Lab Manual

CSC336-Web Technologies

CUI

Department of Computer Science
Islamabad Campus

Lab Contents:

The topics include: HTML; CSS; Javascripts; Server-Side Technologies (PhP); Laravel Framework and use of Database Mysql in Web-based Applications.

Student Outcomes (SO)

S.#	Description
2	Identify, formulate, research literature, and solve complex computing problems reaching substantiated conclusions using fundamental principles of mathematics, computing sciences, and relevant domain disciplines
3	Design and evaluate solutions for complex computing problems, and design and evaluate systems, components, or processes that meet specified needs with appropriate consideration for public health and safety, cultural, societal, and environmental considerations
4	Create, select, adapt and apply appropriate techniques, resources, and modern computing tools to complex computing activities, with an understanding of the limitations
5	Function effectively as an individual and as a member or leader in diverse teams and in multidisciplinary settings.

Intended Learning Outcomes

Sr.#	Description	Blooms Taxonomy Learning Level	SO
CLO -4	Apply the concepts of markup & scripting languages and client-side technologies.	*Applying*	2,4
CLO -5	Develop dynamic applications using current industrial practices.	*Creating*	2-5

Lab Assessment Policy

The lab work done by the student is evaluated using Psycho-motor rubrics defined by the course instructor, viva-voce, project work/performance. Marks distribution is as follows:

Assignments	Lab Mid Term Exam	Lab Terminal Exam	Total
25	25	50	100

Note: Midterm and Final term exams must be computer based.

Table of Contents

Lab 01

HTML Basics

Objective:

- After this lab the students should be able to understand the use of markup language “HTML” and its basic tags.

Activity Outcomes:

The activities provide hands-on practice with the following topics

- Design basic web page using HTML Tags
- Add text formatting tags
- Add lists to web pages
- Add images and videos to the web pages

Instructor Note:

As pre-lab activity, read Chapter 1, 2 from the textbook “Web Design Playground: HTML & CSS the Interactive Way 1st Edition, April 2019 by Paul McFedries”.

1) Useful Concepts

TAG	**Description**
<!DOCTYPE>	Defines the document type
<html>	Defines an HTML document
<head>	Contains metadata/information for the document
<title>	Defines a title for the document
<body>	Defines the document's body
<h1> to <h6>	Defines HTML headings
<p>	Defines a paragraph
 	Inserts a single line break
<hr>	Defines a thematic change in the content
<!--...-->	Defines a comment
<img src=”abc.jpg”>	Inserts an image

<video src=”abc.mp4”>	Inserts Video
	Order list
	Adding list items
	Unoredered list
<dl>	tag defines a description list.
<dt>	defines terms/names
<dd>	describes each term/name
<iframe>	An inline frame is used to embed another document within the current HTML document.

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>
<img src=”abc.jpg”>
<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) Solved Lab Activities

Sr. No	*Allocated Time*	*Level of Complexity*	*CLO Mapping*
Activity 1	*30 Minutes*	*Low*	*CLO-4*
Activity 2	*30 Minutes*	*Low*	*CLO-4*

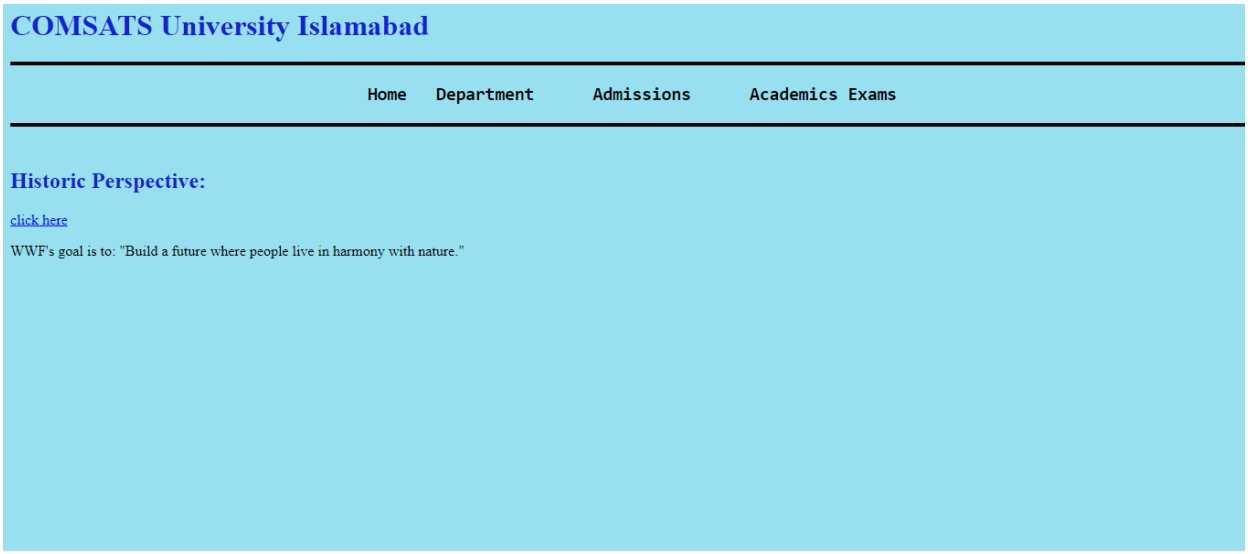
Activity 1

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

Solution

```
<html>
<head>
<title>Home</title>
</head>
<body bgcolor="#98E0F0">
<h1><font color="#1322D6"> COMSATS University Islamabad</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>
```

Output:



Activity 2:

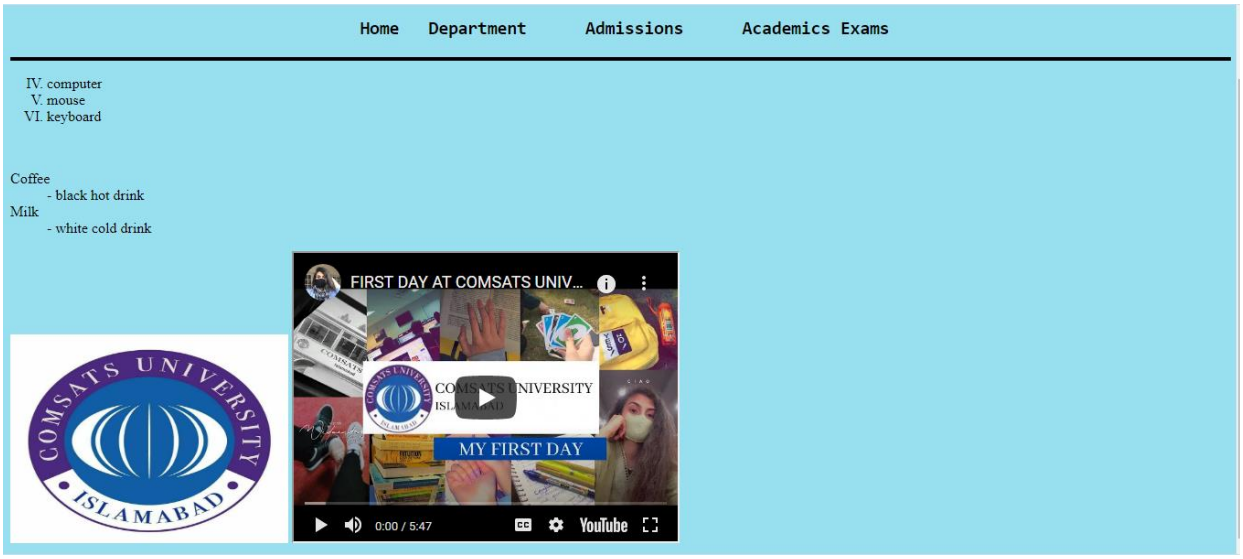
Add list of topics, images and videos to your website

Solution

```
<html>
<head>
<title>Home</title>
</head>
<body bgcolor="#98E0F0">
<h1><font color="#1322D6"> COMSATS University Islamabad </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>
<img src="akweb.jpg" alt="Mountain View farrrr"
```

```
style="width:304px;height:228px;">
<iframe width="420" height="315" src="https://www.youtube.com/embed/
joriayTNmwQ?autoplay=1">
</iframe>
</body> </html>
```

Output



3) Graded Lab Tasks

Note: The instructor can design graded lab activities according to the level of difficult and complexity of the solved lab activities. The lab tasks assigned by the instructor should be evaluated in the same lab.

Lab Task 1

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

Lab Task 2

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

Lab Task 3

Use inline styling to make the webpage created in lab task 2, more aesthetic.

Lab 02

Working with HTML Links and tables

Objective:

After this lab the students should be able to add linking information, creating multiple pages, and navigating in a website and add tables in web pages.

Activity Outcomes:

To familiarize the students with

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

Instructor Note:

As pre-lab activity, read Chapter 3 from the textbook "Web Design Playground: HTML & CSS the Interactive Way 1st Edition, April 2019 by Paul McFedries".

1) Useful Concepts

TAG	**Description**
<a>	The <a> tag defines a hyperlink, which is used to link from one page to another
href	The href attribute specifies the URL of the page the link goes to
<table>	HTML tables allow web developers to arrange data into rows and columns
<tr>	The <tr> HTML element defines a row of cells in a table
<th>	The <th> tag defines a header cell in an HTML table.
<td>	The <td> tag defines a standard data cell in an HTML table.
<thead>	The <thead> tag is used to group header content in an HTML table
<tbody>	The <tbody> tag is used to group the body content

	in an HTML table
<tfoot>	The <tfoot> tag is used to group footer content in an HTML table.

2) Solved Lab Activities

Sr. No	*Allocated Time*	*Level of Complexity*	*CLO Mapping*
Activity 1	*20 Minutes*	*Low*	*CLO-4*
Activity 2	*20 Minutes*	*Low*	*CLO-4*
Activity 3	*20 Minutes*	*Low*	*CLO-4*

Activity 1:

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

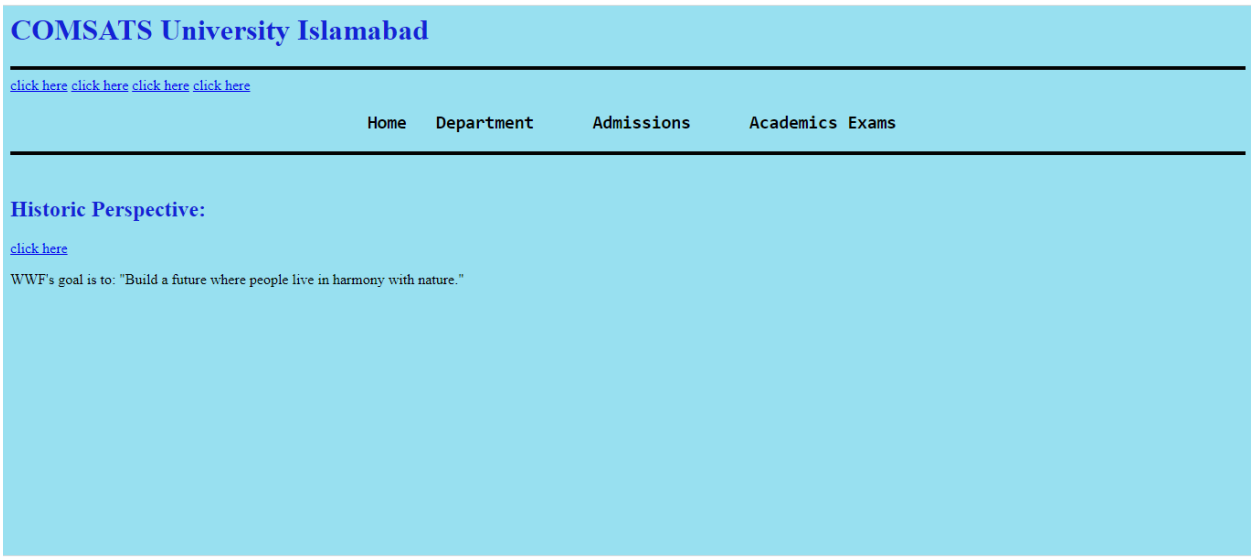
- **Create links to home, departments, admission and exams pages of CUI Islamabad campus**

Solution:

```
<html>
<head>
<title>Home</title>
</head>
<body bgcolor="#98E0F0">
<h1><font color="#1322D6"> COMSATS University Islamabad </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>
```

Output



Activity 2:

Create a table to add the add basic html table including the details of Company, Contact, and Country.

Solution:

```
<!DOCTYPE html>
<html>
<style>
table, th, td {
  border:1px solid black;
}
</style>
<body>

<h2>A basic HTML table</h2>

<table style="width:100%">
  <tr>
    <th>Company</th>
    <th>Contact</th>
    <th>Country</th>
  </tr>
  <tr>
    <td>Alfreds Futterkiste</td>
```

```
    <td>Maria Anders</td>
    <td>Germany</td>
  </tr>
  <tr>
    <td>Centro comercial Moctezuma</td>
    <td>Francisco Chang</td>
    <td>Mexico</td>
  </tr>
</table>

</body>
</html>
```

Output

A basic HTML table

Company	Contact	Country
Alfreds Futterkiste	Maria Anders	Germany
Centro comercial Moctezuma	Francisco Chang	Mexico

Activity 3:

Create a table as shown in the figure given below:

OCTOBER		NOVEMBER	
Sales	Profit	Sales	Profit
$200,00	$50,00	$300,000	$70,000
NOVEMBER WAS MORE PRODUSTIVE			

Solution:

```
<table>
  <thead>
    <tr>
      <th colspan="2">October</th>
      <th colspan="2">November</th>
```

```
    </tr>
  </thead>

  <tbody>
    <tr>
      <td>Sales</td>
      <td>Profit</td>
      <td>Sales</td>
      <td>Profit</td>
    </tr>
    <tr>
      <td>$200,00</td>
      <td>$50,00</td>
      <td>$300,000</td>
      <td>$70,000</td>
    </tr>
  </tbody>

  <tfoot>
    <tr>
      <th colspan= "4">November was more produstive</th>
    </tr>
  </tfoot>
</table>
```

Output

OCTOBER		NOVEMBER	
Sales	Profit	Sales	Profit
$200,00	$50,00	$300,000	$70,000
NOVEMBER WAS MORE PRODUSTIVE			

3) Graded Lab Tasks

Note: The instructor can design graded lab activities according to the level of difficult and complexity of the solved lab activities. The lab tasks assigned by the instructor should be evaluated in the same lab.

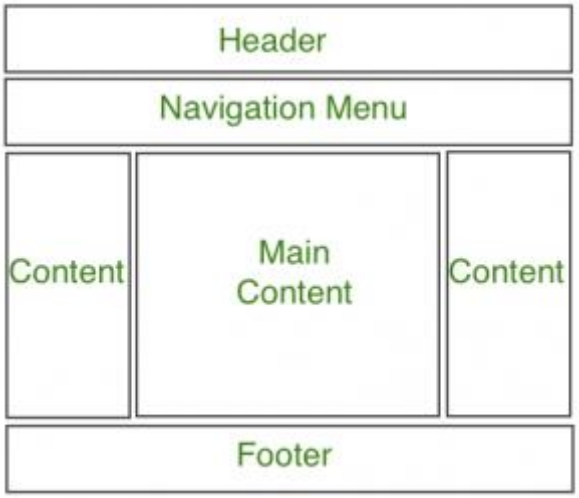
Lab Task 1

Create in-page reference to the list created in the body and marks the locations accordingly.

Add tables to department and admission pages.

Lab Task 2

Use tables to create a page layout given below that displays your profile (Persional Homepage)



Lab 03

HTML FORMS

Objective:

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
- HTML Forms

Instructor Note:

As pre-lab activity, read Chapter 3 from the textbook "Web Design Playground: HTML & CSS the Interactive Way 1st Edition, April 2019 by Paul McFedries".

