhead to your application. Blade template files use the `.blade.php` file extension and are typically stored in the `resources/views` directory.

The complete directory structure of Laravel Blade templates is shown in the screenshot given here.



You can observe that all views are stored in the `resources/views` directory and the default view for Laravel framework is `welcome.blade.php`.

Passing Data to Views

Blade views may be returned from routes or controller using the global `view` helper. Data may be passed to the Blade view using the `view` helper's second argument:

```
Route::get('/', function () {  
    return view('greeting', ['name' => 'Finn']);  
});
```

3) Stage a1 (apply)

Lab Activities:

Activity 1:

In first activity of this lab, we will learn how to display data in blade template of Laravel Framework.

Echoing Data Passed to Views:

You may display data that is passed to your Blade views by wrapping the variable in curly braces. For example, given the following `students` route in the project `studentrecords`:

```
Route::get('/students', function () {  
    return view('students', ['name' => 'Muhammad Ali']);  
});
```

You may display the contents of the `name` variable in `resources/views/students.blade.php` like so:

```
<tr><td>{{$name}}</td></tr>
```

The screenshots of the above steps are given below:

```
EXPLORER
... web.php x welcome.blade.php students.blade.php
OPEN EDITORS
web.php routes
welcome.blade.php resour...
students.blade.php resour...
STUDENTRECORDS
app
bootstrap
config
database
public
resources
routes
api.php
channels.php
console.php
web.php
routes > web.php
9 |
10 | Here is where you can register web routes for your application. These
11 | routes are loaded by the RouteServiceProvider within a group which
12 | contains the "web" middleware group. Now create something great!
13 |
14 | */
15 |
16 | Route::get('/', function () {
17 |     return view('welcome');
18 | });
19 |
20 | Route::get('/students', function () {
21 |     return view('students', ['name' => 'Muhammad Ali']);
22 | });
```

```
EXPLORER
... web.php students.blade.php x
OPEN EDITORS
web.php routes
students.blade.php resour...
STUDENTRECORDS
app
bootstrap
config
database
public
resources
css
js
lang
views
students.blade.php
welcome.blade.php
routes
storage
tests
vendor
resources > views > students.blade.php
13 | <style>
14 |     /*! normalize.css v8.0.1 | MIT License | github.com/necolas/normalize.
15 | </style>
16 |
17 | <style>
18 |     body {
19 |         font-family: 'Nunito';
20 |     }
21 | </style>
22 | </head>
23 | <body class="antialiased">
24 |     <div class="flex justify-center min-h-screen sm:items-center sm:pt-0">
25 |         <table class="title">
26 |             <tr><th>Students List</th></tr>
27 |             <tr><td>{{ $name }}</td></tr>
28 |         </table>
29 |     </div>
30 | </body>
31 |
32 | </html>
33 |
```



Students List
Muhammad Ali

You are not limited to displaying the contents of the variables passed to the view. You may also echo the results of any PHP function. In fact, you can put any PHP code you wish inside of a Blade echo statement:

```
The current UNIX timestamp is {{ time() }}.
```

Activity 2:

In this activity, we will understand the implementation of blade directives in views.

Using Blade Directives in Views:

In addition to displaying data, Blade also provides convenient shortcuts for common PHP control structures, such as conditional statements and loops. These shortcuts provide a very clear, concise way of working with PHP control structures while also remaining familiar to their PHP counterparts.

If Statements

You may construct `if` statements using the `@if`, `@elseif`, `@else`, and `@endif` directives. These directives function identically to their PHP counterparts:

```
@if (count($records) === 1)
    I have one record!
@endif
}elseif (count($records) > 1)
    I have multiple records!
@else
    I don't have any records!
```

Switch Statements

Switch statements can be constructed using the `@switch`, `@case`, `@break`, `@default` and `@endswitch` directives:

```
@switch($i)
    @case(1)
        First case...
```

```

@break

@case(2)
    Second case...

@break

@default
    Default case...

@endswitch

```

Loops

In addition to conditional statements, Blade provides simple directives for working with PHP's loop structures. Again, each of these directives functions identically to their PHP counterparts:

```

@for ($i = 0; $i < 10; $i++)
    The current value is {{ $i }}
@endfor

@foreach ($users as $user)
    <p>This is user {{ $user->id }}</p>
@endforeach

@forelse ($users as $user)
    <li>{{ $user->name }}</li>
@empty
    <p>No users</p>
@endforelse

@while (true)
    <p>I'm looping forever.</p>

```

```
@endwhile
```

Comments

Blade also allows you to define comments in your views.

```
{{-- This comment will not be present in the rendered HTML --}}
```

For more directives visit the following URL:

<https://laravel.com/docs/8.x/blade>

Using Blade Directives in project ‘studentrecords’:

- In the project ‘studentprojects’, we will use loop blade directive to display information about the students. For example, if we pass `$students` array described below having different attributes of the individual students.
- Place the following code snippet in the `routes/web.php` file which is then handled by the view ‘students’ to display data on the Web Page for the user.

```
Route::get('/students', function () {
    $students=[
        ['name'=>'Muhammad Ali', 'Email'=>'ali@gmail.com', 'CNIC' => 1234],
        ['name'=>'Muhammad Usman', 'Email'=>'usman@gmail.com', 'CNIC' => 1234],
        ['name'=>'Muhammad Arslan', 'Email'=>'arslan@gmail.com', 'CNIC' => 1234]
    ];
    return view('students', ['students' => $students]);
});
```

- The data passed from the route file is then received in the view. It is traversed using `@for` loop directive. The code for this action is given below:
- Place the following code in the view ‘views/students.blade.php’. It receives the data passed by the route file and displays it on the page inside the `div` tag. The screenshot for the Web page is also give below:

```
<div class="flex justify-center min-h-screen sm:items-center sm:pt-0">
    <table class="title" border="1" align="center">
        <tr><th bgcolor="#666699" colspan="4">Students List</th></tr>
        <tr>
            <th bgcolor="#666699">S.No</th>
            <th bgcolor="#666699">Name</th>
            <th bgcolor="#666699">Email</th>
            <th bgcolor="#666699">CNIC</th>
```

```

        </tr>

        @for ($i = 0; $i < count($students); $i++)
            <tr>
                <td bgcolor="#6699FF" align="center">{{$i}}</td>
                <td bgcolor="#6699FF" width="150" align="center">{{$st
udents[$i]['name']}}</td>
                <td bgcolor="#6699FF" width="150" align="center">{{$st
udents[$i]['Email']}}</td>
                <td bgcolor="#6699FF" width="150" align="center">{{$st
udents[$i]['CNIC']}}</td>
            </tr>
        @endfor

    </table>

```



| Students List | | | |
|---------------|-----------------|------------------|------|
| S.No | Name | Email | CNIC |
| 0 | Muhammad Ali | ali@gmail.com | 1234 |
| 1 | Muhammad Usman | usman@gmail.com | 1234 |
| 2 | Muhammad Arslan | arslan@gmail.com | 1234 |

Activity 3:

In this activity, we will implement the concept of template inheritance in Laravel Framework.

Template Inheritance

Blade provides a structure for inheritance that allows views to extend, modify, and include other views.

To get started, let's take a look at a simple example. First, we will examine a page layout. Since most web applications maintain the same general layout across various pages, it's convenient to define this layout as a single Blade view:

```

<!-- resources/views/layouts/app.blade.php -->

<html>

  <head>

    <title>App Name - @yield('title')</title>

  </head>

  <body>

    <div class="container">

      @yield('content')

    </div>

  </body>

</html>

```

As you can see, this file contains typical HTML mark-up. However, take note of the `@yield` directives. The `@yield` directive is used to display the contents of a given section.

Now that we have defined a layout for our application, let's define a child page that inherits the layout.

Extending A Layout

When defining a child view, use the `@extends` Blade directive to specify which layout the child view should "inherit". Views which extend a Blade layout may inject content into the layout's sections using `@section` directives. Remember, as seen in the example above, the contents of these sections will be displayed in the layout using `@yield`:

```

<!-- resources/views/child.blade.php -->

@extends('layouts.app')

@section('title', 'Page Title')

@section('content')

  <p>This is my body content.</p>

@endsection

```

The `@yield` directive also accepts a default value as its second parameter. This value will be rendered if the section being yielded is undefined:

```
@yield('content', 'Default content')
```

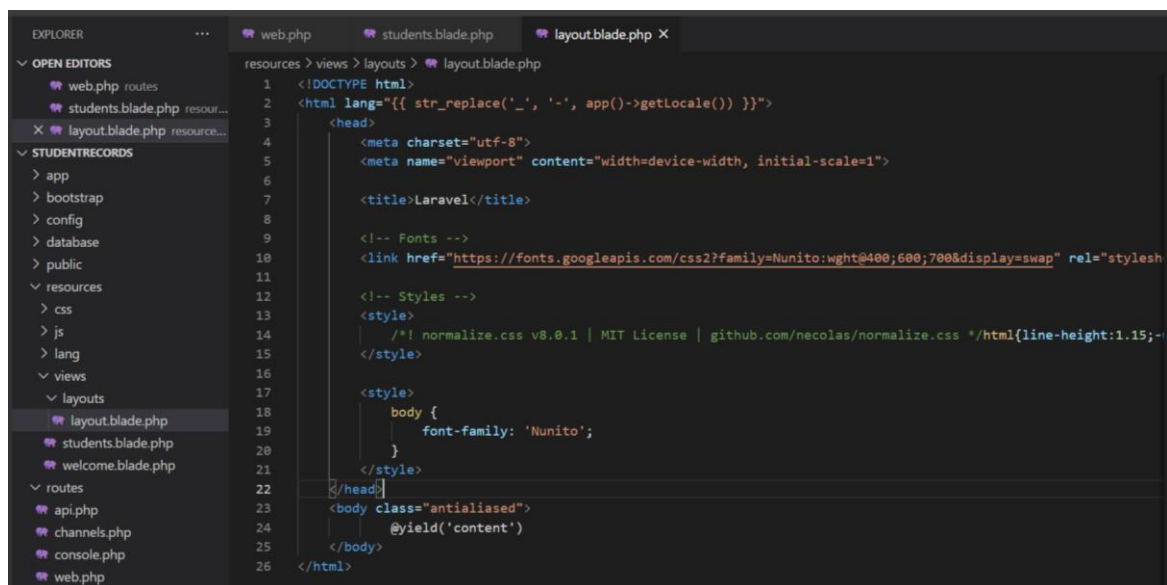
Using Template Inheritance in project ‘studentrecords’:

In project ‘studentrecords’, currently we have got following two views corresponding to two different routes:

- `welcome.blade.php`
- `students.blade.php`

If we look at the code of these template files, they both have the same header and footer parts. It would be better if we can create a generic parent layout which consists of header and footer parts. This generic layout can then be used in both the child layouts using the concept of template inheritance. It will avoid the repetition of code and will help in easily maintaining the code for layouts. The following steps are followed for template inheritance in project ‘studentrecords’:

1. Create a new parent template named, `layout.blade.php` and place header and footer parts of the layout in that file.



```
resources > views > layouts > layout.blade.php
1 <!DOCTYPE html>
2 <html lang="{{ str_replace('_', '-', app()->getLocale()) }}">
3 <head>
4 <meta charset="utf-8">
5 <meta name="viewport" content="width=device-width, initial-scale=1">
6 <title>Laravel</title>
7 <!-- Fonts -->
8 <link href="https://fonts.googleapis.com/css?family=Nunito:wght@400;600;700&display=swap" rel="stylesheet">
9 <!-- Styles -->
10 <style>
11 <!-- normalize.css v8.0.1 | MIT License | github.com/necolas/normalize.css */html{line-height:1.15;-
12 </style>
13 <style>
14 body {
15 font-family: 'Nunito';
16 }
17 </style>
18 </head>
19 <body class="antialiased">
20 @yield('content')
21 </body>
22 </html>
```

2. In the child templates ‘`welcome.blade.php`’ and ‘`students.blade.php`’, use the `@extends` blade directive to specify which layout the child view should "inherit". Child views which extend a blade layout may inject content into the layout's sections using `@section` directives.


```

EXPLORER
web.php
students.blade.php
layout.blade.php

resources > views > students.blade.php
1 @extends('layouts.layout')
2
3
4 @section('content')
5 <div class="flex justify-center min-h-screen sm:items-center sm:pt-0">
6 <table class="title border=1 align="center">
7 <tr><th bgcolor="#666699" colspan="4">Students List</th></tr>
8 <tr>
9 <th bgcolor="#666699">S.No</th>
10 <th bgcolor="#666699">Name</th>
11 <th bgcolor="#666699">Email</th>
12 <th bgcolor="#666699">CNIC</th>
13 </tr>
14
15 @for ($i = 0; $i < count($students); $i++)
16 <tr>
17 <td bgcolor="#6699FF" align="center">{{ $i }}</td>
18 <td bgcolor="#6699FF" width="150" align="center">{{ $students[$i]['name'] }}</td>
19 <td bgcolor="#6699FF" width="150" align="center">{{ $students[$i]['Email'] }}</td>
20 <td bgcolor="#6699FF" width="150" align="center">{{ $students[$i]['CNIC'] }}</td>
21 </tr>
22 @endfor
23 </table>
24 </div>
25
26
27 @endsection

```

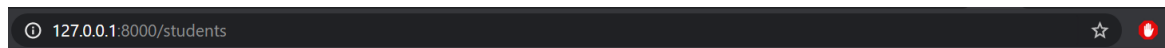
3. Add a `@yield` directive in the parent template `layout.blade.php` to display the contents of a given section.

```

4. <body class="antialiased">
5.     @yield('content')
6. </body>

```

7. Preview of the Web page in the Web browser should look like this:



| Students List | | | |
|---------------|-----------------|------------------|------|
| S.No | Name | Email | CNIC |
| 0 | Muhammad Ali | ali@gmail.com | 1234 |
| 1 | Muhammad Usman | usman@gmail.com | 1234 |
| 2 | Muhammad Arslan | arslan@gmail.com | 1234 |

4) Stage V (verify)

Home Activities:

Activity 1:

Extend the work completed in the Lab to add both internal and external CSS styles in the ongoing project 'studentsproject'.

5) Stage a2 (assess)

Assignment:

Create layouts for company Web application with following pages using Blade template language of the Laravel Framework.

1. Home
2. Profile
3. Clients
4. Products
5. Contact Us

Statement Purpose:

To familiarize students with the following concepts using Laravel Framework:

- Database Connectivity.
- Data Insertion.
- File Uploading.
- Data Retrieval.

Activity Outcomes:

After this lab, the students should be able to store and retrieve information from MySQL database using Laravel Framework.

3) Stage J (Journey)

Introduction

Almost every modern web application interacts with a database. Laravel makes interacting with databases extremely simple across a variety of supported databases using raw SQL, a fluent query builder, and the Eloquent ORM. Currently, Laravel provides first-party support for four databases:

- MySQL 5.6+ (Version Policy)
- PostgreSQL 9.4+ (Version Policy)
- SQLite 3.8.8+
- SQL Server 2017+ (Version Policy)

Database Connection

In a fresh Laravel installation, the `root` directory of your application will contain a `.env.example` file that defines many common environment variables. During the Laravel installation process, this file will automatically be copied to `.env`.

Laravel's default `.env` file contains some common configuration values that may differ based on whether your application is running locally or on a production web server. These values are then retrieved from various Laravel configuration files within the `config` directory using Laravel's `env` function.

The .env file:

Mention the name of the already created MySQL database as the value of the `DB_DATABASE` variable. In the screenshot below, we have specified 'testdb' as the value of the environment variable.

```
EXPLORER
...
.env
x
OPEN EDITORS
x .env
STUDENTRECORDS
view.php
database
public
resources
routes
storage
tests
vendor
.editorconfig
.env
.env.example
.gitattributes
.gitignore
.styleci.yml
artisan
composer.json
composer.lock
package.json
phpunit.xml
README.md
server.php
webpack.mix.js
OUTLINE

.env
1 APP_NAME=Laravel
2 APP_ENV=local
3 APP_KEY=base64:+PKmD/RGNMqHV4KMjenU7Y3AdX11vZUqezm1z1hrvMU=
4 APP_DEBUG=true
5 APP_URL=http://studentrecords.test
6
7 LOG_CHANNEL=stack
8 LOG_LEVEL=debug
9
10 DB_CONNECTION=mysql
11 DB_HOST=127.0.0.1
12 DB_PORT=3306
13 DB_DATABASE=testdb
14 DB_USERNAME=root
15 DB_PASSWORD=
16
17 BROADCAST_DRIVER=log
18 CACHE_DRIVER=file
19 QUEUE_CONNECTION=sync
20 SESSION_DRIVER=file
21 SESSION_LIFETIME=120
22
23 MEMCACHED_HOST=127.0.0.1
24
25 REDIS_HOST=127.0.0.1
26 REDIS_PASSWORD=null
27 REDIS_PORT=6379
28
29 MAIL_MAILER=smt
30 MAIL_HOST=mailhog
```

Once you have configured your database connection, you may run queries using the `DB` facade. The `DB` facade provides methods for each type of query: `select`, `update`, `insert`, `delete`, and `statement`.

Running a Select Query

To run a basic `SELECT` query, you may use the `select` method on the `DB` facade:

```
<?php

namespace App\Http\Controllers;

use App\Http\Controllers\Controller;

use Illuminate\Support\Facades\DB;

class UserController extends Controller
{
    /**
     * Show a list of all of the application's users.
     *
     * @return \Illuminate\Http\Response
     */

    public function index()
```

```

    {
        $users = DB::select('select * from users where active = ?', [1]);
        return view('user.index', ['users' => $users]);
    }
}

```

The first argument passed to the `select` method is the SQL query, while the second argument is any parameter bindings that need to be bound to the query. Typically, these are the values of the `where` clause constraints. Parameter binding provides protection against SQL injection.

The `select` method will always return an `array` of results. Each result within the array will be a PHP `stdClass` object representing a record from the database:

```

use Illuminate\Support\Facades\DB;

$users = DB::select('select * from users');

foreach ($users as $user) {
    echo $user->name;
}

```

Using Named Bindings

Instead of using `?` to represent your parameter bindings, you may execute a query using named bindings:

```

$results = DB::select('select * from users where id = :id', ['id' => 1]);

```

Running an Insert Statement

To execute an `insert` statement, you may use the `insert` method on the `DB` facade. Like `select`, this method accepts the SQL query as its first argument and bindings as its second argument:

```

use Illuminate\Support\Facades\DB;

DB::insert('insert into users (id, name) values (?, ?)', [1, 'Marc']);

```

File uploading using Laravel:

Sometimes, we have to get input from a user in form of a file. Usually, the attached file is handled in two ways:

1. uploading the file to the server while storing its reference in the database and
2. Saving the file in the database.

In this lab, we are going to follow first approach for uploading the file. To upload the file to the server we use the following function:

```
bool move (string $destination, string $filename)
```

```
$pic = $request->file('pic');  
$picName = $pic->getClientOriginalName();  
$pic->move('uploads',$picName);
```

4) Stage **a1** (apply)


Lab Activities:

Activity 1:

Suppose the students uses the following form to get registered with your website. Create a view having a Web Form which collects information from the user and adds that information in the MySQL database 'testdb' using the Laravel Framework.