1) Useful Concepts

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
<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 THML 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) Solved Lab Activities

Sr. No	*Allocated Time*	*Level of Complexity*	*CLO Mapping*
Activity 1	*30 Minutes*	*Medium*	*CLO-4*
Activity 2	*30 Minutes*	*Medium*	*CLO-4*

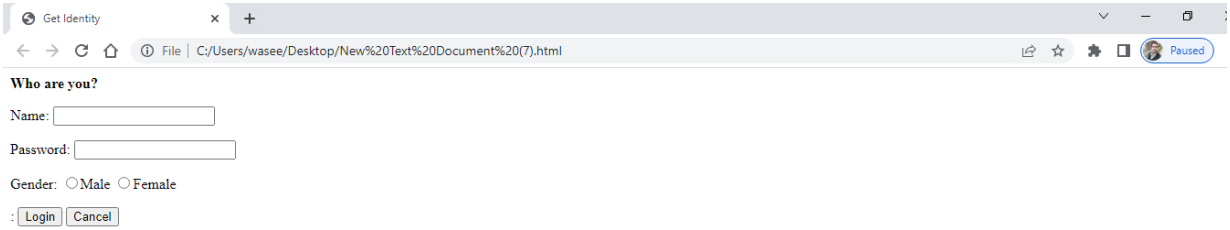
Activity 1:

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>
```

Output:



Activity 2:

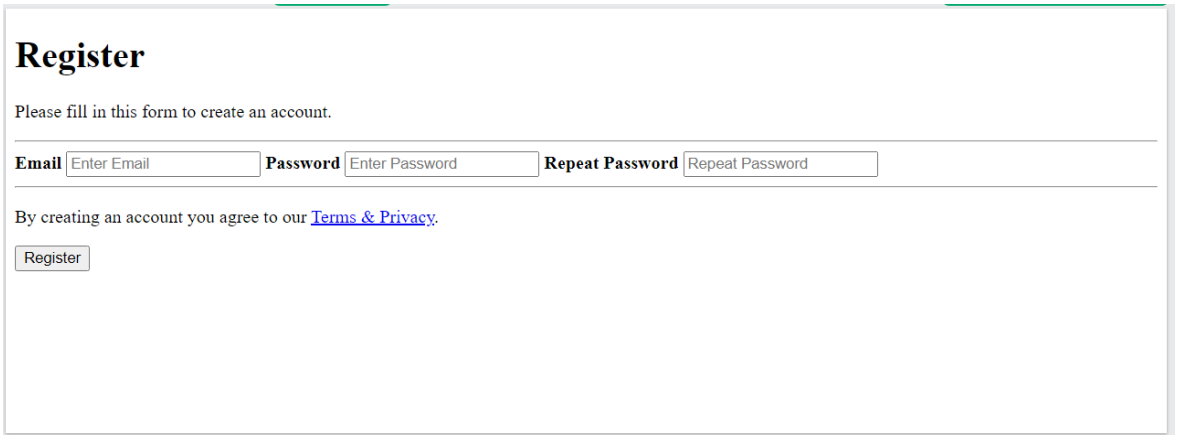
Creating a Signup/Registration form

Solution:

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1">
</head>
<body>

<form>
    <h1>Register</h1>
    <p>Please fill in this form to create an account.</p>
    <hr>

    <label for="email"><b>Email</b></label>
    <input type="text" placeholder="Enter Email" name="email"
id="email" required>

    <label for="psw"><b>Password</b></label>
    <input type="password" placeholder="Enter Password" name="psw"
id="psw" required>

    <label for="psw-repeat"><b>Repeat Password</b></label>
    <input type="password" placeholder="Repeat Password" name="psw-
```

```
repeat" id="psw-repeat" required>
    <hr>
    <p>By creating an account you agree to our <a href="#">Terms &
Privacy</a>.</p>

    <button type="submit" class="registerbtn">Register</button>

 </form>

</body>
</html>
```

Output:

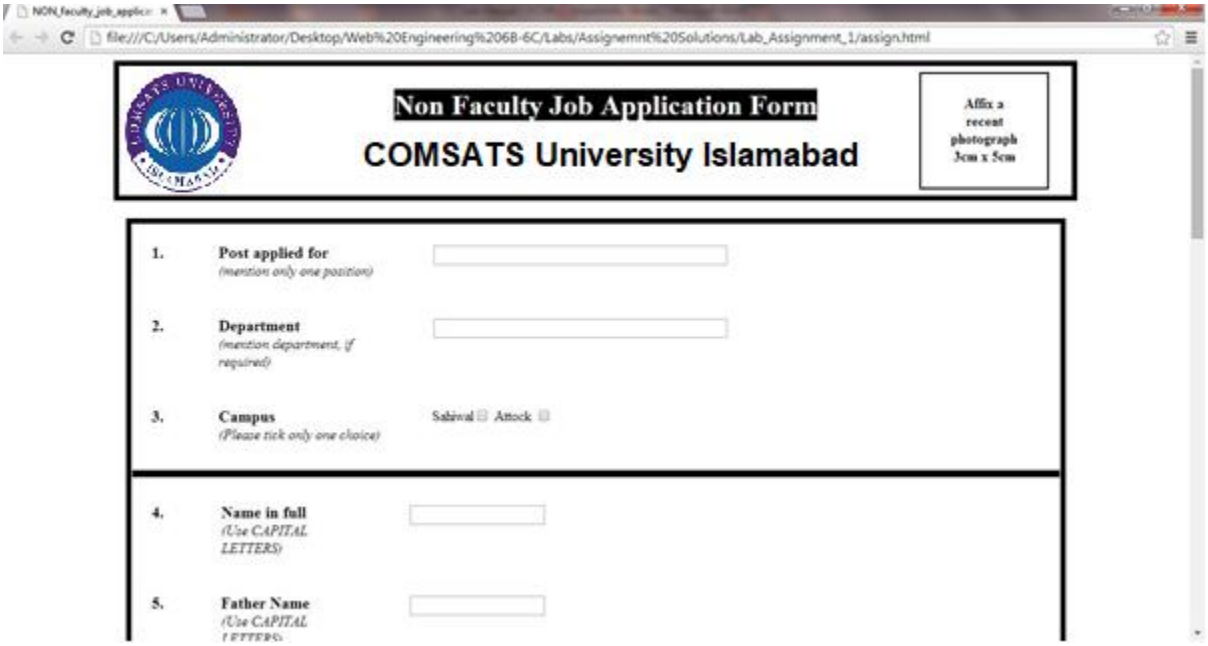


3) Graded Lab Tasks

Note: The instructor can design graded lab activities according to the level of difficult and complexity of the solved lab activities. The lab tasks assigned by the instructor should be evaluated in the same lab.

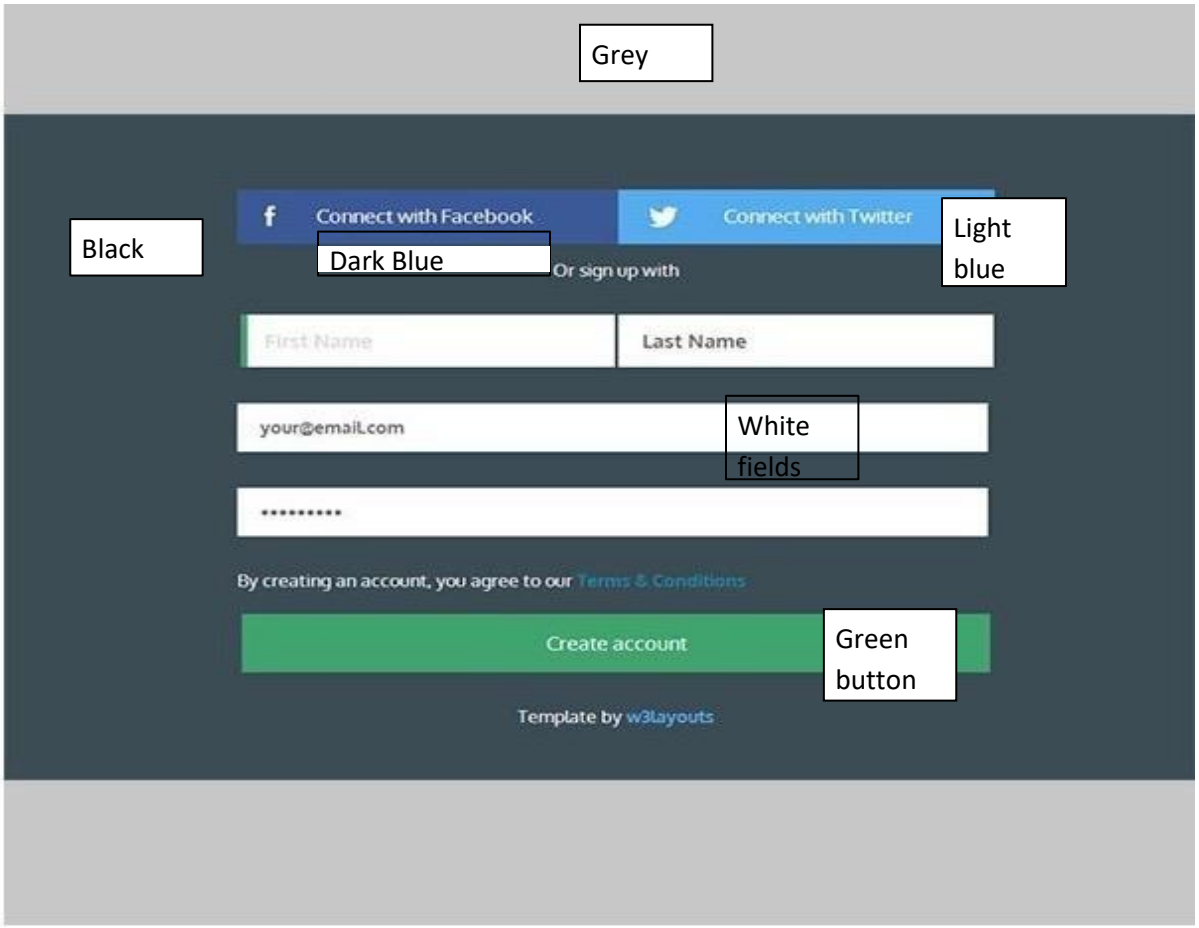
Lab Task 1

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



Lab Task 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.



Lab 04

CSS (Cascading Style Sheet) Basics

Objective:

To familiarize the students with the cascading style sheets.

Activity Outcomes:

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

Instructor Note:

As pre-lab activity, read Chapter 4 from the textbook "Web Design Playground: HTML & CSS the Interactive Way 1st Edition, April 2019 by Paul McFedries".

1) Useful Concepts

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) Solved Lab Activities

Sr. No	*Allocated Time*	*Level of Complexity*	*CLO Mapping*
Activity 1	*15 Minutes*	*Low*	*CLO-4*
Activity 2	*45 Minutes*	*High*	*CLO-4*

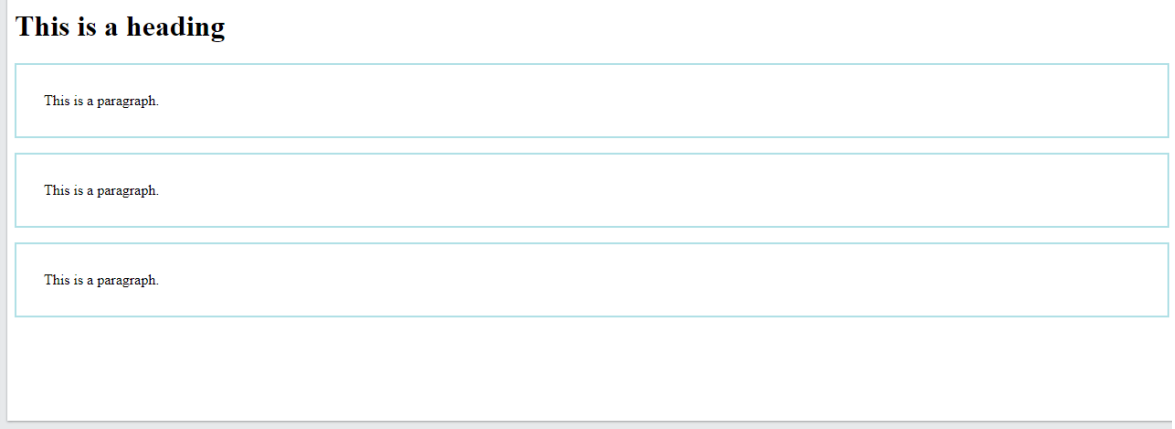
Activity 1:

Apply border and padding properties on the different paragraphs of an HTML document.

Solution:

```
<!DOCTYPE html>
<html>
<head>
<style>
p {
  border: 2px solid powderblue;
  padding: 30px;
}
</style>
</head>
<body>

<h1>This is a heading</h1>

<p>This is a paragraph.</p>
<p>This is a paragraph.</p>
<p>This is a paragraph.</p>

</body>
</html>
```

Output:



Activity 2:

Use the HTML page created in Lab 3 (Signup page) and create an external style sheet that styles the elements of the page including the style for

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

Solution:

External Stylesheet.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;  }
```

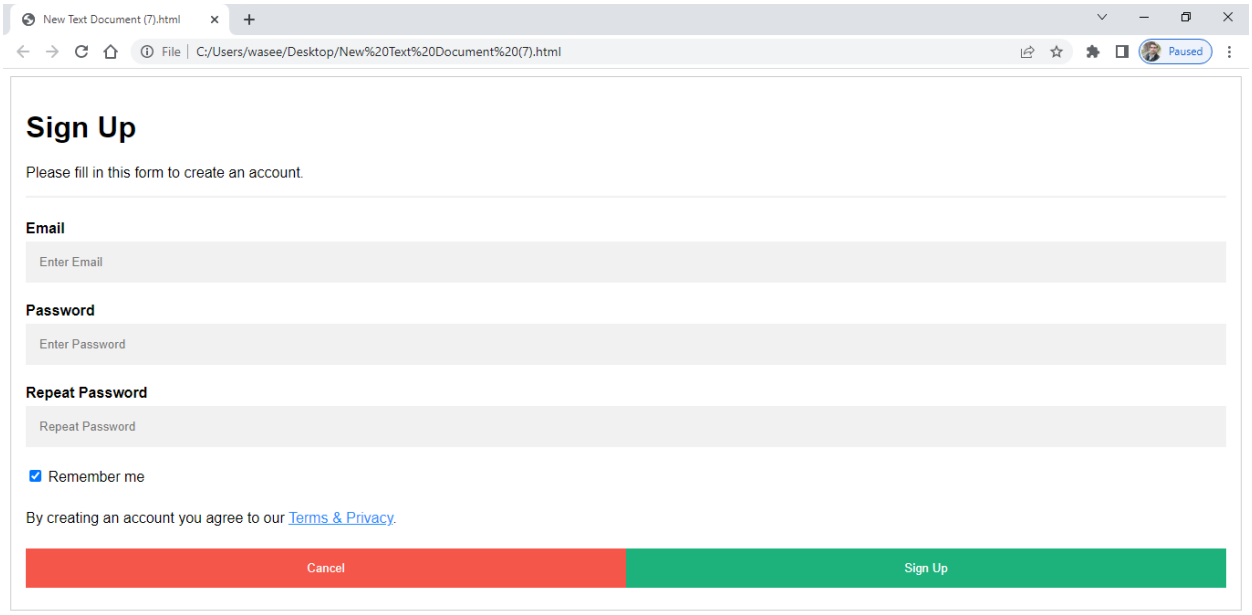
```
<!DOCTYPE html>
<html>
<style>
body {font-family: Arial, Helvetica, sans-serif;}
* {box-sizing: border-box}

/* Full-width input fields */
```

```
input[type=text], input[type=password] {
  width: 100%;
  padding: 15px;
  margin: 5px 0 22px 0;
  display: inline-block;
  border: none;
  background: #f1f1f1;
}

input[type=text]:focus, input[type=password]:focus {
  background-color: #ddd;
  outline: none;
}

hr {
  border: 1px solid #f1f1f1;
  margin-bottom: 25px;
}

/* Set a style for all buttons */
button {
  background-color: #04AA6D;
  color: white;
  padding: 14px 20px;
  margin: 8px 0;
  border: none;
  cursor: pointer;
  width: 100%;
  opacity: 0.9;
}

button:hover {
  opacity:1;
}

/* Extra styles for the cancel button */
.cancelbtn {
  padding: 14px 20px;
  background-color: #f44336;
}

/* Float cancel and signup buttons and add an equal width */
.cancelbtn, .signupbtn {
  float: left;
  width: 50%;
}
```

```
/* Add padding to container elements */
.container {
  padding: 16px;
}

/* Clear floats */
.clearfix::after {
  content: "";
  clear: both;
  display: table;
}

/* Change styles for cancel button and signup button on extra small
screens */
@media screen and (max-width: 300px) {
  .cancelbtn, .signupbtn {
     width: 100%;
  }
}
</style>
<body>

<form action="/action_page.php" style="border:1px solid #ccc">
  <div class="container">
    <h1>Sign Up</h1>
    <p>Please fill in this form to create an account.</p>
    <hr>

    <label for="email"><b>Email</b></label>
    <input type="text" placeholder="Enter Email" name="email"
required>

    <label for="psw"><b>Password</b></label>
    <input type="password" placeholder="Enter Password" name="psw"
required>

    <label for="psw-repeat"><b>Repeat Password</b></label>
    <input type="password" placeholder="Repeat Password" name="psw-
repeat" required>

    <label>
      <input type="checkbox" checked="checked" name="remember"
style="margin-bottom:15px"> Remember me
    </label>

    <p>By creating an account you agree to our <a href="#"
style="color:dodgerblue">Terms & Privacy</a>.</p>
```

```
    <div class="clearfix">
      <button type="button" class="cancelbtn">Cancel</button>
      <button type="submit" class="signupbtn">Sign Up</button>
    </div>
  </div>
</form>

</body>
</html>
```

Output:

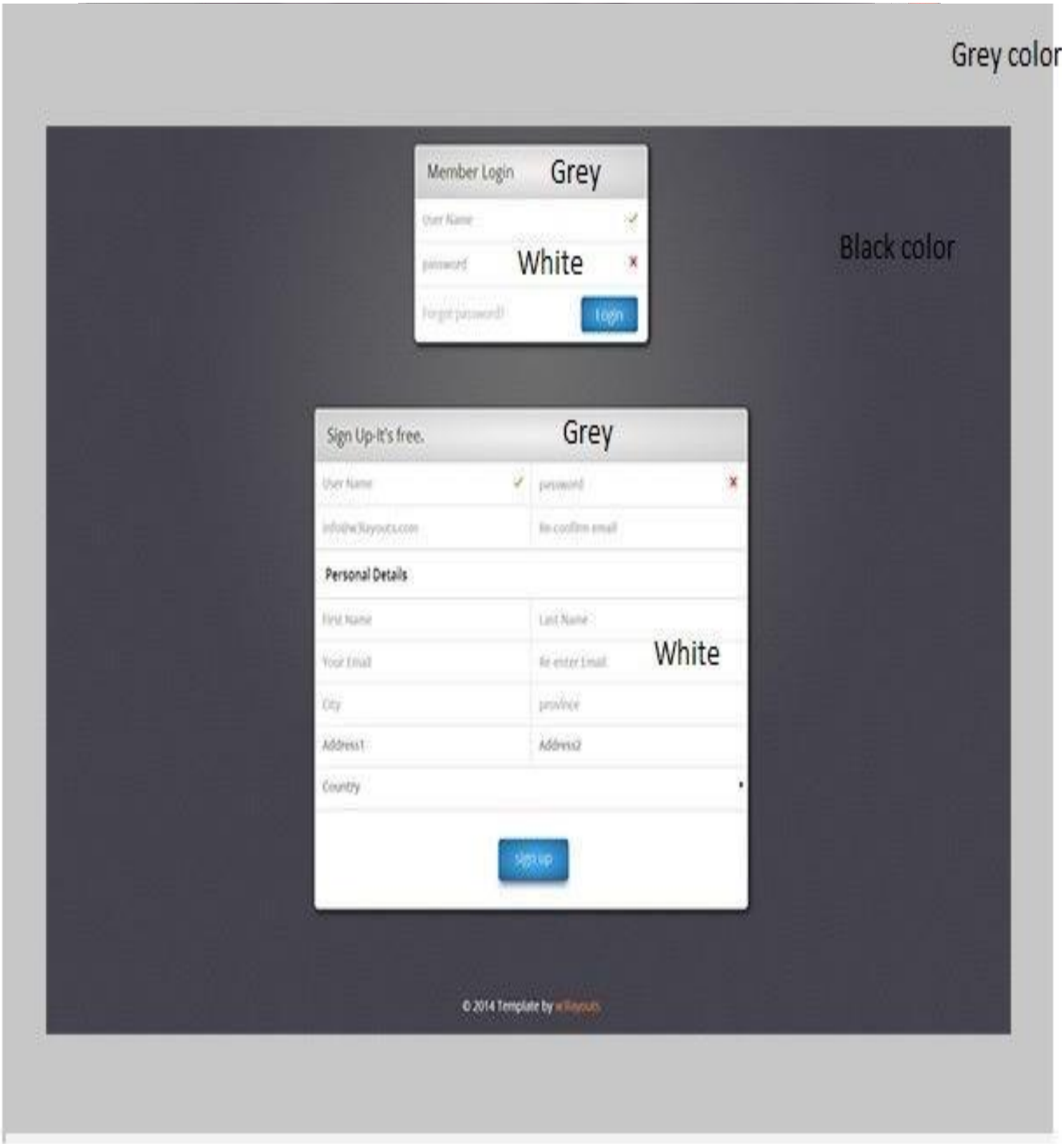


3) Graded Lab Tasks

Note: The instructor can design graded lab activities according to the level of difficult and complexity of the solved lab activities. The lab tasks assigned by the instructor should be evaluated in the same lab.

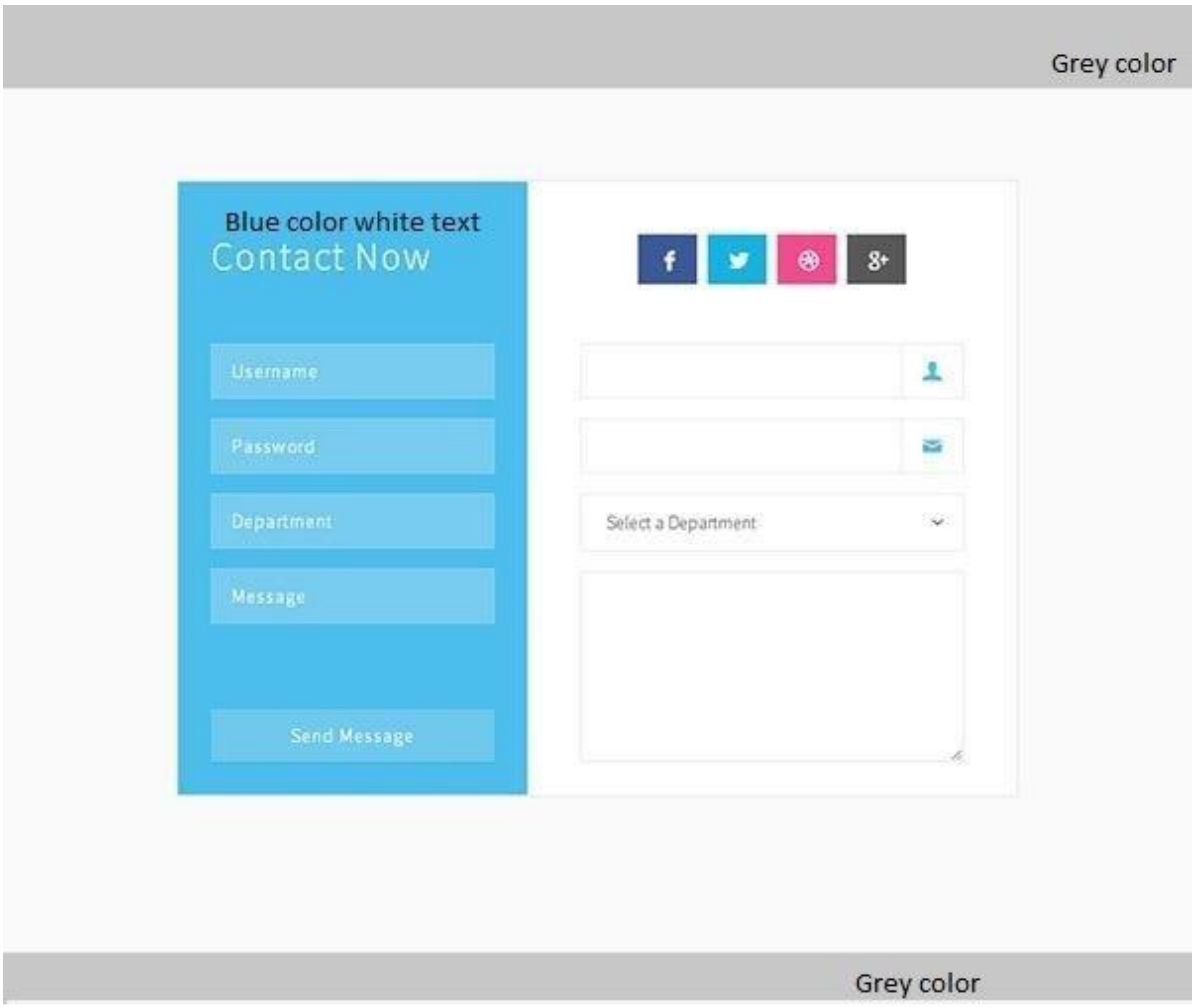
Lab Task 1

Design the web template given below using tables and external CSS



Lab Task 2

Design the web template given below using tables and external CSS



Lab 05

Designing page layout using DIV and CSS

Objective:

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
- the use of CSS for page layout designing

Instructor Note:

As pre-lab activity, read Chapter 4 from the textbook "Web Design Playground: HTML & CSS the Interactive Way 1st Edition, April 2019 by Paul McFedries".

1) Useful Concepts

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: …… . 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) Solved Lab Activities

Sr. No	***Allocated Time***	***Level of Complexity***	***CLO Mapping***
Activity 1	***30 Minutes***	***Medium***	***CLO-4***
Activity 2	***30 Minutes***	***Medium***	***CLO-4***

Activity 1:

Use div and CSS properties to design a responsive form that contains First name, Last Name, Country (dropdown), Subject and submit button.

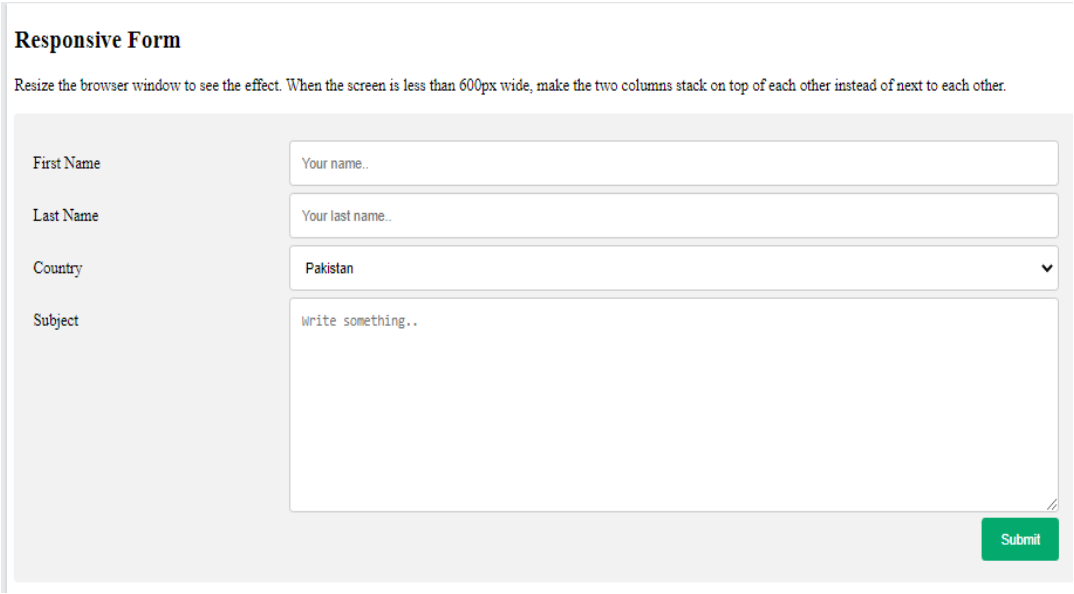
Solution:

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1">
<style>
* {
  box-sizing: border-box;
}

input[type=text], select, textarea {
  width: 100%;
  padding: 12px;
  border: 1px solid #ccc;
  border-radius: 4px;
  resize: vertical;
}

label {
  padding: 12px 12px 12px 0;
  display: inline-block;
}

input[type=submit] {
  background-color: #04AA6D;
  color: white;
  padding: 12px 20px;
  border: none;
  border-radius: 4px;
  cursor: pointer;
  float: right;
}

input[type=submit]:hover {
  background-color: #45a049;
}

.container {
  border-radius: 5px;
  background-color: #f2f2f2;
  padding: 20px;
}

.col-25 {
  float: left;
  width: 25%;
  margin-top: 6px;
```

```
}

.col-75 {
  float: left;
  width: 75%;
  margin-top: 6px;
}

/* Clear floats after the columns */
.row:after {
  content: "";
  display: table;
  clear: both;
}

/* Responsive layout - when the screen is less than 600px wide, make
the two columns stack on top of each other instead of next to each
other */
@media screen and (max-width: 600px) {
  .col-25, .col-75, input[type=submit] {
    width: 100%;
    margin-top: 0;
  }
}
</style>
</head>
<body>

<h2>Responsive Form</h2>
<p>Resize the browser window to see the effect. When the screen is
less than 600px wide, make the two columns stack on top of each other
instead of next to each other.</p>

<div class="container">
  <form action="/action_page.php">
    <div class="row">
      <div class="col-25">
        <label for="fname">First Name</label>
      </div>
      <div class="col-75">
        <input type="text" id="fname" name="firstname"
placeholder="Your name..">
      </div>
    </div>
    <div class="row">
      <div class="col-25">
        <label for="lname">Last Name</label>
      </div>
```

```
      <div class="col-75">
        <input type="text" id="lname" name="lastname"
placeholder="Your last name..">
      </div>
    </div>
    <div class="row">
      <div class="col-25">
        <label for="country">Country</label>
      </div>
      <div class="col-75">
         <select id="country" name="country">
          <option value="Pakistan">Pakistan</option>
          <option value="Canada">Canada</option>
          <option value="usa">USA</option>
        </select>
      </div>
    </div>
    <div class="row">
      <div class="col-25">
        <label for="subject">Subject</label>
      </div>
      <div class="col-75">
        <textarea id="subject" name="subject" placeholder="Write
something.." style="height:200px"></textarea>
      </div>
    </div>
    <div class="row">
      <input type="submit" value="Submit">
    </div>
  </form>
</div>

</body>
</html>
```

Output:



Activity 2:

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

- ***a header section to display the title, style this title with CSS***
- ***a form container that 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;backround-color:gray">First</div>
```

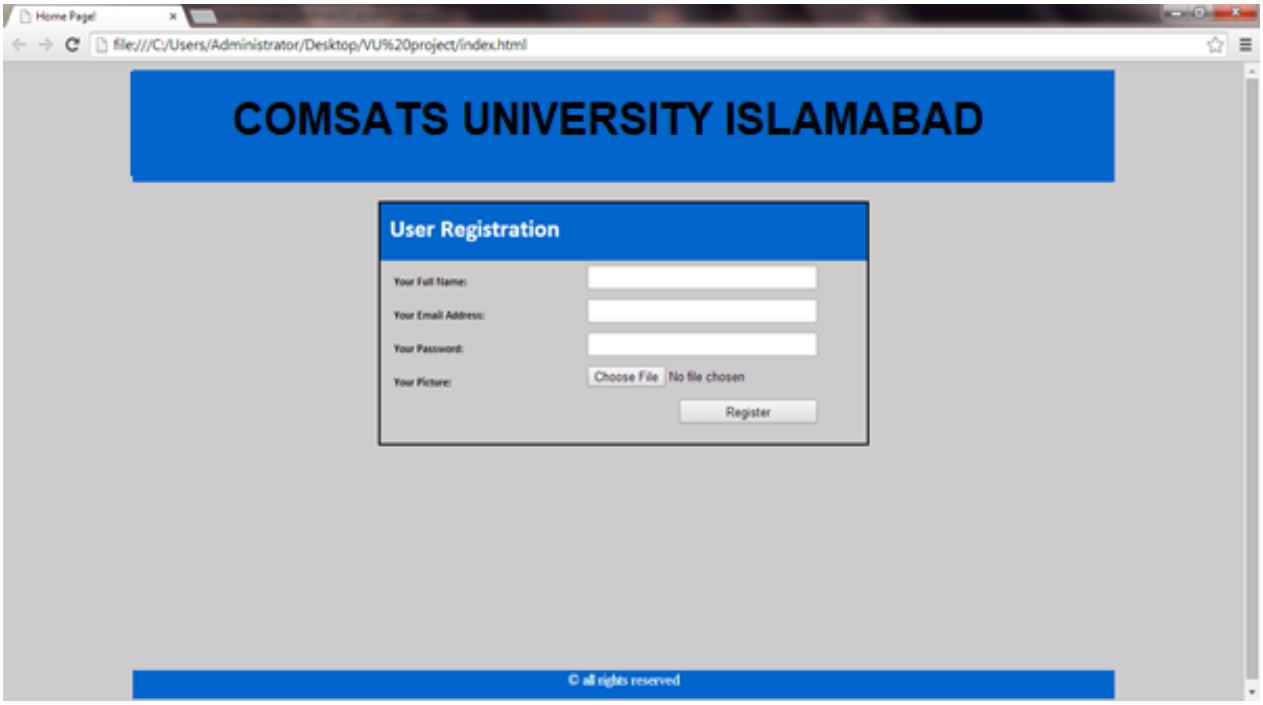
```
<div style="width:100px;backround-color:red">second</div>
</div>
</body>
</html>
```

```
<!DOCTYPE html>
<html>
<style>
body {font-family: Arial, Helvetica, sans-serif;}
* {box-sizing: border-box}