In the database "testdb", create a table "users" with columns names user_Id, user_Name, user_Email, user_CNIC, user_Comments and user_Picture. Write a PHP code that:

1. Uploads the picture to the server (in uploads folder).
2. Inserts the user record in the database.



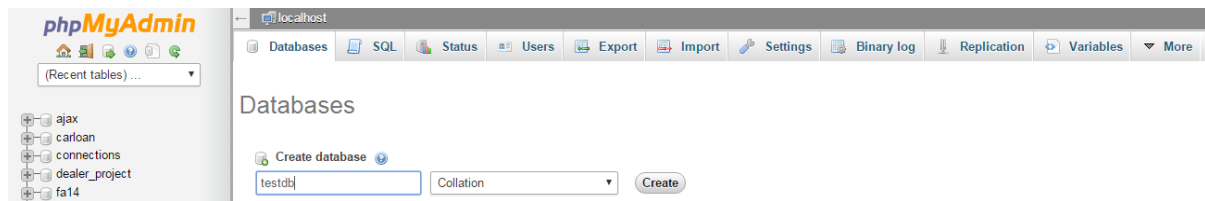
The image shows a web form titled "User Registration Form" with a blue background. It contains the following fields and controls:

- First Name * (text input)
- Last Name * (text input)
- Email Address * (text input)
- CNIC No. * (text input)
- Telephone Number (text input)
- Your Picture (file upload control with "Choose File" and "No file chosen" buttons)
- Comments on Your Self * (text area)
- Submit (button)

Solution:

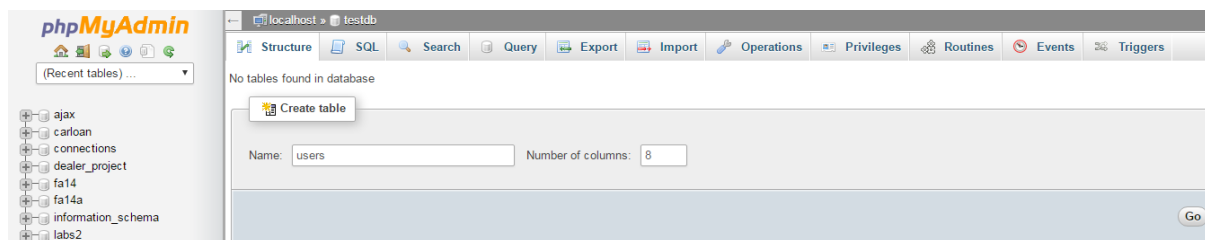
Step 1:

Go to phpMyAdmin and create the database `testdb`:



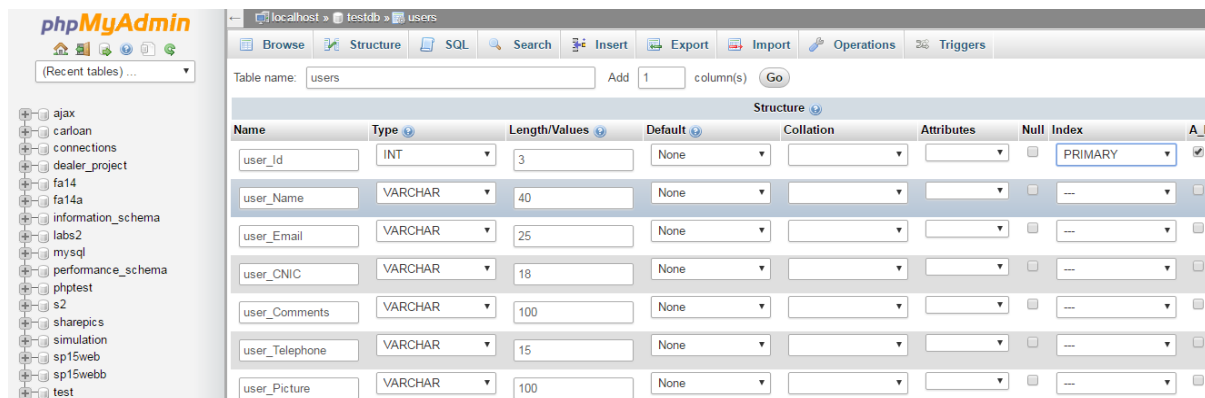
Step 2:

Create the table `users`:



Step 3:

Define the columns:



Step 1:

Create a new template with the title `'create.blade.php'`. Write the following code in the `'view/students/create.blade.php'` file.

```
@extends('layouts.layout')

@section('content')

    <form action="/students/create" method="POST" enctype="multipart/form-data">
        @csrf
        <table width="450px" border="0" bgcolor="#6699FF" align="center">
            <tr>
                <th valign="top" colspan="2" bgcolor="#666699">
```

```

        User Registration Form
    </th>
</tr>
<tr>
    <td valign="top">
        <label for="first_name">First Name *</label>
    </td>
    <td valign="top">
        <input type="text" name="first_name" maxlength="50" size="30">
    </td>
</tr>
<tr>
    <td colspan="2" rowspan="2">
        <label for="last_name">Last Name *</label>
    </td>
    <td colspan="2" rowspan="2">
        <input type="text" name="last_name" maxlength="50" size="30">
    </td>
</tr>
<tr>
</tr>
<tr>
    <td colspan="2" rowspan="2">
        <label for="email">Email Address *</label>
    </td>
    <td colspan="2" rowspan="2">
        <input type="text" name="email" maxlength="80" size="30">
    </td>
</tr>
<tr>
</tr>
<tr>
    <td colspan="2" rowspan="2">
        <label for="first_name">CNIC No. *</label>
    </td>
    <td colspan="2" rowspan="2">
        <input type="text" name="cnic" maxlength="50" size="30">
    </td>
</tr>
<tr>
</tr>
<tr>
    <td colspan="2" rowspan="2">
        <label for="telephone">Telephone Number</label>
    </td>
    <td colspan="2" rowspan="2">
        <input type="text" name="telephone" maxlength="30" size="30">
    </td>
</tr>
<tr>
</tr>
<tr>
    <td colspan="2" rowspan="2">
        <label for="telephone">Your Picture</label>
    </td>
    <td colspan="2" rowspan="2">
        <input type="file" name="pic" maxlength="30" size="30">
    </td>
</tr>
<tr>
</tr>

```



```

        </tr>
        <tr>
            <td valign="top">
                <label for="comments">Comments on Your Self *</label>
            </td>
            <td valign="top">
                <textarea name="comments" maxlength="1000" cols="25" rows="6">
</textarea>
            </td>
        </tr>
        <tr>
            <td colspan="2" style="text-align:center">
                <input type="submit" value="Submit">
            </td>
        </tr>
    </table>
</form>
@endsection

```

Step 2:

Define the following route in the file 'routes/web.php'

```
Route::post('/students/create', 'App\Http\Controllers\StudentController@store');
```

Step 3:

Write the following PHP function in the file

'app/Http/Controllers/StudentController.php'

```

public function store(Request $request) {
    $fname = $request->input('first_name');
    $lname = $request->input('last_name');
    $name = $fname." ".$lname;
    $email = $request->input('email');
    $cnic = $request->input('cnic');
    $tel = $request->input('telephone');
    $comments = $request->input('comments');

    $pic = $request->file('pic');
    $picName = $pic->getClientOriginalName();
    $picType = $pic->getClientOriginalExtension();
    $picSize = $pic->getSize();

    $pic->move('uploads',$picName);
    $destination = 'uploads/'.$picName;
}

```

```

DB::unprepared("insert into users (user_name, user_Email, user_CNIC, user_Comments, user_Telephone, user_Picture) values ('$name','$email','$cnic','$comments','$tel','$destination')");
return redirect('/students/create');
}

```

Step 4:

Visit the following URL to insert record in database.

<http://127.0.0.1:8000/students/create>

Activity 2:

Create a new view that retrieves information from the MySQL database and displays that information on the Web Page.

| Name | Email | CNIC | Comments | Telephone | Photo | Delete | Edit |
|-------------|----------------|------------------|--|--------------|-------|--------|------|
| Asad Asad | asad@gmail.com | 12345-1234567-1 | This is updated sample comment. | 0321-5579634 | | DELETE | EDIT |
| Adeel Ahmed | adeel@gmai.com | 12345-12334567-1 | I am newly registered student n COMSATS. | 0301-5237896 | | DELETE | EDIT |

Solution:

Step 1:

Define the following route in the file 'routes/web.php'

```
Route::get('/index', 'App\Http\Controllers\StudentController@index');
```

Step 2:

Write the following PHP function in the file

'app/Http/Controllers/StudentController.php'

```

public function index() {
    $students = DB::select("select * from users");
    return view('students.index', ['students' => $students]);
}

```

Step 3:

Create a new template with the title 'index.blade.php'. Write the following code in the 'view/students/index.blade.php' file.

```

@extends('layouts.layout')

@section('content')
    <table border="1" align="center">
        <tr>
            <th bgcolor="#666699">Name</th>
            <th bgcolor="#666699">Email</th>
            <th bgcolor="#666699">CNIC</th>
            <th bgcolor="#666699">Comments</th>
            <th bgcolor="#666699">Telephone</th>
            <th bgcolor="#666699">Photo</th>
            <th bgcolor="#666699">Delete</th>
            <th bgcolor="#666699">Edit</th>
        </tr>
        @foreach ($students as $student)
            <tr>
                <td bgcolor="#6699FF" width="150" align="center">{{ $student-
>user_Name }}</td>
                <td bgcolor="#6699FF" width="200" align="center">{{ $student-
>user_Email }}</td>
                <td bgcolor="#6699FF" width="200" align="center">{{ $student-
>user_CNIC }}</td>
                <td bgcolor="#6699FF" width="300" align="center">{{ $student-
>user_Comments }}</td>
                <td bgcolor="#6699FF" width="100" align="center">{{ $student-
>user_Telephone }}</td>
                <td bgcolor="#6699FF" width="100" align="center"><img src = {{ $st
udent->user_Picture }}></td>
                <td bgcolor="#6699FF" width="100" align="center">
                <a href="delete/{{ $student-
>user_id }}" onclick="return confirm('Do you really want to delete this record
?')">DELETE</a></td>
                <td bgcolor="#6699FF" width="100" align="center">
                <a href="update/{{ $student-
>user_id }}" onclick="return confirm('Do you really want to update this record
?')">EDIT</a></td>
            </tr>
        @endforeach
    </table>
@endsection

```

Step 4:

Opening the following URL <http://127.0.0.1:8000/index> will display the Web Page carrying information about all the student records.

| Name | Email | CNIC | Comments | Telephone | Photo | Delete | Edit |
|-------------|-----------------|------------------|--|--------------|---|--------|------|
| Asad Asad | asad@gmail.com | 12345-1234567-1 | This is updated sample comment. | 0321-5579634 |  | DELETE | EDIT |
| Adeel Ahmed | adeel@gmail.com | 12345-12334567-1 | I am newly registered student n COMSATS. | 0301-5237896 |  | DELETE | EDIT |

5) Stage V (verify)

Home Activities:

Activity 1:

Create the following form with the search field. Write a PHP script that searches and displays users with the same email address as entered by the user.