/* Full-width input fields */
input[type=text], input[type=password] {
  width: 100%;
  padding: 15px;
  margin: 5px 0 22px 0;
  display: inline-block;
  border: none;
  background: #f1f1f1;
}

input[type=text]:focus, input[type=password]:focus {
  background-color: #ddd;
  outline: none;
}

hr {
  border: 1px solid #f1f1f1;
  margin-bottom: 25px;
}

/* Set a style for all buttons */
button {
  background-color: #04AA6D;
  color: white;
  padding: 14px 20px;
  margin: 8px 0;
  border: none;
  cursor: pointer;
  width: 100%;
  opacity: 0.9;
}

button:hover {
  opacity:1;
}
```

```
/* Extra styles for the cancel button */
.cancelbtn {
  padding: 14px 20px;
  background-color: #f44336;
}

/* Float cancel and signup buttons and add an equal width */
.cancelbtn, .signupbtn {
  float: left;
  width: 50%;
}

/* Add padding to container elements */
.container {
  padding: 16px;
}

/* Clear floats */
.clearfix::after {
  content: "";
  clear: both;
  display: table;
}

/* Change styles for cancel button and signup button on extra small
screens */
@media screen and (max-width: 300px) {
  .cancelbtn, .signupbtn {
     width: 100%;
  }
}
</style>
<body>

<form action="/action_page.php" style="border:1px solid #ccc">
  <div class="container">
    <h1>Sign Up</h1>
    <p>Please fill in this form to create an account.</p>
    <hr>

    <label for="email"><b>Email</b></label>
    <input type="text" placeholder="Enter Email" name="email"
required>

    <label for="psw"><b>Password</b></label>
    <input type="password" placeholder="Enter Password" name="psw"
```

```
required>

    <label for="psw-repeat"><b>Repeat Password</b></label>
    <input type="password" placeholder="Repeat Password" name="psw-
repeat" required>

    <label>
      <input type="checkbox" checked="checked" name="remember"
style="margin-bottom:15px"> Remember me
    </label>

    <p>By creating an account you agree to our <a href="#"
style="color:dodgerblue">Terms & Privacy</a>.</p>

    <div class="clearfix">
      <button type="button" class="cancelbtn">Cancel</button>
      <button type="submit" class="signupbtn">Sign Up</button>
    </div>
  </div>
</form>

</body>
</html>
```

Output:

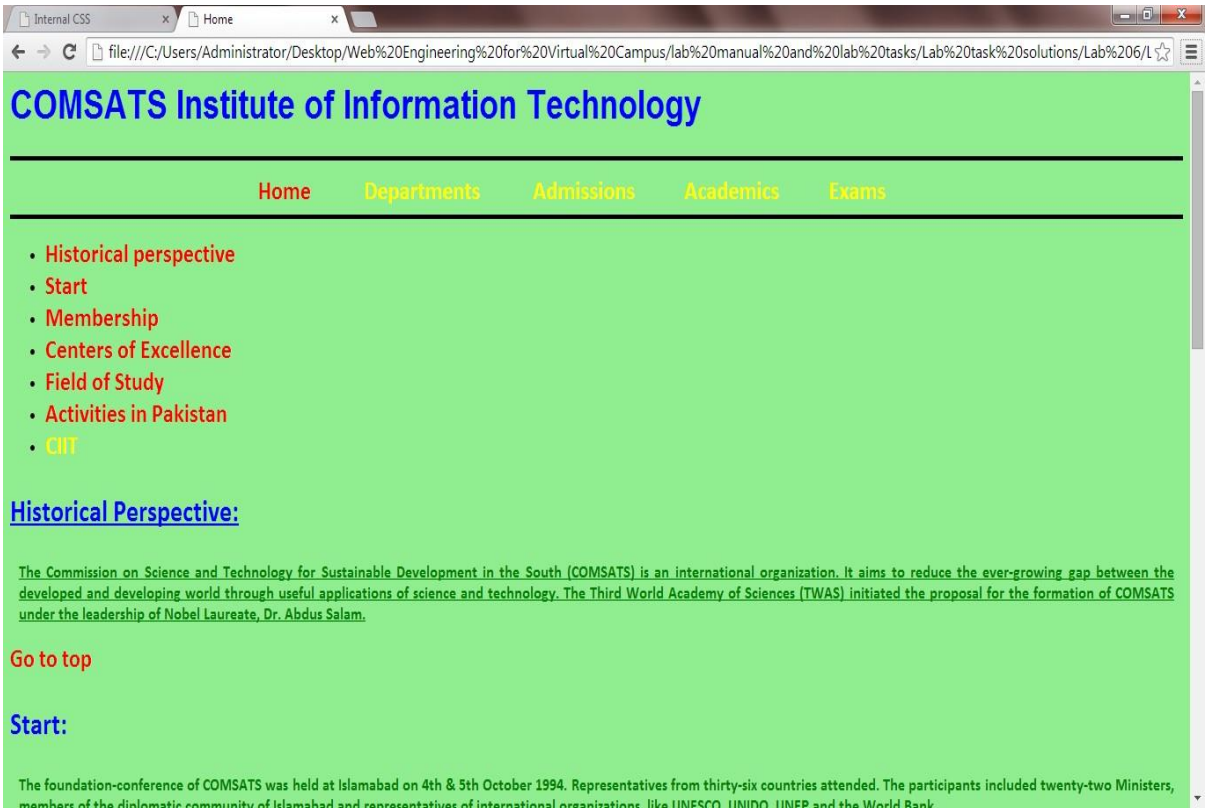


3) Graded Lab Tasks

Note: The instructor can design graded lab activities according to the level of difficult and complexity of the solved lab activities. The lab tasks assigned by the instructor should be evaluated in the same lab.

Lab Task 1

Design the web template given below using tables and external CSS



Lab Task 2

Design the web page for Comsats Alumni using External CSS.

Lab 06

JavaScript Basics

Objective:

To familiarize the students with Java Script.

Activity Outcomes:

After this lab the students should be able to understand

- Java Script basics
- validate the form using Java Script.

Instructor Note:

As pre-lab activity, read Chapter 8 from the textbook “Web Programming with HTML5, CSS, and JavaScript Pap/Psc John Dean, Year 2018

1) Useful Concepts

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
İnter script: <script>, </script> tags are used to start and end a javascript block
External script: JavaScript is written in a seprate 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 loosly 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
  - o document.write(“hello world”)
  - o document.write(a)
- □ prompt(); is used to take input from users
  - o 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) Solved Lab Activities

Sr. No	*Time Allocated*	*Complexity*	*CLO-Mapping*
Activity 1	*10 Minutes*	*Medium*	*CLO-4*
Activity 2	*10 Minutes*	*Medium*	*CLO-4*
Activity 3	*10 Minutes*	*Medium*	*CLO-4*
Activity 4	*10 Minutes*	*Medium*	*CLO-4*
Activity 5	*10 Minutes*	*Medium*	*CLO-4*

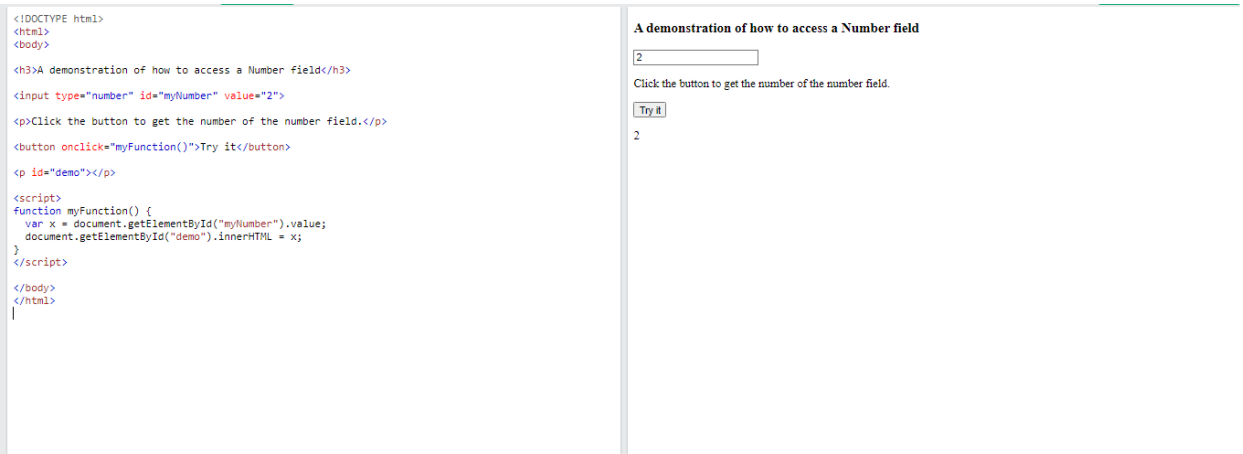
Activity 1:

Use JS to take input from user to enter a number and display the entered number by clicking on a button "Tryit".

Solution:

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

<h3>A demonstration of how to access a Number field</h3>

<input type="number" id="myNumber" value="2">

<p>Click the button to get the number of the number field.</p>

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

<p id="demo"></p>

<script>
function myFunction() {
  var x = document.getElementById("myNumber").value;
  document.getElementById("demo").innerHTML = x;
}
</script>

</body>
</html>
```

Output:



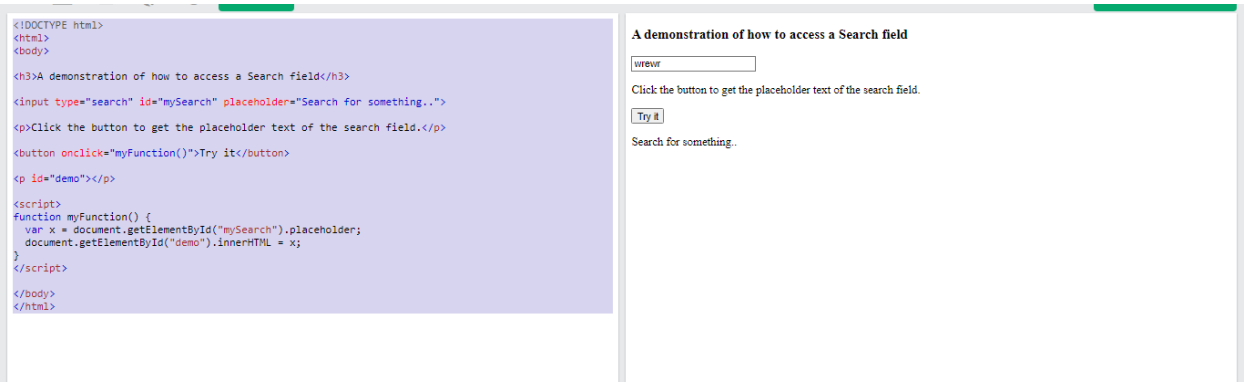
Activity 2:

Use JS to get the placeholder text of the search field by Clicking the button Try it

Input/output and variables:

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

<h3>A demonstration of how to access a Search field</h3>

<input type="search" id="mySearch" placeholder="Search for
something..">

<p>Click the button to get the placeholder text of the search
field.</p>

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

<p id="demo"></p>

<script>
function myFunction() {
  var x = document.getElementById("mySearch").placeholder;
  document.getElementById("demo").innerHTML = x;
}
</script>

</body>
</html>
```

Output:



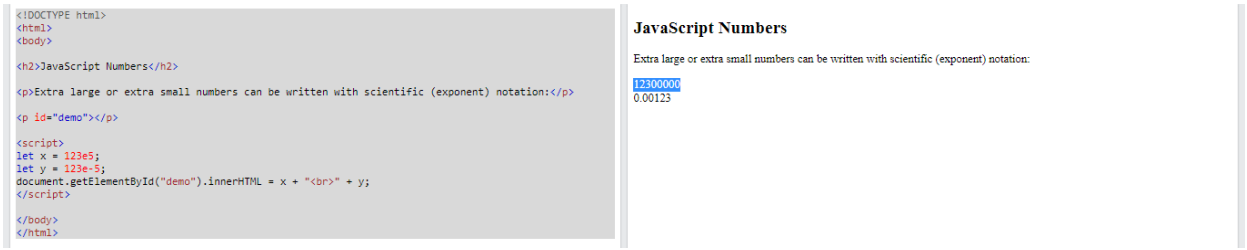
Activity 3:

Use JS to write extra large or extra small numbers in exponent or scientific notation.
e.g 123e5 = 12300000 or 123e-5 = 0.00123

Input/output and variables:

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

<h2>JavaScript Numbers</h2>

<p>Extra large or extra small numbers can be written with scientific
(exponent) notation:</p>

<p id="demo"></p>

<script>
let x = 123e5;
let y = 123e-5;
document.getElementById("demo").innerHTML = x + "<br>" + y;
</script>

</body>

</html>
```

Output:



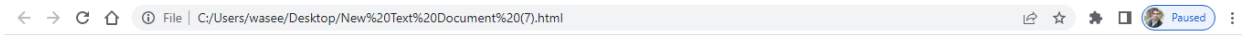
Activity 4:

Use JS Date Function to display the date and time on HTML document.

Input/output and variables:

```
<!DOCTYPE html>
<html lang="en-US">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1">
    <title>Today's Date</title>
</head>

<body>

  <script>
      let d = new Date();
      document.body.innerHTML = "<h1>Today's date is " + d + "</h1>"
  </script>

</body>

</html>
```

Output:



Today's date is Wed Apr 27 2022 23:14:35 GMT+0500 (Pakistan Standard Time)

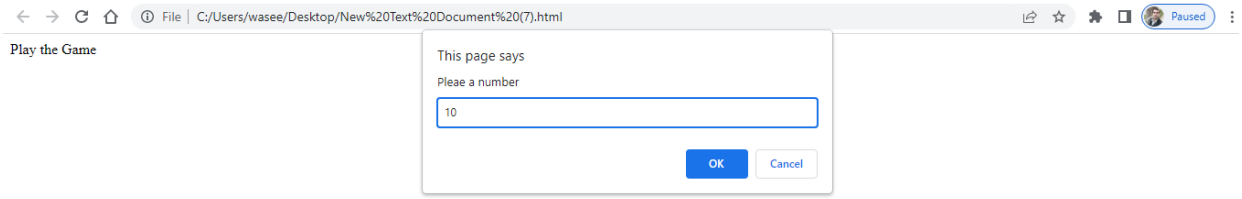
Activity 5:

Use JS Input/output, variables, Functions, Conditional statements to create a game. Get a random number using Math.random function in the JS. In the game it asks user to enter a number. If the entered number matches the randomly generated number it generates an alert you won else it displays the actual number.

Solution:

```
<html>
<head>

<title>Using If condition</title>
<script language="javascript">
function playGame()
{
var res=Math.round(Math.random()*10)
var num=prompt("Pleae 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>
```

Output:



3) Graded Lab Tasks

Note: The instructor can design graded lab activities according to the level of difficult and complexity of the solved lab activities. The lab tasks assigned by the instructor should be evaluated in the same lab.

Lab Task 1-3

- *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.*

Lab Task 4:

- *Write a JS code to validate the signup form in Lab 3.*

Lab 07

JavaScript Timers

Objectives:

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.
- Write timer functions in html for creating Website Carousel.

Instructor Note:

As pre-lab activity, read Chapter 8 from the textbook "Web Programming with HTML5, CSS, and JavaScript Pap/Psc John Dean, Year 2018. And the website https://www.w3schools.com/js/js_timing.asp for detailed functionalities.

1) Useful Concepts

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);

2) Solved Lab Activities

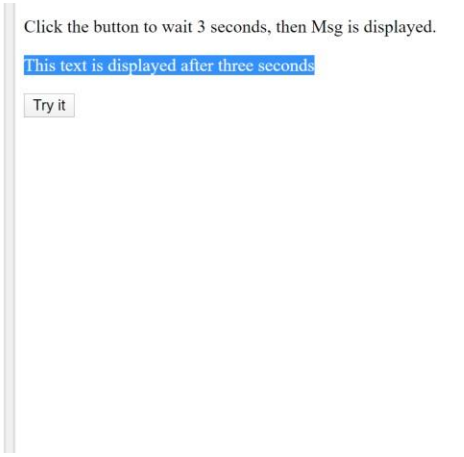
Sr. No	***Time Allocated***	***Complexity***	***CLO-Mapping***
Activity 1	***20 Minutes***	***Medium***	***CLO-1***
Activity 2	***20 Minutes***	***Medium***	***CLO-1***
Activity 3	***20 Minutes***	***Medium***	***CLO-1***

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 3 sec.

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>
```



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:

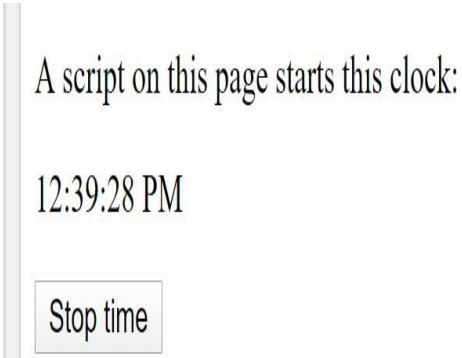
```
<!DOCTYPE html>
<html>
<body>

<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>
```

Activity 3:

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

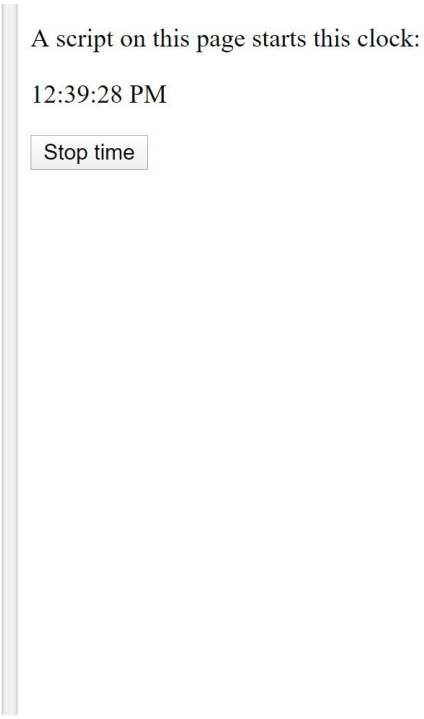


Activity 3:

In forth 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>
```



3) Graded Lab Tasks

Note: The instructor can design graded lab activities according to the level of difficult and complexity of the solved lab activities. The lab tasks assigned by the instructor should be evaluated in the same lab.

Lab Task 1

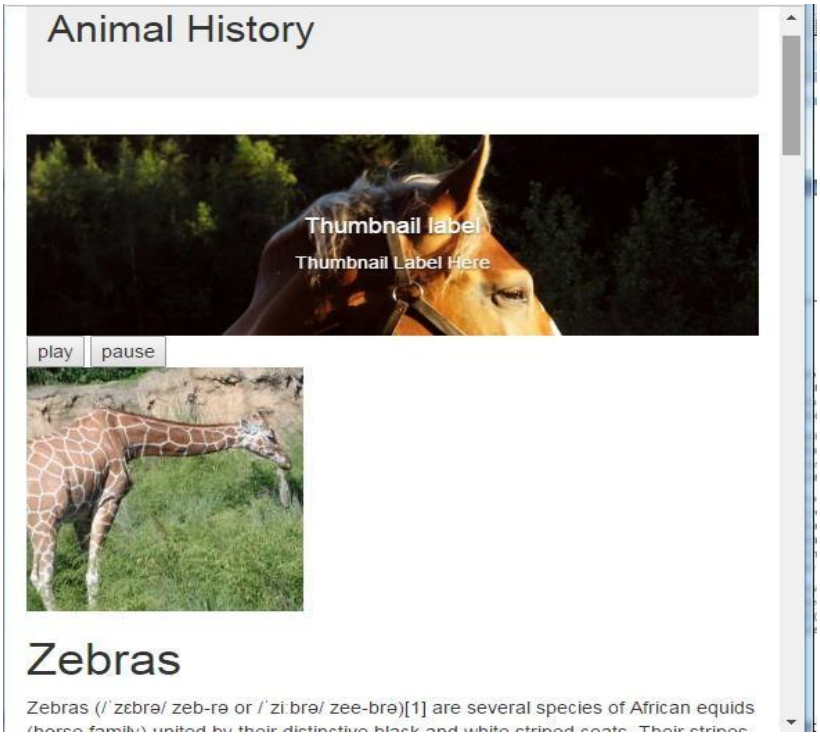
Create a Website Carousel using JavaScript for a responsive animal history web page with its layout for different screen sizes as provided below. Following are the supplementary details about different sections of the page:

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.

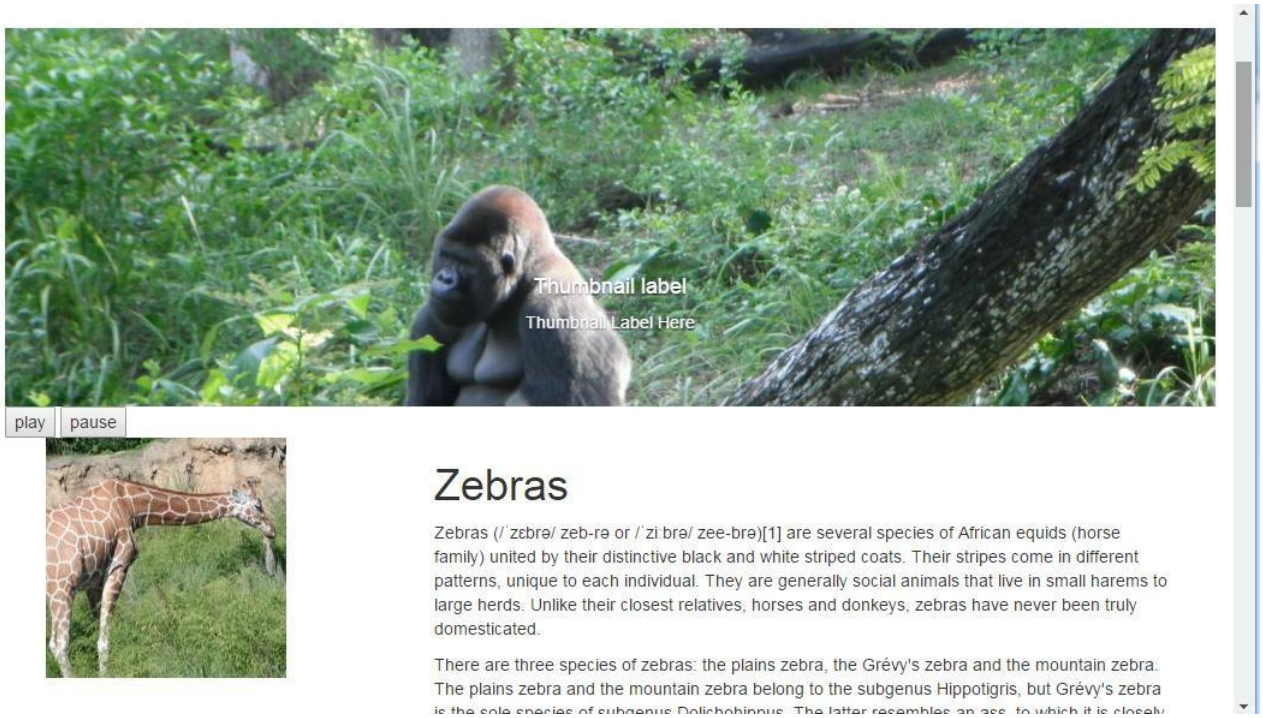
2- Students are supposed to write a JavaScript code for imageSlider with Play and Pause buttons. Slider should stop when Pause button is pressed and start when Play button is clicked.

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

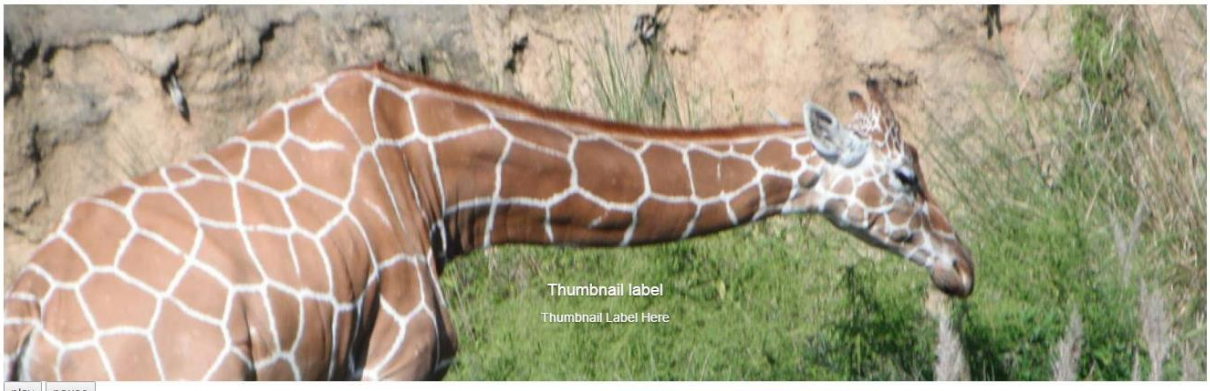


Medium Screen



Large Screen

Animal History



play pause
play pause

Zebras

Zebras (/ˈzebrə/ zeb-rə or /ˈziːbrə/ 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 linage

Leopard

Lab 08

JavaScript Form Validation

Objective:

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:

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

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 Chapter 9 from the textbook "Web Programming with HTML5, CSS, and JavaScript Pap/Psc John Dean, Year 2018.

1) Useful Concepts

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

2) Solved Lab Activites

Sr. No	*Time Allocated*	*Complexity*	*CLO-Mapping*
Activity 1	*20 Minutes*	*Medium*	*CLO-4*
Activity 2	*20 Minutes*	*Medium*	*CLO-4*
Activity 3	*20 Minutes*	*Medium*	*CLO-4*

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:

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

<p>Click the button to do a global search for characters NOT inside the brackets [h] in a string.</p>

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

<p id="demo"></p>

<script>
function myFunction() {
    var str = "Is this all there is?";
    var patt1 = /[^h]/g;
    var result = str.match(patt1);
    document.getElementById("demo").innerHTML = result;
}
</script>

</body>
</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:

```html
<script src="https://code.jquery.com/jquery-3.1.1.min.js" integrity="sha256-hVVnYaiADRTO2PzUGmuLJr8BLUSjGIZsDYGmIJLv2b8="
crossorigin="anonymous"></script>

<form method="post">
        <label for="email" id="email-ariaLabel">Your email address:</label>
        <input id="email" name="email" type="email" class="required"/>
         <span id="error">Not a valid email address</span>
         <span id="success">A valid email address!</span>
</form>

<script>
$(document).ready(function(){
$("#success").hide();
$("#error").hide();
$("#email").keyup(function () {
    var email = $("#email").val();
// i    Perform case-insensitive matching
// g    Perform a global match (find all matches rather than stopping after the first match)
// m    Perform multiline matching
var re = /[A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-Z]{2,4}/igm;
if (re.test(email)) {
    $('#success').show();
    $('#error').hide();
} else {
    $('#error').show();
    $('#success').hide();
}
});
});
</script>
```

3) Graded Lab Tasks

Note: The instructor can design graded lab activities according to the level of difficult and complexity of the solved lab activities. The lab tasks assigned by the instructor should be evaluated in the same lab.

Lab Task 1

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.

Lab 10

PHP Basics

Objective:

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

Instructor Note:

As pre-lab activity, read Chapter 1 from the textbook "Beginning PHP and MySQL: From Novice to Professional 5th ed. Edition by by Frank M. Kromann, Year 2018"

1) Useful Concepts

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) Solved Lab Activities

Sr. No	*Time Allocated*	*Complexity*	*CLO-Mapping*
Activity 1	*15 Minutes*	*Medium*	*CLO-4*
Activity 2	*15 Minutes*	*Medium*	*CLO-4*
Activity 3	*15 Minutes*	*Medium*	*CLO-4*
Activity 4	*15 Minutes*	*Medium*	*CLO-4*

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>";
?>
```

Activity 2:

Write the PHP code that creates a table as given below

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

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++)

 {
```

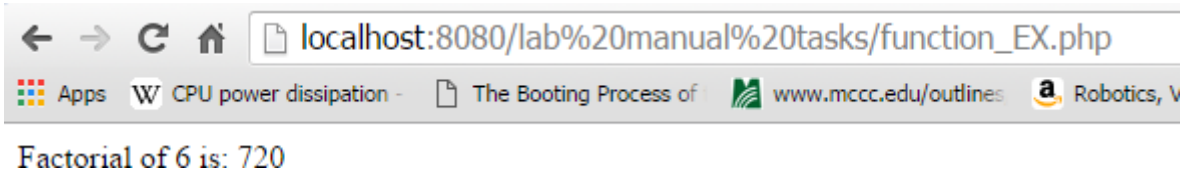
Activity 4:

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);
?>
```

Output:



3) Graded Lab Tasks

Note: The instructor can design graded lab activities according to the level of difficult and complexity of the solved lab activities. The lab tasks assigned by the instructor should be evaluated in the same lab.

Lab Tasks

1. *Write a PHP program that finds the sum of all prime numbers from an array.*
2. *Write a nested for loop in the PHHP 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
```

Lab 11

Laravel Setup and Basics

Objective:

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
- create a basic application using the Laravel Framework.

Instructor Note:

As pre-lab activity, read Chapter 1, 2 from the textbook “Laravel Up and Running, A Framework for Building Modern PHP Apps, 2nd Edition, Matt. Stauffer, Oreilly. Year 2019”

1) Useful Concepts

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) Solved Lab Activities

Sr. No	*Allocated Time*	*Level of Complexity*	*CLO Mapping*
Activity 1	*20 Minutes*	*High*	*CLO-5*
Activity 2	*20 Minutes*	*High*	*CLO-5*
Activity 3	*20 Minutes*	*High*	*CLO-5*

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/

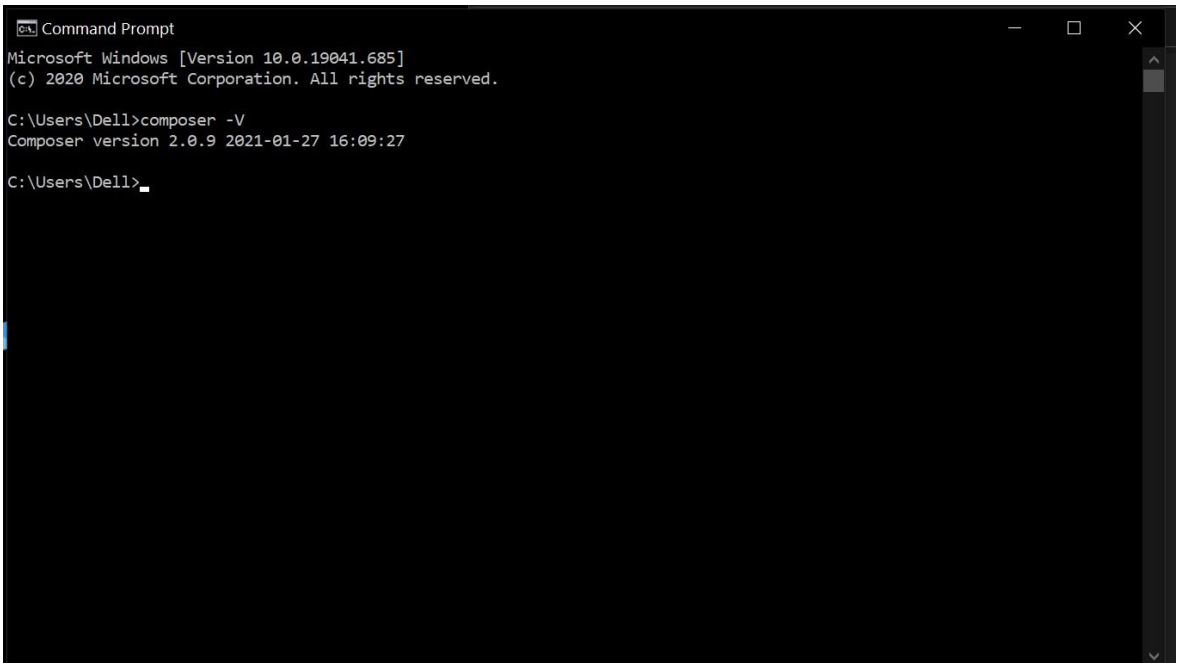
Windows Installer

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.

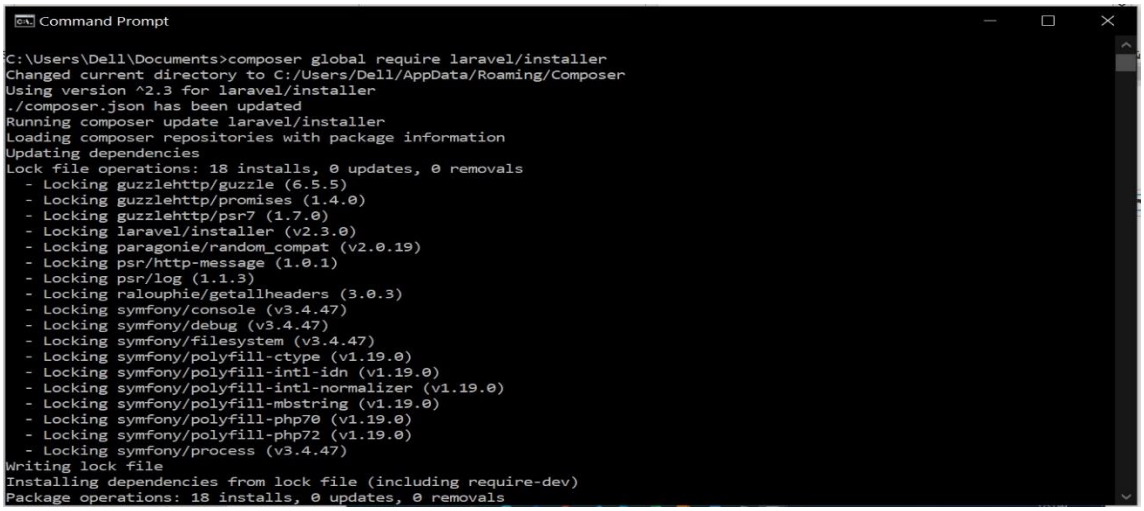


The Laravel Installer

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.