127.0.0.1:8000/search

User Search Form

search *

6) Stage a2 (assess)

Assignment:

Create Web Forms for the following Web Pages for collecting user information using the Laravel Framework.

1. Profile
2. Contact Us

Statement Purpose:

To familiarize students with the update and delete CRUD Operations in MySQL using Laravel Framework.

Activity Outcomes:

After this lab, the students should be able to delete and update records in MySQL using Laravel Framework.

1) Stage J (Journey)

Introduction

Running an Update Statement

The `update` method should be used to update existing records in the database. The number of rows affected by the statement is returned by the method:

```
use Illuminate\Support\Facades\DB;

$affected = DB::update('update users set votes = 100 where name = ?', ['Anita']);
```

Running a Delete Statement

The `delete` method should be used to delete records from the database. Like `update`, the number of rows affected will be returned by the method:

```
use Illuminate\Support\Facades\DB;



$deleted = DB::delete('delete from users');
```

2) Stage a1 (apply)

Lab Activities:

Activity 1:

Write the code that retrieves all of the records added in the table `users` in the `testdb` database. Add another column in each row that displays the delete option to the users (as shown in the following Figure). When the user clicks on the delete link a confirm box should open and after confirmation that row should have been deleted from the table.

| Name | Email | CNIC | Comments | Telephone | Photo | Delete | Edit |
|--------------|-----------------|------------------|--|--------------|---|--------|------|
| Asad Mehmood | asad@gmail.com | 12345-1234567-1 | This is sample comment. | 0321-5579634 |  | DELETE | EDIT |
| Adeel Ahmed | adeel@gmail.com | 12345-12334567-1 | I am newly registered student n COMSATS. | 0301-5237896 |  | DELETE | EDIT |

Solution:

Step 1:

Create a new template with the title 'index.blade.php'. Write the following code in the 'view/students/index.blade.php' file.

```
@extends('layouts.layout')

@section('content')
    <table border="1" align="center">
        <tr>
            <th bgcolor="#666699">Name</th>
            <th bgcolor="#666699">Email</th>
            <th bgcolor="#666699">CNIC</th>
            <th bgcolor="#666699">Comments</th>
            <th bgcolor="#666699">Telephone</th>
            <th bgcolor="#666699">Photo</th>
            <th bgcolor="#666699">Delete</th>
            <th bgcolor="#666699">Edit</th>
        </tr>
        @foreach ($students as $student)
            <tr>
                <td bgcolor="#6699FF" width="150" align="center">{{ $student->user_Name }}</td>
                <td bgcolor="#6699FF" width="200" align="center">{{ $student->user_Email }}</td>
                <td bgcolor="#6699FF" width="200" align="center">{{ $student->user_CNIC }}</td>
                <td bgcolor="#6699FF" width="300" align="center">{{ $student->user_Comments }}</td>
                <td bgcolor="#6699FF" width="100" align="center">{{ $student->user_Telephone }}</td>
                <td bgcolor="#6699FF" width="100" align="center"><img src = {{ $student->user_Picture }}></td>
                <td bgcolor="#6699FF" width="100" align="center">
                    <a href="delete/{{ $student->user_id }}" onclick="return confirm('Do you really want to delete this record?')">DELETE</a></td>
                <td bgcolor="#6699FF" width="100" align="center">
```

```

        <a href="update/{ { $student-
>user_id } }" onclick="return confirm('Do you really want to update this record
?')">EDIT</a></td>
        </tr>
    @endforeach
</table>
@endsection

```

Step 2:

Define the following route in the file `routes/web.php` for the user click at delete link within the data table.

```
Route::get('delete/{id}', 'App\Http\Controllers\StudentController@destroy');
```

Step 3:

Write the following PHP function in the file `app/Http/Controllers/StudentController.php`

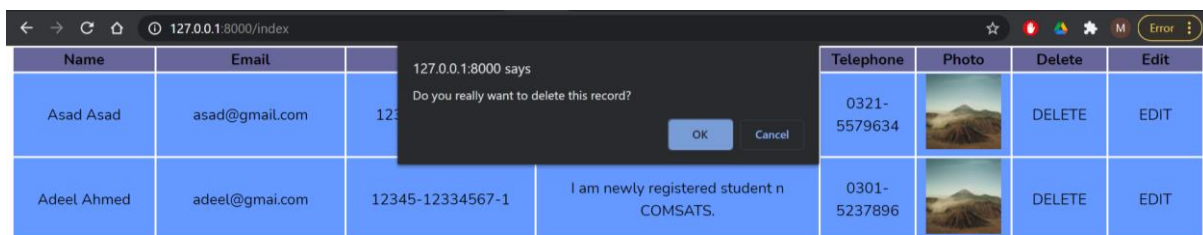
```

public function destroy($id) {
    DB::unprepared("delete from users where user_id = '$id'");
    return redirect('/index');
}

```

Step 4:

Open the following URL: <http://127.0.0.1:8000/index>. Click on the corresponding DELETE link of the row to delete the record. The following confirm dialog box will be displayed.



Step 5:

When the user clicks on the OK button, the record will be deleted from the database and the updated Web Page will be displayed to the user.



Activity 2:

When the user clicks on the EDIT link, another form displaying the existing record should open. When user submits this form after updating the values; the new values should replace the existing ones in the database.

Solution:

Step 1:

When the user clicks on the Update link in the `index.blade.php` (See [Step 1 of Activity 1](#)), the following route should be activated.

```
Route::get('update/{id}', 'App\Http\Controllers\StudentController@update');
```

Step 2:

Write the following PHP function in the file

`'app/Http/Controllers/StudentController.php'`

```
public function update($id) {
    $users = DB::select("select * from users where user_id = ?",[$id]);
    return view('students.update',['users' => $users]);
}
```

Step 3:

Create a new template with the title `'update.blade.php'`. Write the following code in the `'view/students/update.blade.php'` file.

```
@extends('layouts.layout')

@section('content')
<form action="/students/update/{{ $users[0]-
>user_id }}" method="POST" enctype="multipart/form-data">
    @csrf
    <table width="450px" border="0" bgcolor="#6699FF" align="center">
        <tr>
            <th valign="top" colspan="2" bgcolor="#666699">
                User Registration Form
            </th>
        </tr>
        <tr>
            <td valign="top">
                <label for="first_name">First Name *</label>
            </td>
            <td valign="top">
```



```

        <input type="text" name="first_name" maxlength="50" size="
30" value = {{ $users[0]->user_Name }}>
    </td>
</tr>
<tr>
    <td valign="top">
        <label for="last_name">Last Name *</label>
    </td>
    <td valign="top">
        <input type="text" name="last_name" maxlength="50" size="3
0" value ={{ $users[0]->user_Name }}>
    </td>
</tr>
<tr>
    <td valign="top">
        <label for="email">Email Address *</label>
    </td>
    <td valign="top">
        <input type="text" name="email" maxlength="80" size="30" v
alue ={{ $users[0]->user_Email }}>
    </td>
</tr>
<tr>
    <td valign="top">
        <label for="first_name">CNIC No. *</label>
    </td>
    <td valign="top">
        <input type="text" name="cnic" maxlength="50" size="30" va
lue ={{ $users[0]->user_CNIC}}>
    </td>
</tr>
<tr>
    <td valign="top">
        <label for="telephone">Telephone Number</label>
    </td>
    <td valign="top">
        <input type="text" name="telephone" maxlength="30" size="3
0" value ={{ $users[0]->user_Telephone}}>
    </td>
</tr>
<tr>
    <td valign="top">
        <label for="telephone">Your Picture</label>
    </td>
    <td valign="top">
        <input type="file" name="pic" maxlength="30" size="30">
    </td>
</tr>
<tr>
    <td valign="top">

```

```

        <label for="comments">Comments on Your Self *</label>
    </td>
    <td valign="top">
        <textarea name="comments" maxlength="1000" cols="25" rows=
"6">{{ $users[0]->user_Comments }}</textarea>
    </td>
</tr>
<tr>
    <td colspan="2" style="text-align:center">
        <input type="submit" value="Update">
    </td>
</tr>
</table>
</form>
@endsection

```

Step 4:

Define the following routes in the file 'routes/web.php'

```
Route::post('/students/update/{id}', 'App\Http\Controllers\StudentController@display');
```

Write the following PHP function in the file

'app/Http/Controllers/StudentController.php'

```

public function display(Request $request, $id){
    $fname = $request->input('first_name');
    $lname = $request->input('last_name');
    $name = $fname." ".$lname;
    $email = $request->input('email');
    $cnic = $request->input('cnic');
    $tel = $request->input('telephone');
    $comments = $request->input('comments');

    $pic = $request->file('pic');
    $picName = $pic->getClientOriginalName();
    $picType = $pic->getClientOriginalExtension();
    $picSize = $pic->getSize();

    $pic->move('uploads',$picName);
    $destination = 'uploads/'.$picName;

    DB::update("update table_users set user_Name=?, user_Email=?, user
_CNIC=?, user_Comments=?, user_Telephone=?, user_Picture=? where user_id = ?",
[$name, $email, $cnic, $comments, $tel, $destination, $id]);

```

```

return redirect('index');
}

```

Step 5:

Open the following URL: <http://127.0.0.1:8000/index>.

| Name | Email | CNIC | Comments | Telephone | Photo | Delete | Edit |
|-------------|-----------------|------------------|--|--------------|-------|--------|------|
| Adeel Ahmed | adeel@gmail.com | 12345-12334567-1 | I am newly registered student n COMSATS. | 0301-5237896 | | DELETE | EDIT |

Step 6:

- The code for the view `update.blade.php` is provided in the **Step 3**, which will receive the data and display it in the Web Form using Blade directives. Note that all the input fields in the form are filled by retrieving data from the database.