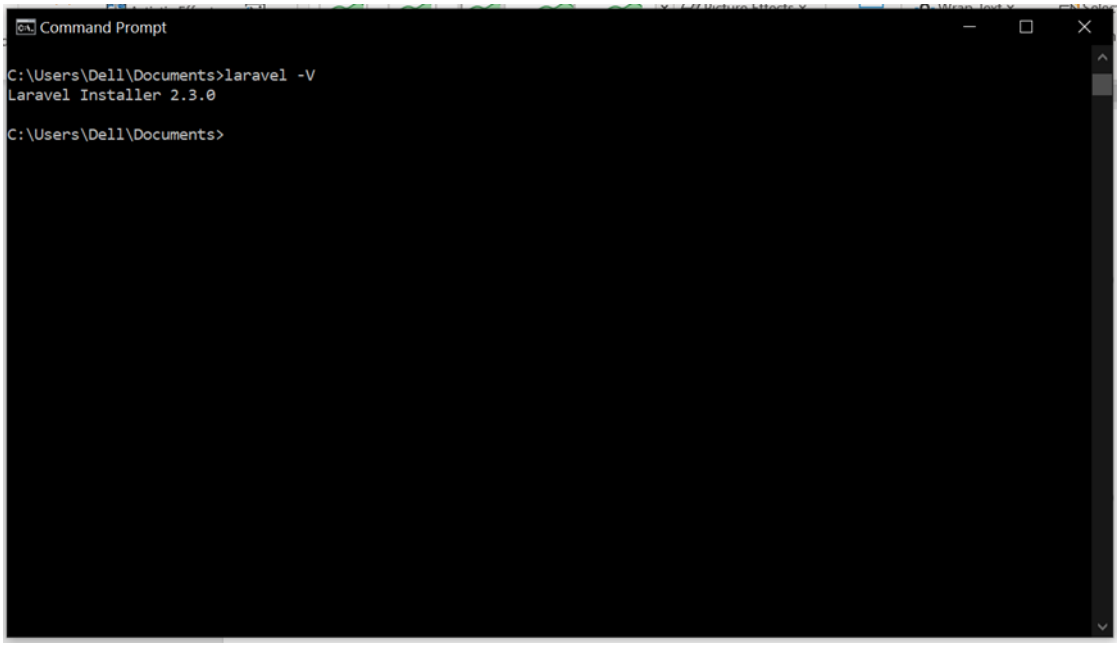


Verify Laravel Installation.

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.



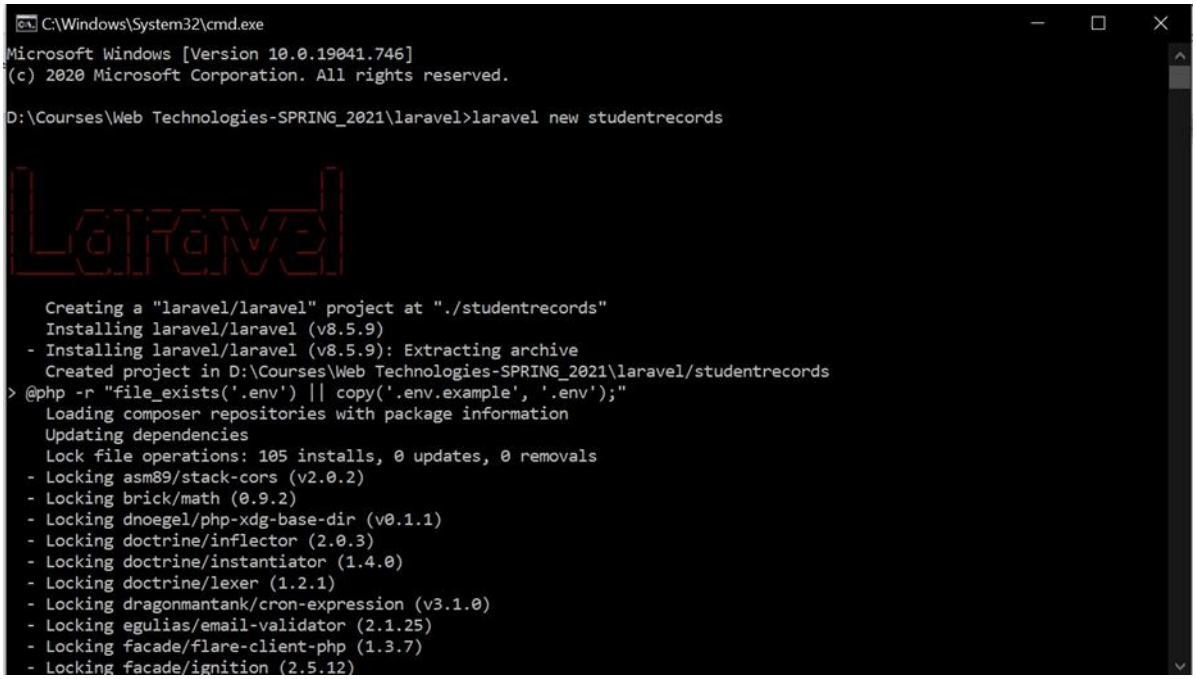
Activity 2:
Creating your first project with the Laravel Framework. It will include,

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

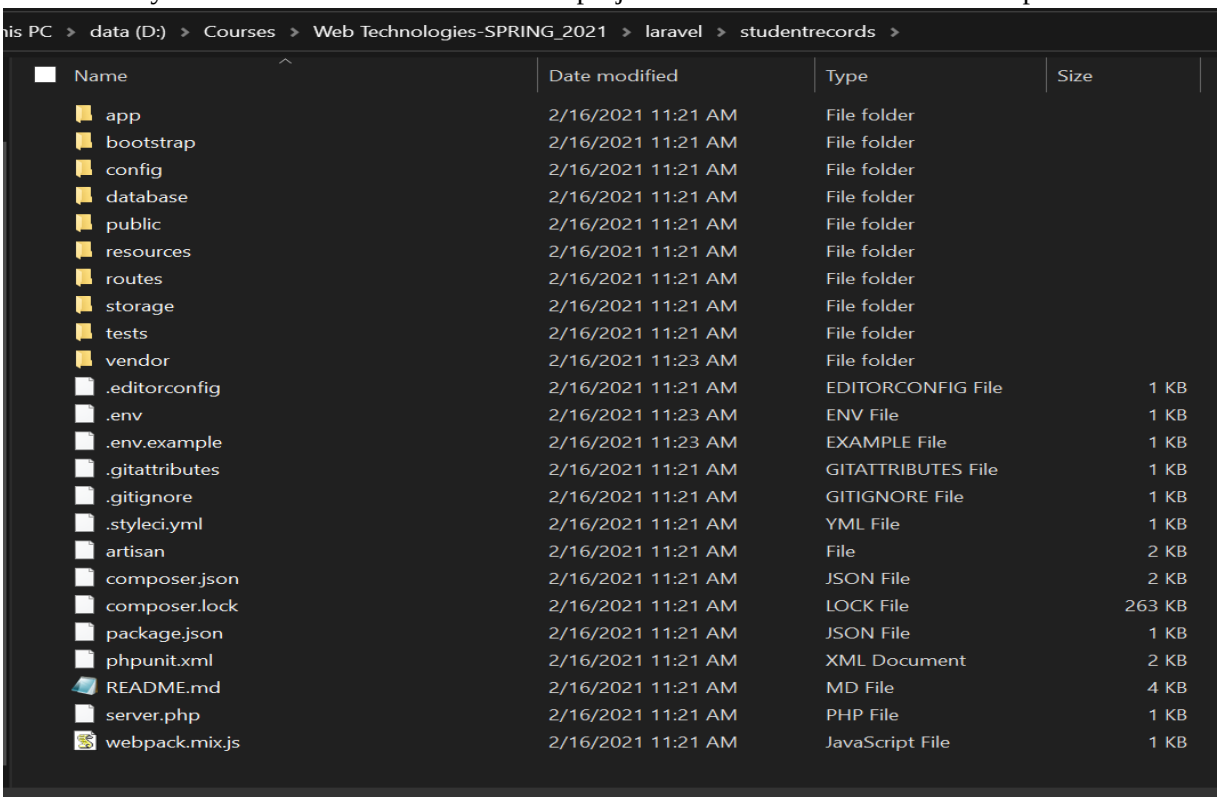
Solution:

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



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



Preview the Project.

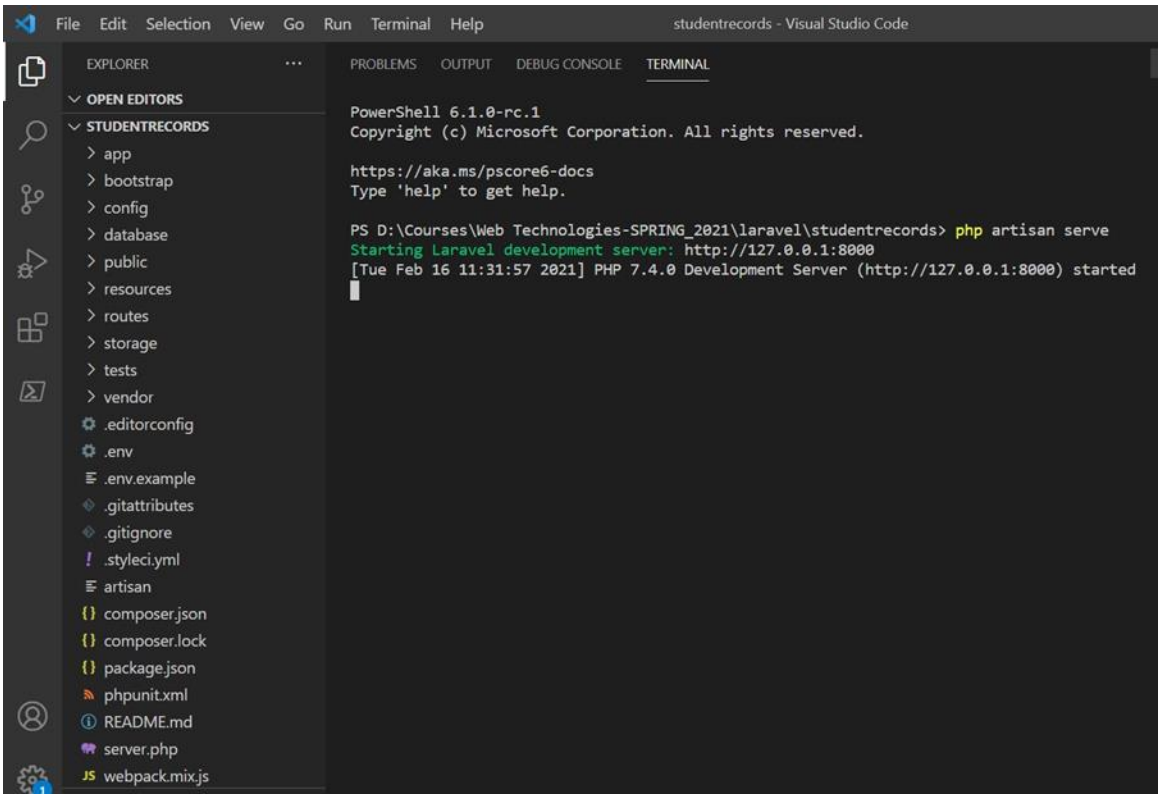
To view the website in the browser, we can use artisan. Artisan is the command line interface included with Laravel. Artisan exists at the root of your application as the artisan and provides a number of

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

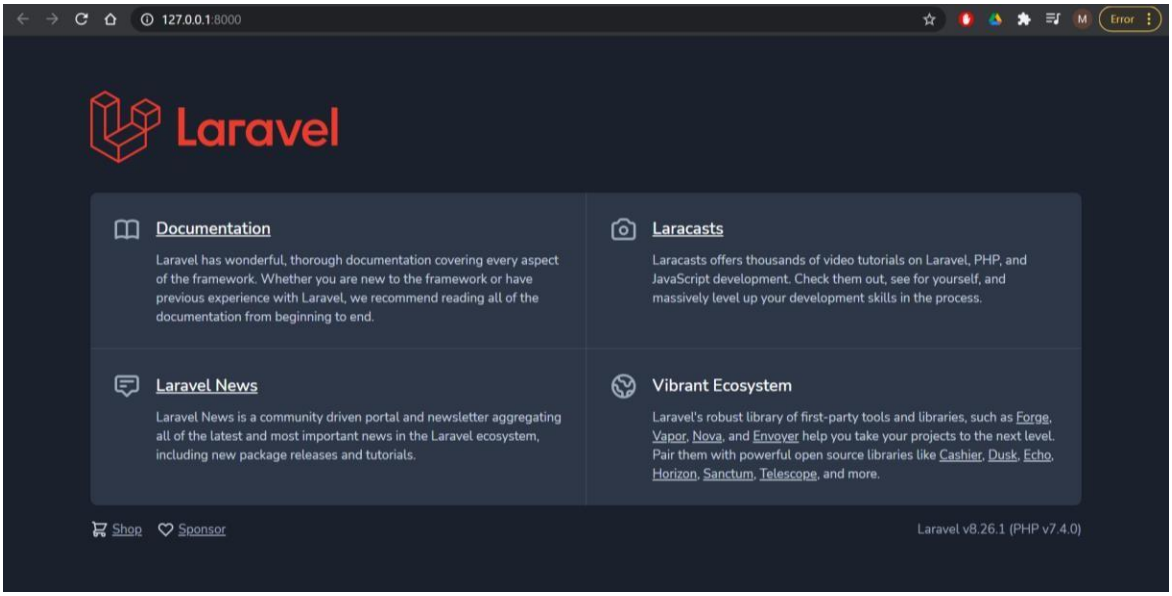
To preview the project, we can use the following command in the command prompt.

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.

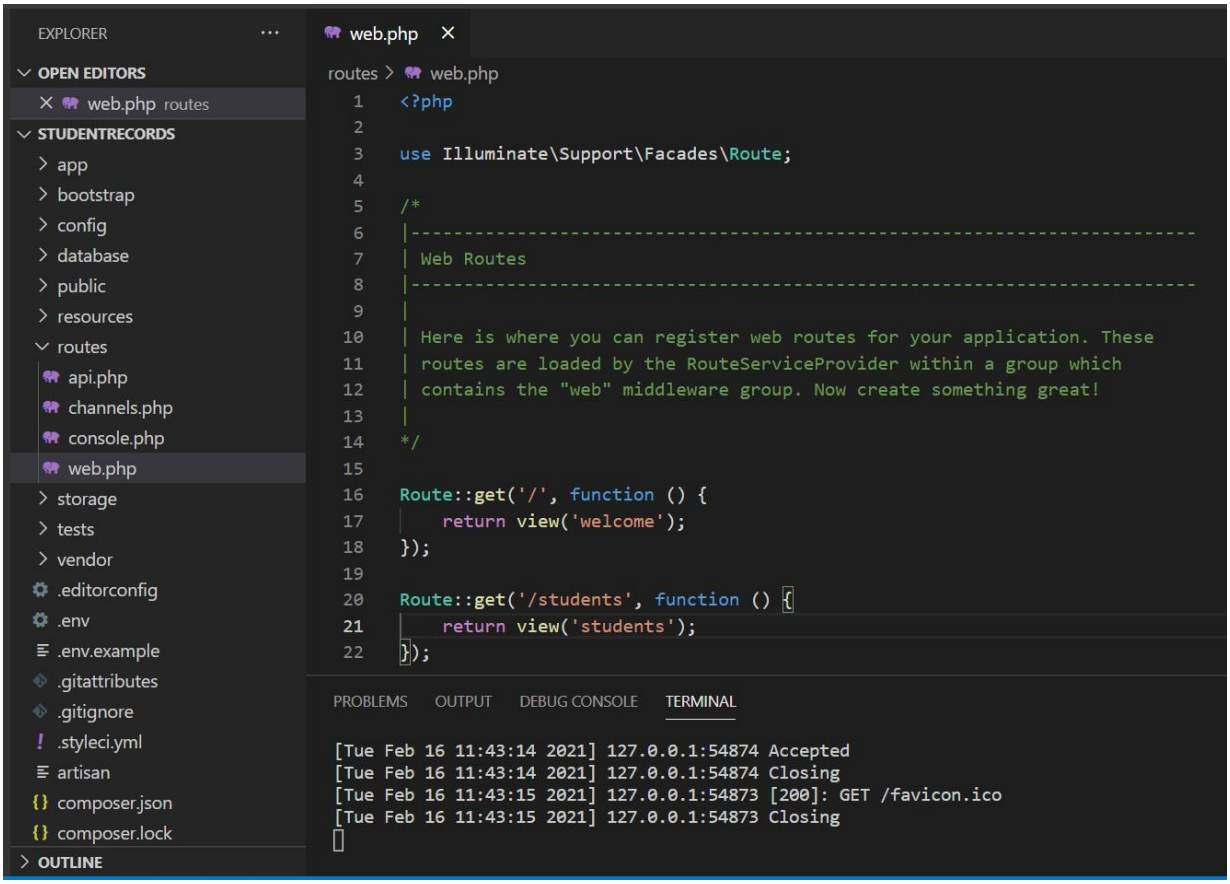
```
Route::get('/', function () {
    return view('welcome');
});

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');
});
```

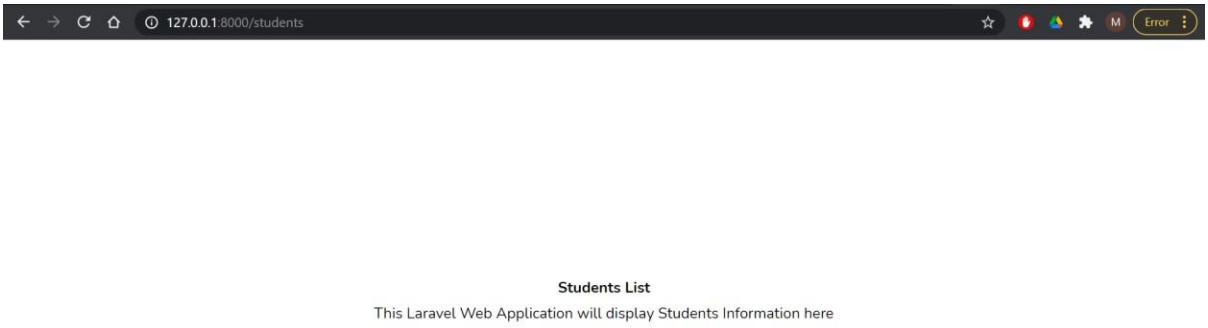
This code above looks for a view in resources/views/home.blade.php or resources/views/home.php and loads its contents and parses any inline PHP or control structures until you have just the view's output. Once you return it, it is passed on to the rest of the application and eventually returned to the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

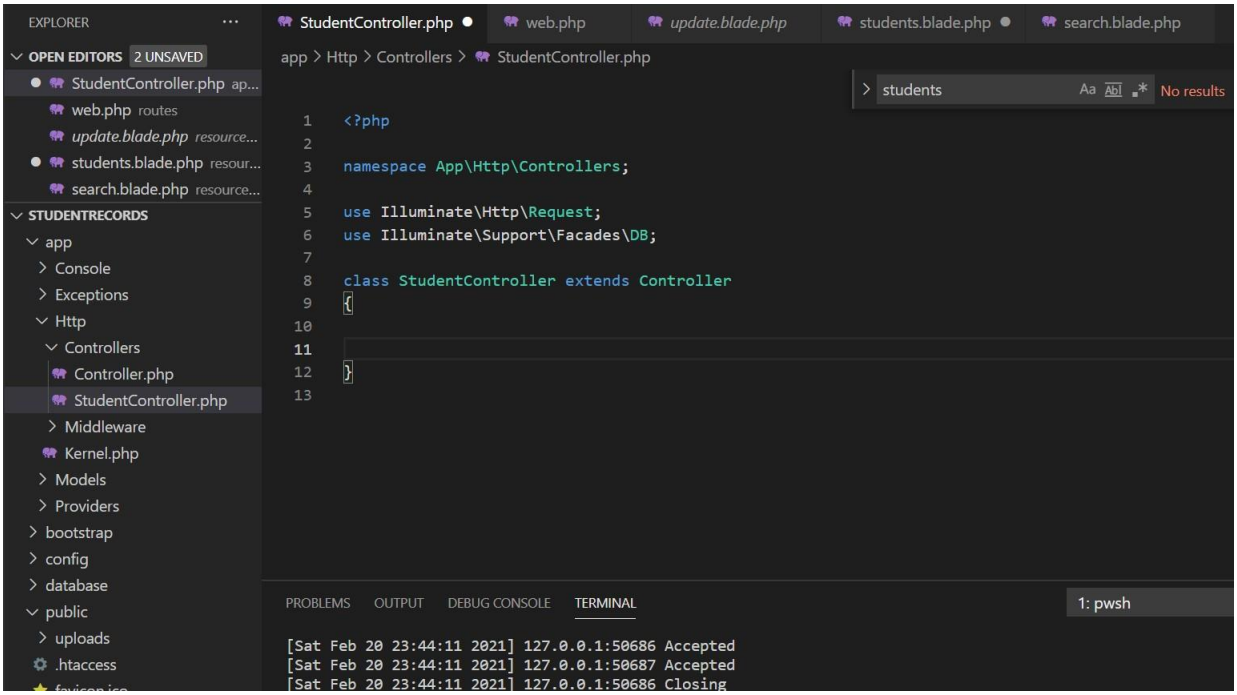
Route::get('base URI','controller@method');

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 StudenController.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) Graded Lab Tasks

Note: The instructor can design graded lab activities according to the level of difficult and complexity of the solved lab activities. The lab tasks assigned by the instructor should be evaluated in the same lab.

Lab Task 1

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

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

Lab Task 2

Define routes and views for the following Web pages of a company using Laravel Framework.
Home, Profile, Clients, Products, Contact Us.

Lab 12

Laravel Blade Templating

Objective:

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

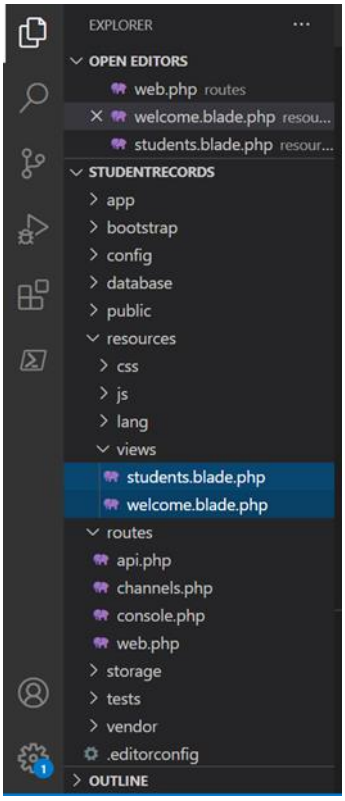
- Blade template in Laravel for creating different views in the Web Application.

Instructor Note:

As pre-lab activity, read Chapter 4 from the textbook "Laravel Up and Running, A Framework for Building Modern PHP Apps, 2nd Edition, Matt. Stauffer, Oreilly. Year 2019"

1) Useful Concepts

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']);

});
```

2) Solved Lab Activities

Sr. No	*Allocated Time*	*Level of Complexity*	*CLO Mapping*
Activity 1	*20 Minutes*	*High*	*CLO-5*
Activity 2	*20 Minutes*	*High*	*CLO-5*
Activity 3	*20 Minutes*	*High*	*CLO-5*

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:

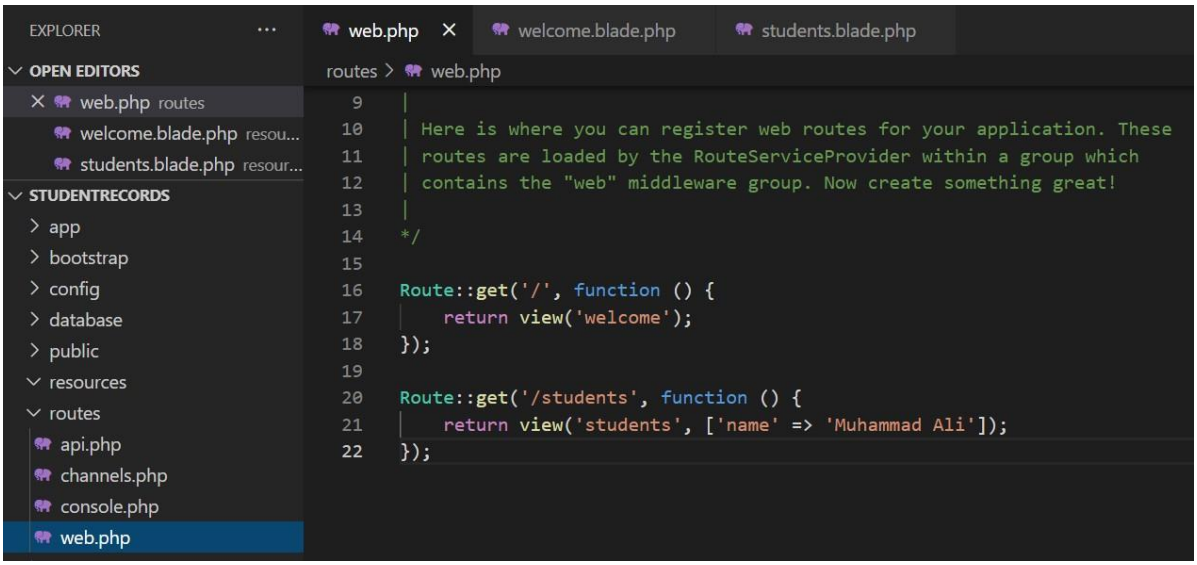
Solution:

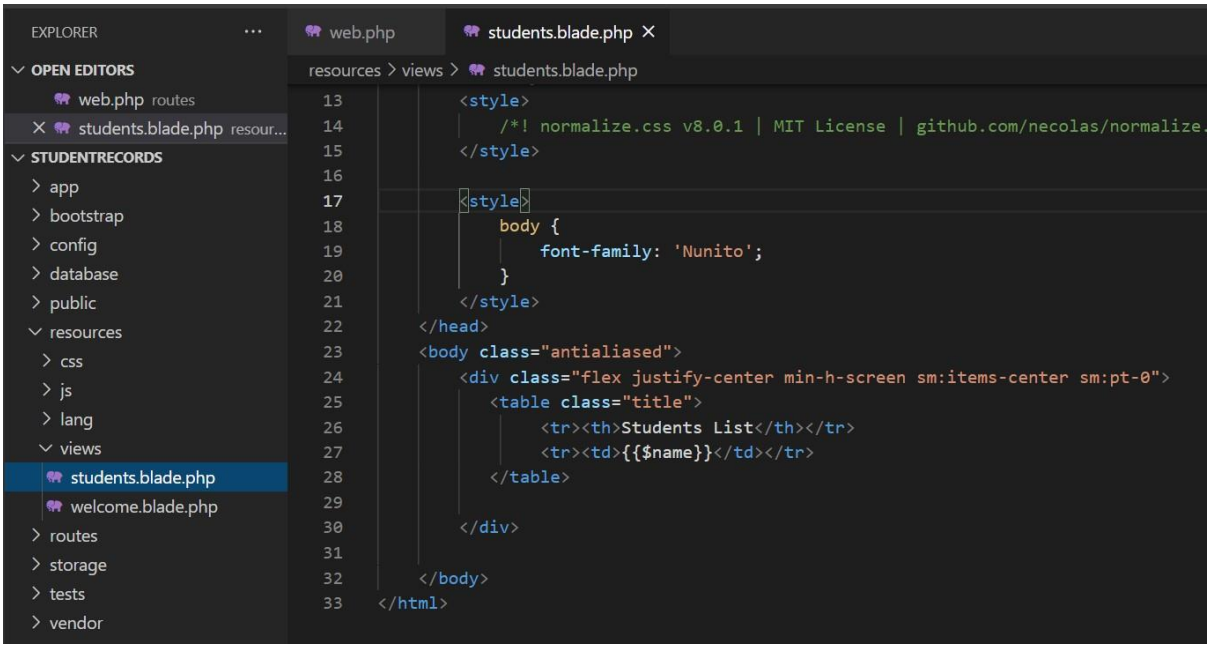
```
Route::get('/students', function () {
    return view('students', ['name' => 'Muhammad Ali']);
});
```

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:





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.

Solution:

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

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

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

Switch Statements

Switch statements can be constructed using the directives:

@switch, @case, @break, @default and @endswitch

```
@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.

Solution:

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>
```

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')
```

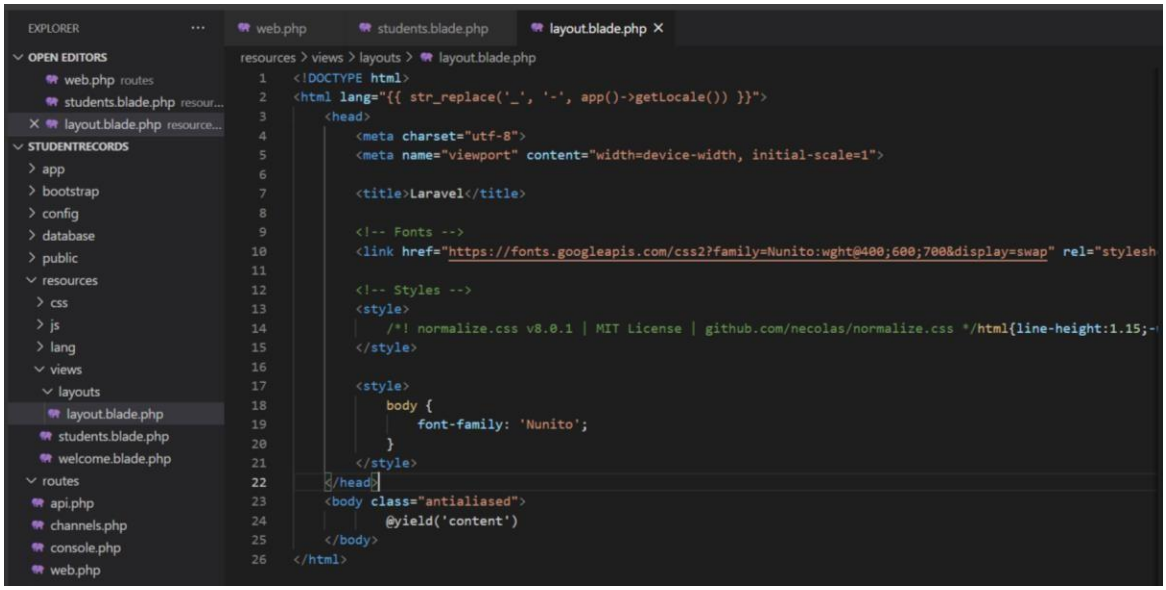
Using Template Inheritance in project ‘studentrecords’:

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

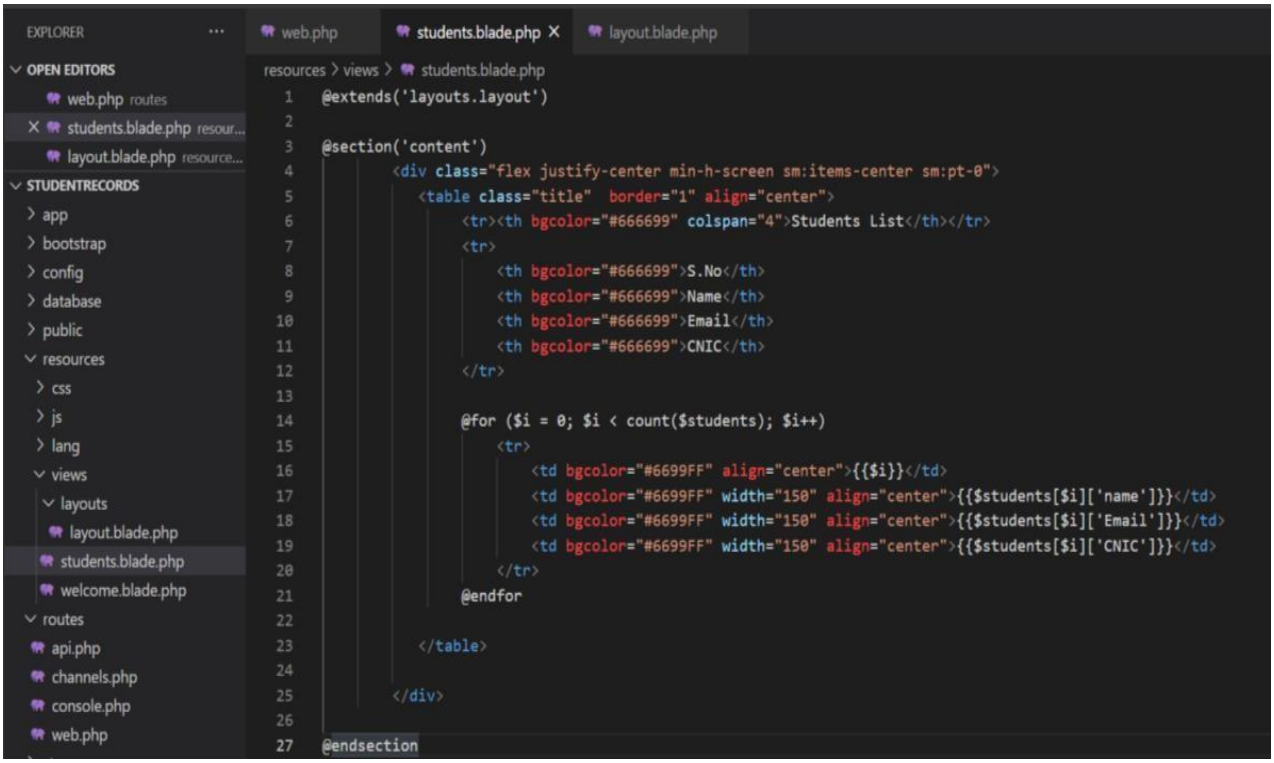
- welcome.blade.php
- students.blade.php

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 ‘studentsrecords’:

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



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.