User Registration Form

First Name *

Last Name *

Email Address *

CNIC No. *

Telephone Number

Your Picture No file chosen

Comments on Your Self *

Step 7:

- After updating data, the page will be redirected to <http://127.0.0.1:8000/index>. The following screenshot shows the updated Web Page with the modified data.

| Name | Email | CNIC | Comments | Telephone | Photo | Delete | Edit |
|-------------|-----------------|------------------|---|--------------|-------|--------|------|
| Adeel Hasan | adeel@gmail.com | 12345-12334567-1 | I am newly registered student in COMSATS. DATA IS UPDATED | 0301-5237896 | | DELETE | EDIT |

3) Stage a2 (assess)

Assignment:

Add features for updating and deleting the records of company employees using Laravel Framework.

1. Home
2. Profile
3. Clients
4. Products
5. Contact Us

Statement Purpose:

To familiarize students with the Database Migration Operations in MySQL using Laravel Framework.

Activity Outcomes:

After this lab, the students should be able to create Tables in MySQL Database using Database Migrations in Laravel Framework.

4) Stage J (Journey)

Introduction

Database migrations, also known as schema migrations, database schema migrations, or simply migrations, are controlled sets of changes developed to modify the structure of the objects within a relational database.

Migrations help transition database schemas from their current state to a new desired state, whether that involves adding tables and columns, removing elements, splitting fields, or changing types and constraints.

The Laravel Schema facade provides database agnostic support for creating and manipulating tables across all of Laravel's supported database systems. Typically, migrations will use this facade to create and modify database tables and columns.

Generating Migrations

You may use the `make:migration` **Artisan command** to generate a database migration. The new migration will be placed in your `database/migrations` directory.

Laravel will use the name of the migration to attempt to guess the name of the table and whether or not the migration will be creating a new table. If Laravel is able to determine the table name from the migration name, Laravel will pre-fill the generated migration file with the specified table. Otherwise, you may simply specify the table in the migration file manually.

Migration Structure

A migration class contains two methods: `up` and `down`. The `up` method is used to add new tables, columns, or indexes to your database, while the `down` method should reverse the operations performed by the `up` method.

Within both of these methods, you may use the Laravel schema builder to expressively create and modify tables. To learn about all of the methods available on the Schema builder, check out its documentation. For example, the following migration creates a flights table:

```

<?php

use Illuminate\Database\Migrations\Migration;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Support\Facades\Schema;

class CreateFlightsTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('flights', function (Blueprint $table) {
            $table->id();
            $table->string('name');
            $table->string('airline');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {

```

```
Schema::drop('flights');  
}  
}
```

Running Migrations

To run all of your outstanding migrations, execute the migrate Artisan command:

```
php artisan migrate
```

5) Stage **a1** (apply)

Lab Activities:

Activity 1:

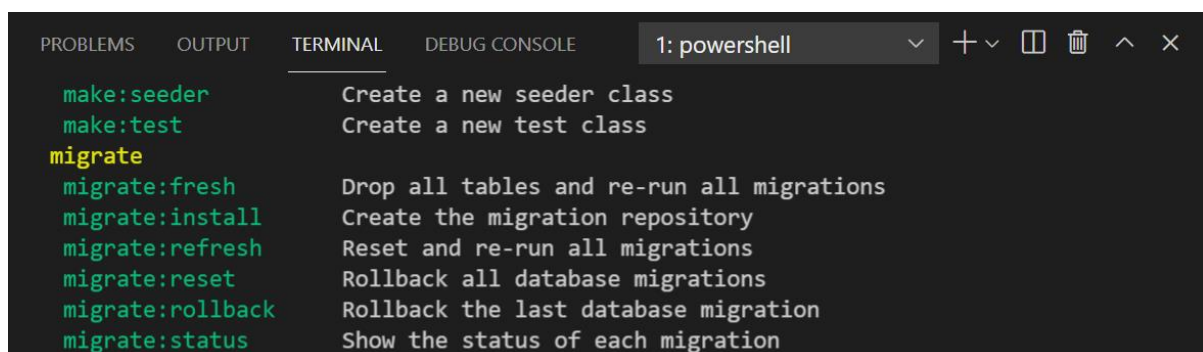
In this activity, students have to understand different commands under the category of migrations.

Solution:

To get the list of commands under the category of migrations, type the following command in the command prompt:

```
php artisan list
```

It will return a list of all available artisan commands. Scroll to the category of migrate. The description of these commands is provided below:



```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE  1: powershell  + v  [ ]  [ ]  ^  x  
make:seeder  Create a new seeder class  
make:test    Create a new test class  
migrate  
migrate:fresh  Drop all tables and re-run all migrations  
migrate:install  Create the migration repository  
migrate:refresh  Reset and re-run all migrations  
migrate:reset   Rollback all database migrations  
migrate:rollback  Rollback the last database migration  
migrate:status  Show the status of each migration
```

1. Rolling Back Migrations

To roll back the latest migration operation, you may use the rollback Artisan command. This command rolls back the last "batch" of migrations, which may include multiple migration files:

```
php artisan migrate:rollback
```

The `migrate:reset` command will roll back all of your application's migrations:

```
php artisan migrate:reset
```

2. Drop All Tables & Migrate

The `migrate:fresh` command will drop all tables from the database and then execute the `migrate` command. It also deletes manually created tables. Be careful while executing this command because it will not regenerate manually generated table due to the absence of migration file for that table.

```
php artisan migrate:fresh
```

3. Roll Back & Migrate Using A Single Command

The `migrate:refresh` command will roll back all of your migrations and then execute the migrate command. This command effectively re-creates your entire database. It will not remove manually added tables in the database.

```
php artisan migrate:refresh
```

Activity 2:

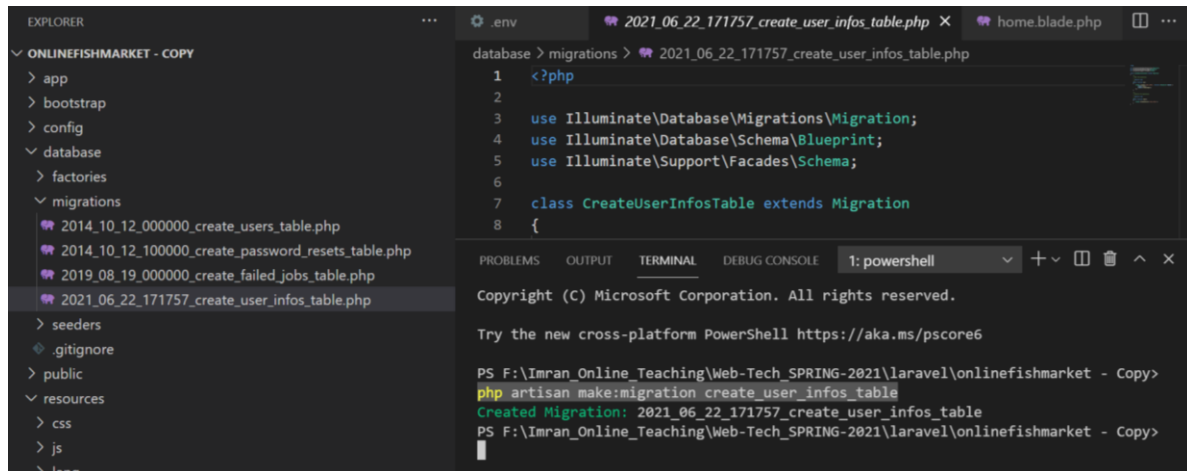
In the project 'onlinefishmarket', create user_infos and details tables in the database fmarket-migration using migrations commands.

Solution:

Step 1:

To create the table user_infos, type the following command in the command prompt:

```
php artisan make:migration create_user_infos_table
```



The screenshot shows the Visual Studio Code interface. On the left, the Explorer pane shows the project structure with the file '2021_06_22_171757_create_user_infos_table.php' selected in the migrations folder. The main editor shows the content of this migration file:

```
1 <?php
2
3 use Illuminate\Database\Migrations\Migration;
4 use Illuminate\Database\Schema\Blueprint;
5 use Illuminate\Support\Facades\Schema;
6
7 class CreateUserInfosTable extends Migration
8 {
```

Below the editor, the Terminal pane shows the command prompt output:

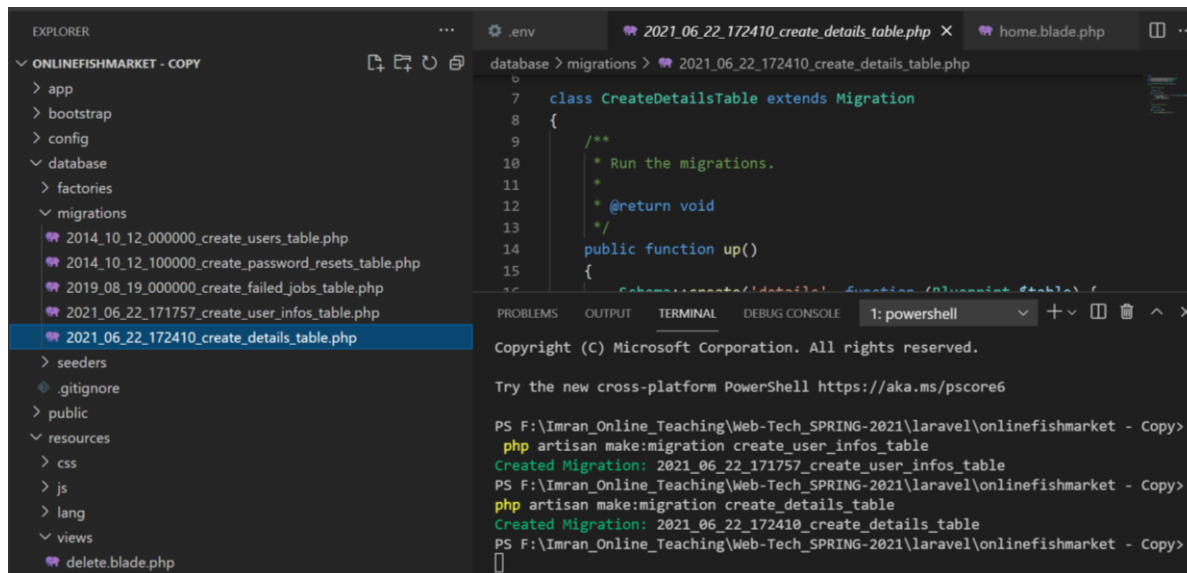
```
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS F:\Imran_Online_Teaching\Web-Tech_SPRING-2021\laravel\onlinefishmarket - Copy>
php artisan make:migration create_user_infos_table
Created Migration: 2021_06_22_171757_create_user_infos_table
PS F:\Imran_Online_Teaching\Web-Tech_SPRING-2021\laravel\onlinefishmarket - Copy>
```

Similarly type the following command for creating details table.

```
php artisan make:migration create_details_table
```



The screenshot shows the Visual Studio Code interface. On the left, the Explorer pane shows the project structure with the file '2021_06_22_172410_create_details_table.php' selected in the migrations folder. The main editor shows the content of this migration file:

```
6
7 class CreateDetailsTable extends Migration
8 {
9
10 /**
11  * Run the migrations.
12  * @return void
13  */
14 public function up()
15 {
```

Below the editor, the Terminal pane shows the command prompt output:

```
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

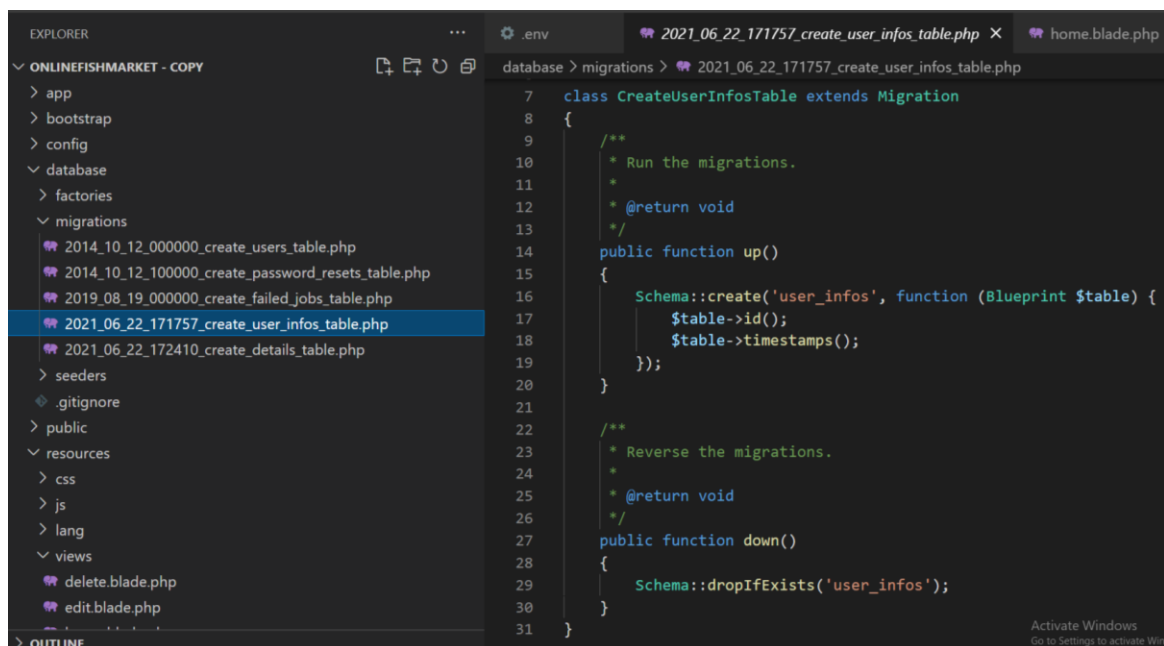
PS F:\Imran_Online_Teaching\Web-Tech_SPRING-2021\laravel\onlinefishmarket - Copy>
php artisan make:migration create_user_infos_table
Created Migration: 2021_06_22_171757_create_user_infos_table
PS F:\Imran_Online_Teaching\Web-Tech_SPRING-2021\laravel\onlinefishmarket - Copy>
php artisan make:migration create_details_table
Created Migration: 2021_06_22_172410_create_details_table
PS F:\Imran_Online_Teaching\Web-Tech_SPRING-2021\laravel\onlinefishmarket - Copy>
```

Step 2:

The created migration files in the **database/migrations/** are the following:

- **2021_06_22_171757_create_user_infos_table.php**
- **2021_06_22_172410_create_details_table.php**

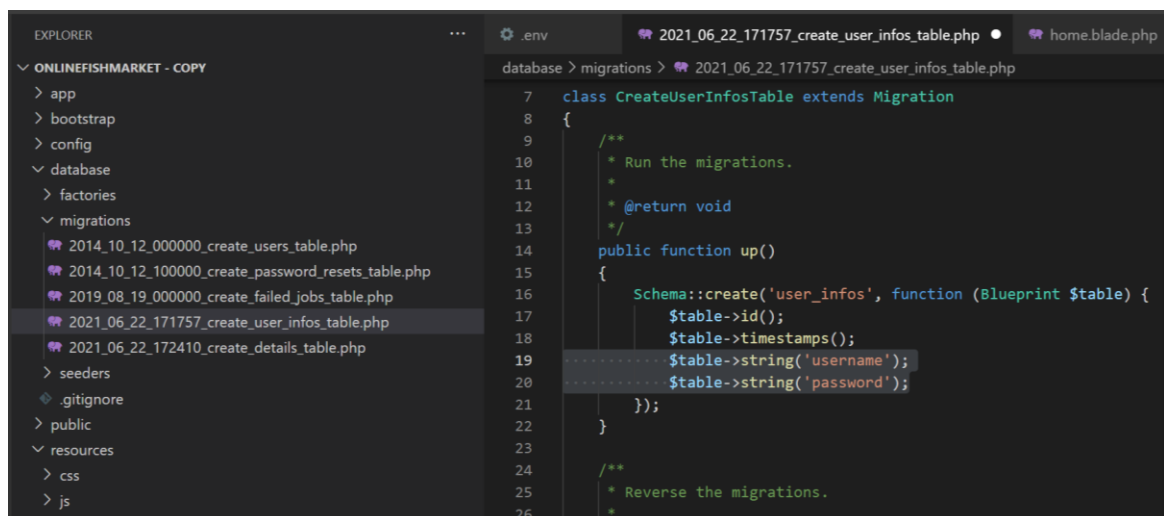
The screenshot below shows that the migration file created by the Laravel contains a boilerplate code with two functions up and down. The up function is responsible for creating the table and down function is used for deleting the table.



```
7 class CreateUserInfosTable extends Migration
8 {
9     /**
10      * Run the migrations.
11      *
12      * @return void
13      */
14     public function up()
15     {
16         Schema::create('user_infos', function (Blueprint $table) {
17             $table->id();
18             $table->timestamps();
19         });
20     }
21
22     /**
23      * Reverse the migrations.
24      *
25      * @return void
26      */
27     public function down()
28     {
29         Schema::dropIfExists('user_infos');
30     }
31 }
```

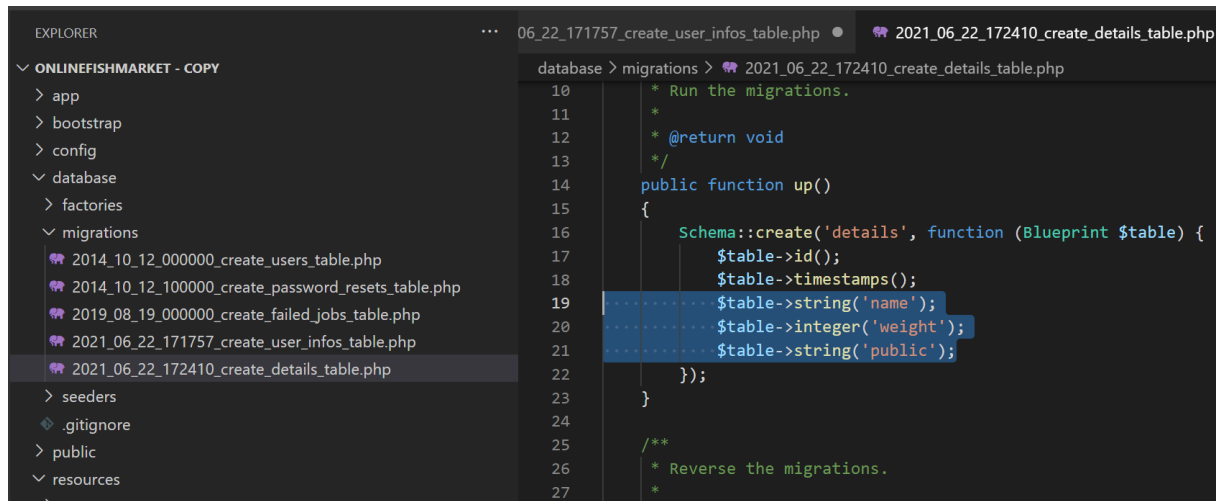
Step 3:

Add two more columns (username and password) in the user_infos table as we needed them for saving usernames and passwords of the user accounts created on our Web application.



```
7 class CreateUserInfosTable extends Migration
8 {
9     /**
10      * Run the migrations.
11      *
12      * @return void
13      */
14     public function up()
15     {
16         Schema::create('user_infos', function (Blueprint $table) {
17             $table->id();
18             $table->timestamps();
19             $table->string('username');
20             $table->string('password');
21         });
22     }
23
24     /**
25      * Reverse the migrations.
26      *
27      * @return void
28      */
29     public function down()
30     {
31         Schema::dropIfExists('user_infos');
32     }
33 }
```

Similarly, add three more columns (name, weight and public) in the details table as we needed them for saving records of the fish data created in our Web application.