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>
```

4. 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

3) Graded Lab Tasks

Note: The instructor can design graded lab activities according to the level of difficult and complexity of the solved lab activities. The lab tasks assigned by the instructor should be evaluated in the same lab.

Lab Task 1

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

Lab Task 2

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*

Lab 13

Laravel and MySQL

Objective:

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 MySQLdatabase using Laravel Framework.

Instructor Note:

As pre-lab activity, read Chapter 5 from the textbook "Laravel Up and Running, A Framework for Building Modern PHP Apps, 2nd Edition, Matt. Stauffer, Oreilly. Year 2019"

1) Useful Concepts

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)

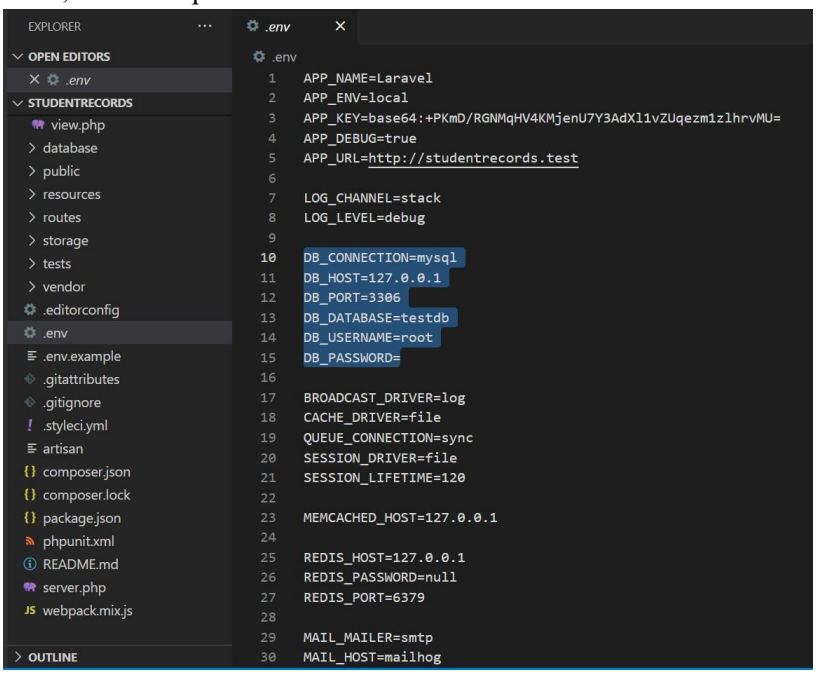
Database Connection

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.



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
<?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);
```

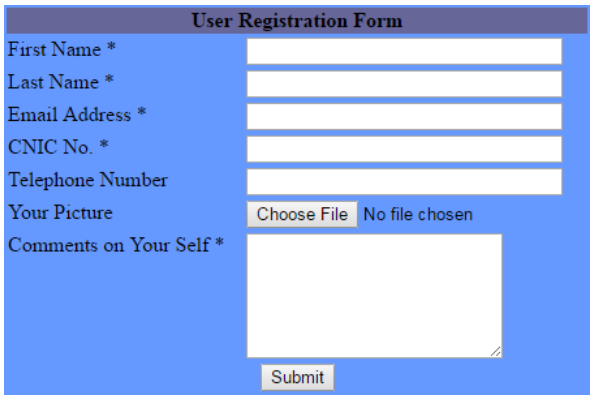
2) Solved Lab Activities

Sr. No	*Allocated Time*	*Level of Complexity*	*CLO Mapping*
Activity 1	*30 Minutes*	*High*	*CLO-5*
Activity 2	*30 Minutes*	*High*	*CLO-5*

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:

- ***Uploads the picture to the server (in uploads folder).***
- ***Inserts the user record in the database.***
- 

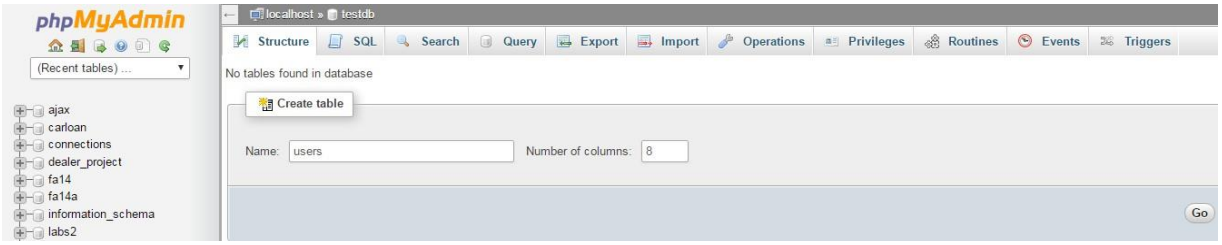
Solution:

Step 1:

Go to phpMyAdmin and create the database testdb:

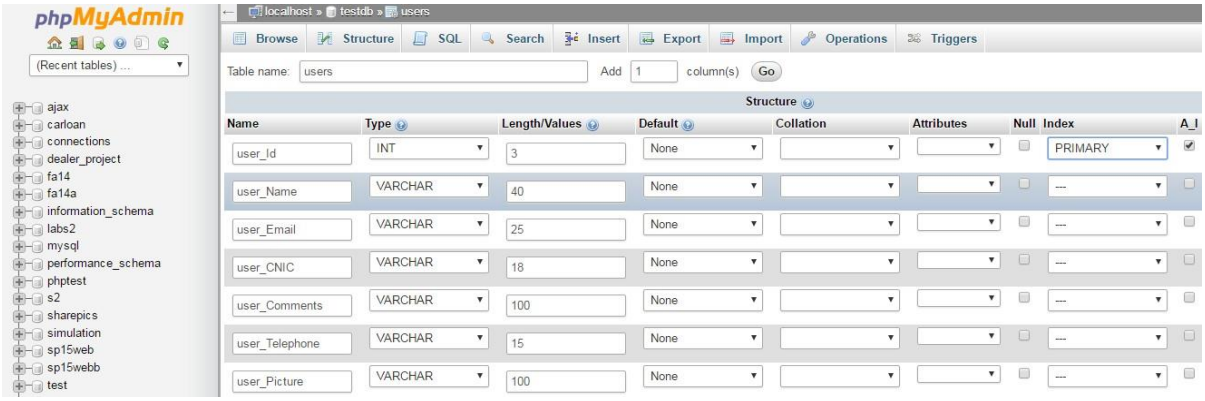
Step 2:

Create the table `users`:



Step 2:

Define Coloumns :



<u>Laravel Step 1</u>

Step 1:

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

External CSS

```
@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 valign="top"">
<label for="last_name">Last Name *</label>
</td>
<td valign="top">
<input type="text" name="last_name" maxlength="50" size="30">
</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">
</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">
</td>
</tr>
<tr>
<td valign="top">
<label for="telephone">Telephone Number</label>
</td>
<td valign="top">
<input type="text" name="telephone" maxlength="30" size="30">
</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"></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:

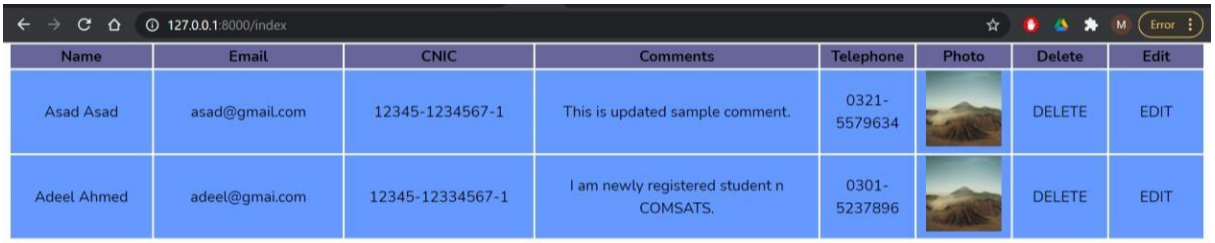
Visit the following URL to insert record in database.

http://127.0.0.1:8000/students/create

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

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</td>
<th bgcolor="#666699">Email</td>
<th bgcolor="#666699">CNIC</td>
<th bgcolor="#666699">Comments</td>
<th bgcolor="#666699">Telephone</td>
<th bgcolor="#666699">Photo</td>
<th bgcolor="#666699">Delete</td>
<th bgcolor="#666699">Edit</td>
</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 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@gmai.com	12345-12334567-1	I am newly registered student n COMSATS.	0301-5237896		DELETE	EDIT

3) Graded Lab Tasks

Note: The instructor can design graded lab activities according to the level of difficult and complexity of the solved lab activities. The lab tasks assigned by the instructor should be evaluated in the same lab.

Lab Task 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.

Lab Task 2

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

1. *Profile*
2. *Contact Us*

Lab 14

CRUD Operations

Objective:

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

- Insert, retrieve, delete and update records in MySQL using Laravel Framework.

Instructor Note:

As pre-lab activity, read Chapter 5 from the textbook "Laravel Up and Running, A Framework for Building Modern PHP Apps, 2nd Edition, Matt. Stauffer, Oreilly. Year 2019"

1) Useful Concepts

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:

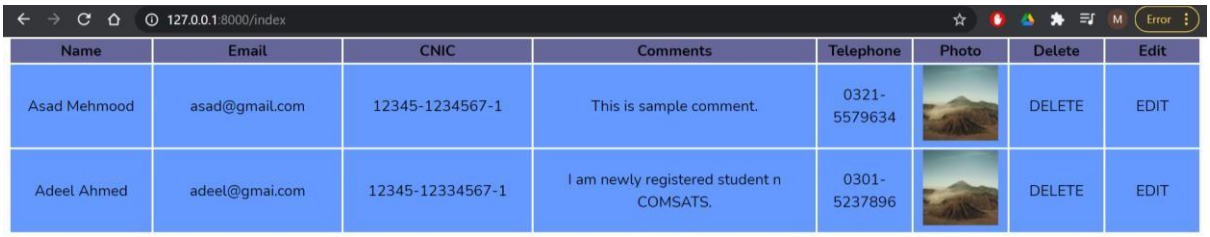
```
use Illuminate\Support\Facades\DB;
$deleted = DB::delete('delete from users');
```

2) Solved Lab Activities

Sr. No	*Allocated Time*	*Level of Complexity*	*CLO Mapping*
Activity 1	*30 Minutes*	*High*	*CLO-5*
Activity 2	*30 Minutes*	*High*	*CLO-5*

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@gmai.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</td>
<th bgcolor="#666699">Email</td>
<th bgcolor="#666699">CNIC</td>
<th bgcolor="#666699">Comments</td>
<th bgcolor="#666699">Telephone</td>
<th bgcolor="#666699">Photo</td>
<th bgcolor="#666699">Delete</td>
<th bgcolor="#666699">Edit</td>
</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');
```

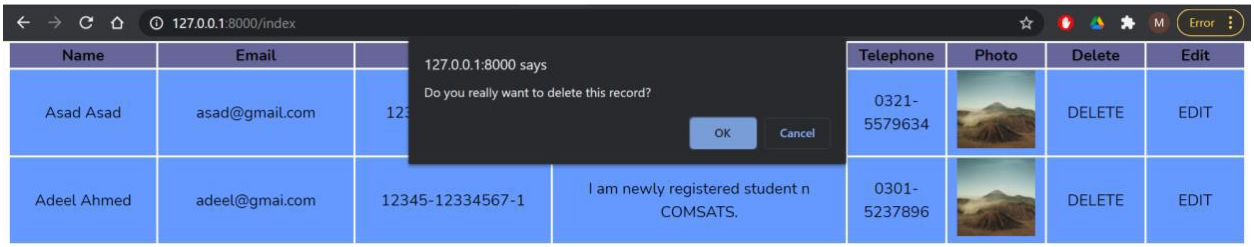
Step 3:

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="30" 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" value ={{
$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" value ={{
$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="30" 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**'

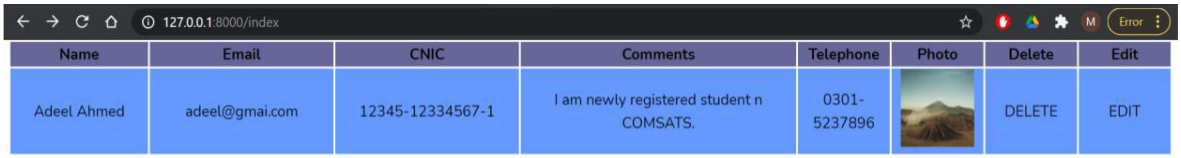
```
Route::post('/students/update/{id}','App\Http\Controllers\StudentController@displ
ay');
```

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');
            }
```

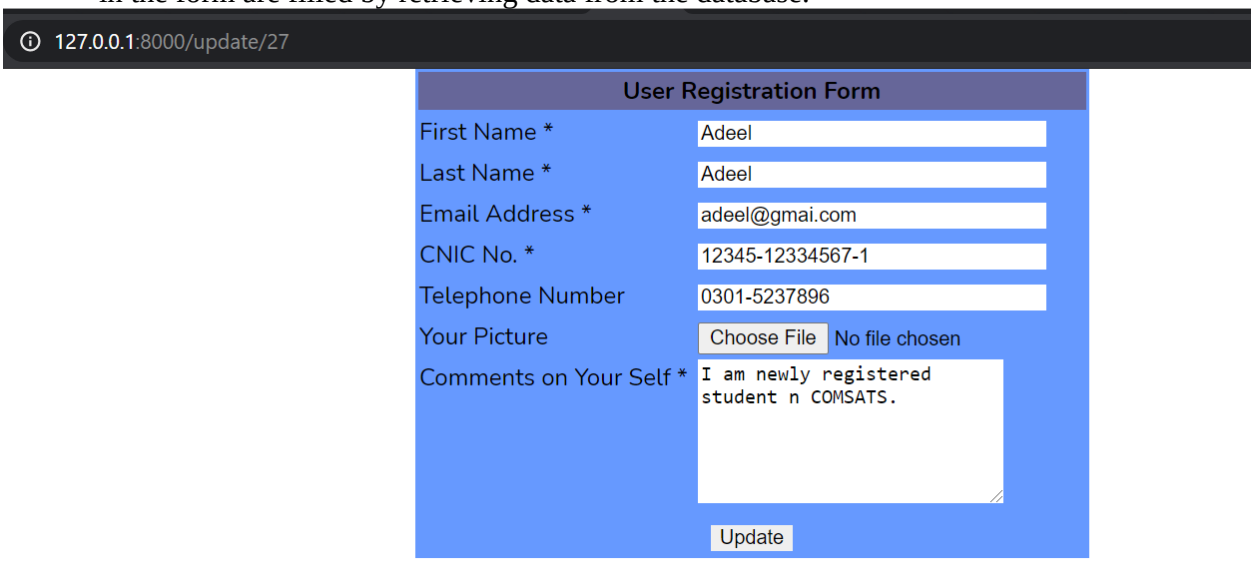
Step 5:

Open the following URL: **http://127.0.0.1:8000/index**.



Step 6:

1. 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.



Step 7:

1. 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.

3) Graded Lab Tasks

Note: The instructor can design graded lab activities according to the level of difficult and complexity of the solved lab activities. The lab tasks assigned by the instructor should be evaluated in the same lab.

Lab Task 1

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