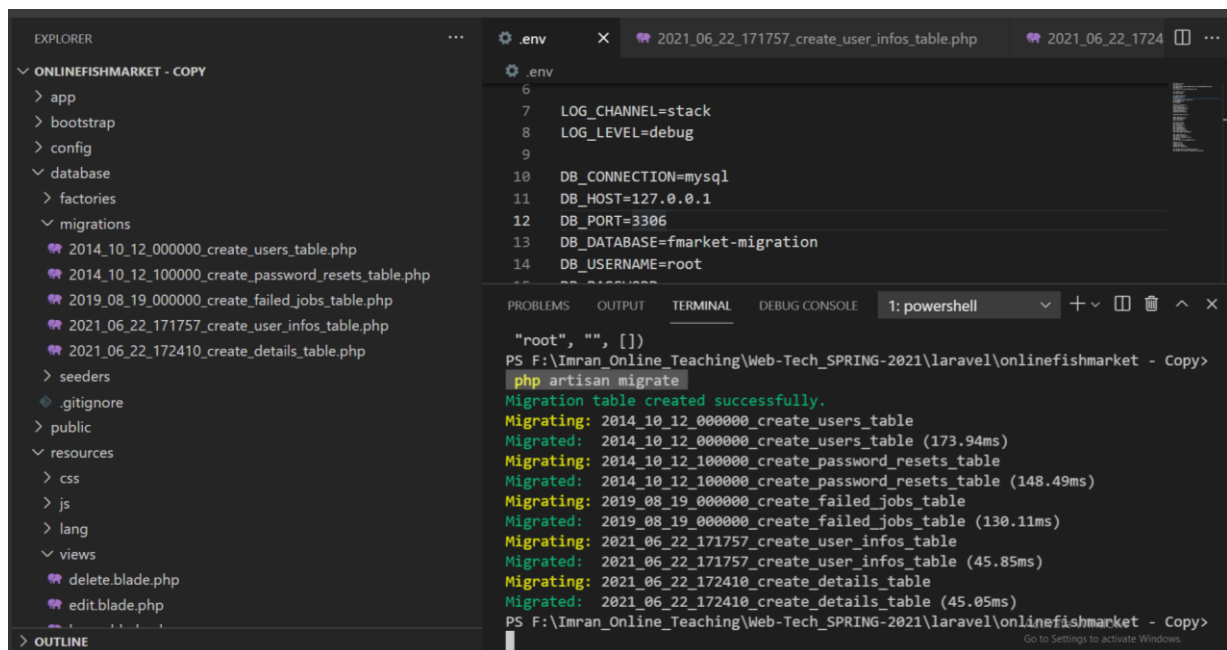
```
EXPLORER    ... 06_22_171757_create_user_infos_table.php    2021_06_22_172410_create_details_table.php
ONLINEFISHMARKET - COPY
  > app
  > bootstrap
  > config
  > database
  > factories
  > migrations
    2014_10_12_000000_create_users_table.php
    2014_10_12_100000_create_password_resets_table.php
    2019_08_19_000000_create_failed_jobs_table.php
    2021_06_22_171757_create_user_infos_table.php
    2021_06_22_172410_create_details_table.php
  > seeders
  > .gitignore
  > public
  > resources
  > ...

database > migrations > 2021_06_22_172410_create_details_table.php
10      * Run the migrations.
11      *
12      * @return void
13      */
14      public function up()
15      {
16          Schema::create('details', function (Blueprint $table) {
17              $table->id();
18              $table->timestamps();
19              $table->string('name');
20              $table->integer('weight');
21              $table->string('public');
22          });
23      }
24
25      /**
26      * Reverse the migrations.
27      *
```

Step 4:

Run the following command for creating the tables in the in the database fmarket-migration.

```
php artisan migrate
```



```
EXPLORER    .env    ... 2021_06_22_171757_create_user_infos_table.php    2021_06_22_172410_create_details_table.php
ONLINEFISHMARKET - COPY
  > app
  > bootstrap
  > config
  > database
  > factories
  > migrations
    2014_10_12_000000_create_users_table.php
    2014_10_12_100000_create_password_resets_table.php
    2019_08_19_000000_create_failed_jobs_table.php
    2021_06_22_171757_create_user_infos_table.php
    2021_06_22_172410_create_details_table.php
  > seeders
  > .gitignore
  > public
  > resources
  > css
  > js
  > lang
  > views
  delete.blade.php
  edit.blade.php
  > OUTLINE

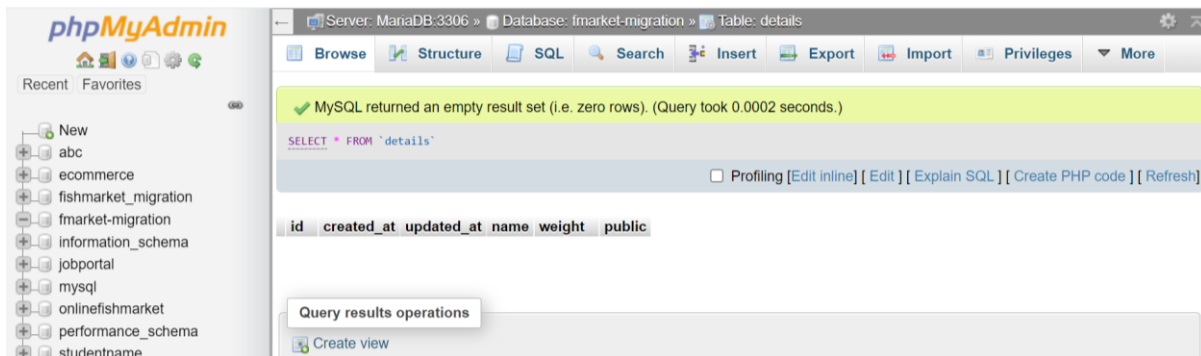
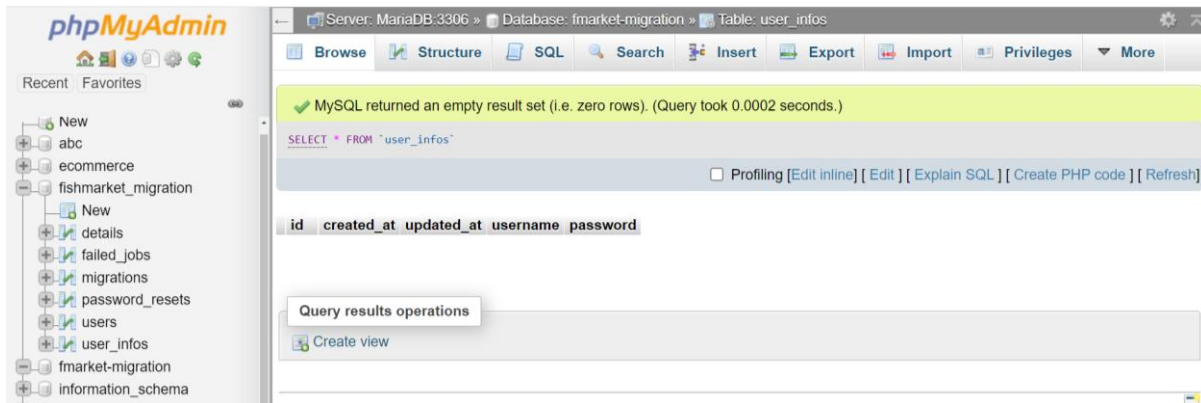
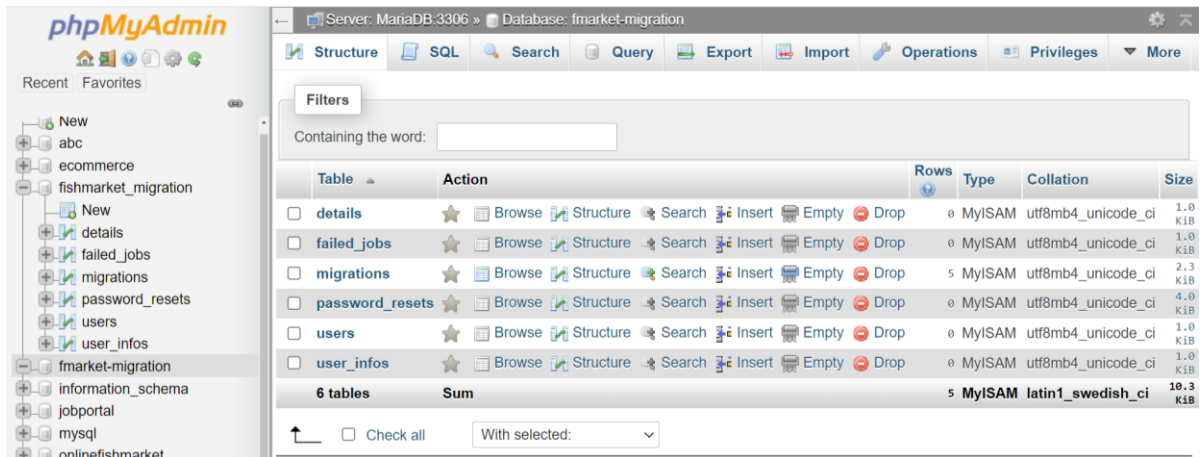
.env
6
7 LOG_CHANNEL=stack
8 LOG_LEVEL=debug
9
10 DB_CONNECTION=mysql
11 DB_HOST=127.0.0.1
12 DB_PORT=3306
13 DB_DATABASE=fmarket-migration
14 DB_USERNAME=root

PROBLEMS    OUTPUT    TERMINAL    DEBUG CONSOLE    1: powershell
PS F:\Imran_Online_Teaching\Web-Tech_SPRING-2021\laravel\onlinefishmarket - Copy>
php artisan migrate
Migration table created successfully.
Migrating: 2014_10_12_000000_create_users_table
Migrated: 2014_10_12_000000_create_users_table (173.94ms)
Migrating: 2014_10_12_100000_create_password_resets_table
Migrated: 2014_10_12_100000_create_password_resets_table (148.49ms)
Migrating: 2019_08_19_000000_create_failed_jobs_table
Migrated: 2019_08_19_000000_create_failed_jobs_table (130.11ms)
Migrating: 2021_06_22_171757_create_user_infos_table
Migrated: 2021_06_22_171757_create_user_infos_table (45.85ms)
Migrating: 2021_06_22_172410_create_details_table
Migrated: 2021_06_22_172410_create_details_table (45.05ms)
PS F:\Imran_Online_Teaching\Web-Tech_SPRING-2021\laravel\onlinefishmarket - Copy>
```

Step 5:

Switch to the phpMyAdmin interface using the following URL, and view the changes made by the above command.

```
http://localhost/phpmyadmin/
```



Activity 3:

In the 'onlinefishmarket' project, create Models for interacting with the database using Eloquent ORM commands.

Solution:

Eloquent

Laravel includes Eloquent, an object-relational mapper (ORM) that makes it enjoyable to interact with your database. When using Eloquent, each database table has a corresponding "Model" that is used to interact with that table. In addition to retrieving records from the

database table, Eloquent models allow you to insert, update, and delete records from the table as well.

Generating Model Classes

To get started, let's create an Eloquent model. Models typically live in the `app/Models` directory and extend the `Illuminate\Database\Eloquent\Model` class. You may use the `make:model` [Artisan command](#) to generate a new model:

```
php artisan make:model Flight
```

Table Names

After glancing at the example above, you may have noticed that we did not tell Eloquent which database table corresponds to our `Flight` model. By convention, the "snake case", plural name of the class will be used as the table name unless another name is explicitly specified. So, in this case, Eloquent will assume the `Flight` model stores records in the `flights` table, while an `AirTrafficController` model would store records in an `air_traffic_controllers` table.

If your model's corresponding database table does not fit this convention, you may manually specify the model's table name by defining a `table` property on the model:

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class Flight extends Model
{
    /**
     * The table associated with the model.
     *
     * @var string
     */
}
```

```
protected $table = 'my_flights';
```

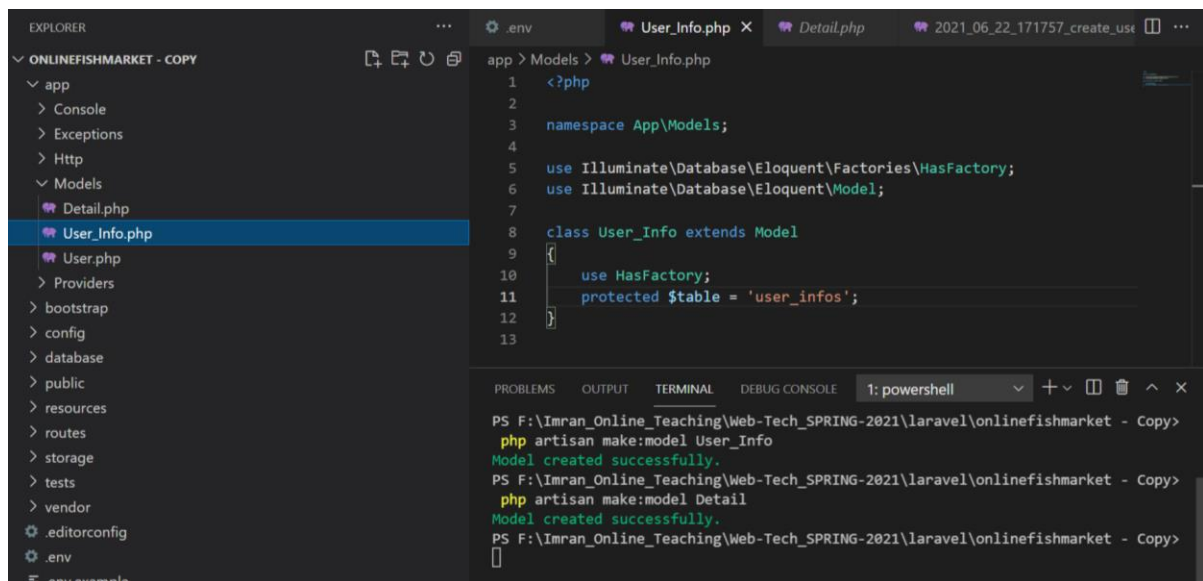
```
}
```

Step 1:

Run the following commands to create models for interacting with the database tables `user_infos` and `details`.

```
php artisan make:model User_Info
```

```
php artisan make:model Detail
```



The screenshot shows an IDE with a file explorer on the left and a code editor on the right. The file explorer shows the project structure, including the `Models` directory. The code editor displays the `User_Info.php` file, which contains the following code:

```
1 <?php
2
3 namespace App\Models;
4
5 use Illuminate\Database\Eloquent\Factories\HasFactory;
6 use Illuminate\Database\Eloquent\Model;
7
8 class User_Info extends Model
9 {
10     use HasFactory;
11     protected $table = 'user_infos';
12 }
13
```

The terminal at the bottom shows the execution of the following commands and their output:

```
PS F:\Imran_Online_Teaching\Web-Tech_SPRING-2021\laravel\onlinefishmarket - Copy>
php artisan make:model User_Info
Model created successfully.
PS F:\Imran_Online_Teaching\Web-Tech_SPRING-2021\laravel\onlinefishmarket - Copy>
php artisan make:model Detail
Model created successfully.
PS F:\Imran_Online_Teaching\Web-Tech_SPRING-2021\laravel\onlinefishmarket - Copy>
```

Step 2:

Replace the queries in the `UserController` class with the Eloquent Model class methods.

```
function store() {

    $user = new User_Info;
    $user->username = Request::input('username');
    $user->password = Request::input('password');

    $user->save();

    //$uname = Request::input('username');
    //$pass = Request::input('password');

    //DB::unprepared("insert into users (username,password) values ('$uname', '$pass')");
    //DB::insert('insert into users (username, password) values (?, ?)', [
    $uname, $pass]);
```

```
    return redirect('index');
}
```

```
function match() {
    $uname = Request::input('username');
    $pass = Request::input('password');

    $loginData = User_Info::where('username', $uname)->get();

    //$loginData = DB::select('select password from users where username =
    ?', [$uname]);

    if (count($loginData) > 0){

        foreach ($loginData as $tablepass) {

            if (($tablepass->password) == $pass){
                return redirect('home');
            }
            else{
                $error='Password does not match';
                return view('login')->with('error',$error);
            }
        }
    }
    //return redirect('login');
}
```

```
function home() {
    //$fishData = DB::select('select * from details');

    $fishData = Detail::all();

    return view('home', ["fishData"=>$fishData]);

}
```

```
function storeFish() {

    //$fishname = Request::input('fishname');
    //$fishweight = Request::input('fishweight');
```

```

$public = Request::input('public');

if ($public == "yes")
    $decision="yes";
else
    $decision="no";

$detail = new Detail;
$detail->name = Request::input('fishname');
$detail->weight = Request::input('fishweight');
$detail->public = $decision;

$detail->save();

//DB::insert('insert into details (name, weight, public,updatedtime) v
alues (?, ?, ?,?)', [$fishname , $fishweight, $decision,null]);
return redirect('home');
}

```

```

function viewFish() {
    $id= request('id');
    // $fishData = DB::select('select * from details where id = ?',[$id]);

    $fishData = Detail::where('id', $id)->get();

    return view('edit', ["fishData"=>$fishData,'id'=>$id]);
}

```

```

function updateFish($id) {

    $fishweight = Request::input('fishweight');
    $public = Request::input('public');

    if ($public == "yes")
        $decision="yes";
    else
        $decision="no";

    date_default_timezone_set("Asia/Karachi");
    $dt = date("Y-m-d H:i:s");

    $detail = Detail::find($id);
    $detail->weight = $fishweight;
}

```



```

    $detail->public = $decision;
    $detail->save();

    //DB::update('update details set weight=?, public=?, updatedtime=? where id= ?', [$fishweight, $decision, $dt , $id]);
    return redirect('home');
}

```

```

function deleteFish() {
    $id= request('id');

    Detail::destroy($id);
    //DB::delete('delete from details where id= ?', [$id]);
    return redirect('home');
}

```

6) Stage **a2** (assess)

Assignment:

Create tables using migrations for the project company employees using Laravel Framework.

6. Home
7. Profile
8. Clients
9. Products
10. Contact Us