Diploma in

IT, Networking and Cloud

Module 3

Backend Web Technologies & Frameworks

Lab Manual

Disclaimer: The content is curated for educational purposes only.

Table of Contents

Learning Outcome

After completing this module, the student should be able to configure an embedded database with different web pages using XAMPP and MySQL
To meet the learning outcome, a student must complete the following activities

1. Install using XAMPP software package for MySQL, PHP and other utilities (1Hr)
2. Connect to MySQL database system and create new database (1Hr)
3. Create tables in new database and enter sample data (1Hr)
4. Import data into database table from CSV file (1Hr)
5. Read data from database table, all data, filtered data based on SQL queries (2Hr)
6. Update single record, and multiple records based on criteria (2Hr)
7. Delete selected records from table (1Hr)
8. Work with SQL stored procedures (1Hr)
9. Perform transaction in database tables using commit and rollback operations (5Hr)
10. Demonstrate use of transaction control in relational databases (5Hr)
11. Perform user management activities in MySQL database, create, modify user, set privileges (5Hr)

Activity 1

Aim: Install using XAMPP software package for MySQL, PHP and other utilities

Learning outcome: Able to configure embedded databases with different web pages using XAMPP and MySQL.

Duration: 1 hour

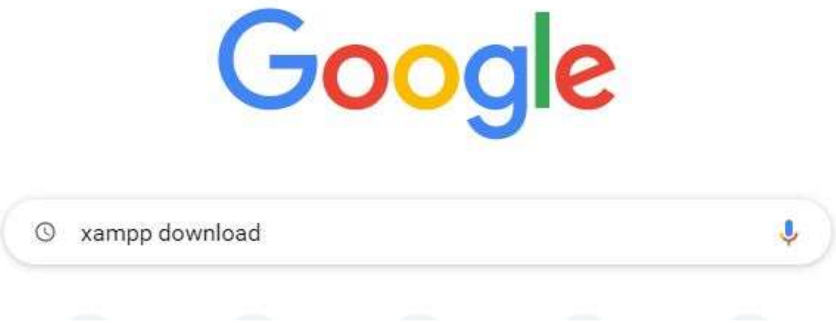
List of Hardware/Software requirements:

1. Laptop/Computer with Windows 10/11
2. XAMPP software
3.

Installation Steps of XAMPP Software:

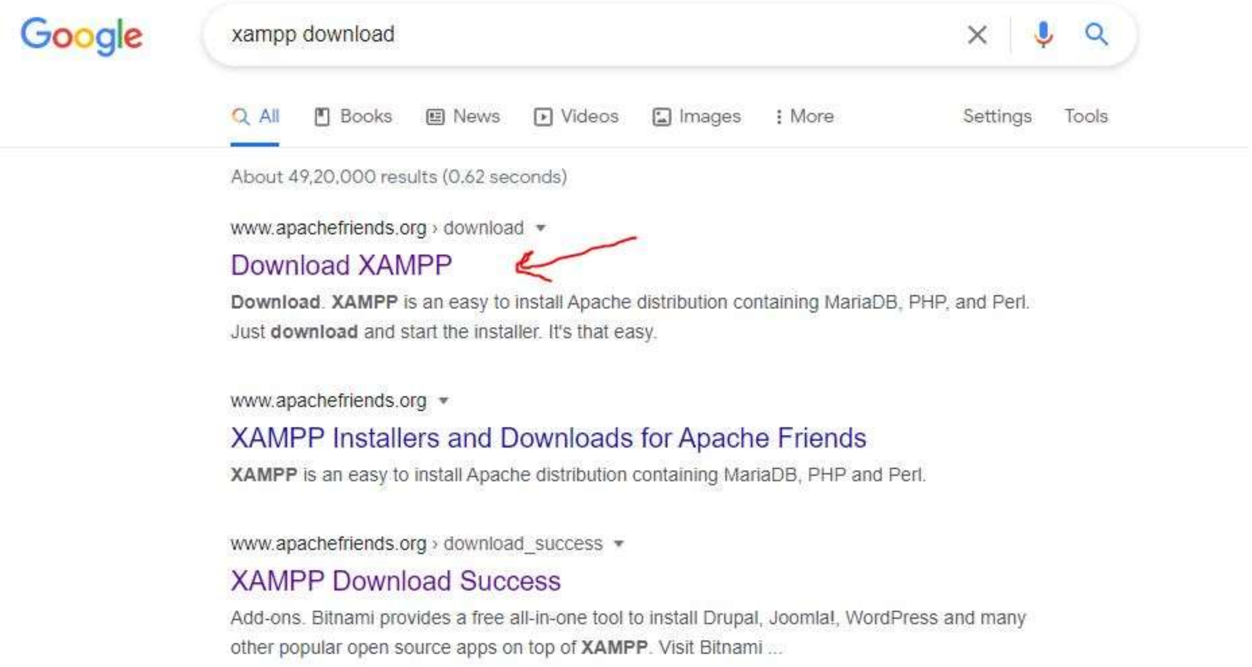
Steps to install XAMPP On Windows: -

Step 1: - Go to the Google and search “XAMPP Download”.



xampp download

Step 2: - Choose the first one www.apachefriends.org , please click on this and you will get the site for download.



About 49,20,000 results (0.62 seconds)

www.apachefriends.org › download

Download XAMPP

Download. **XAMPP** is an easy to install Apache distribution containing MariaDB, PHP, and Perl. Just **download** and start the installer. It's that easy.

www.apachefriends.org

XAMPP Installers and Downloads for Apache Friends

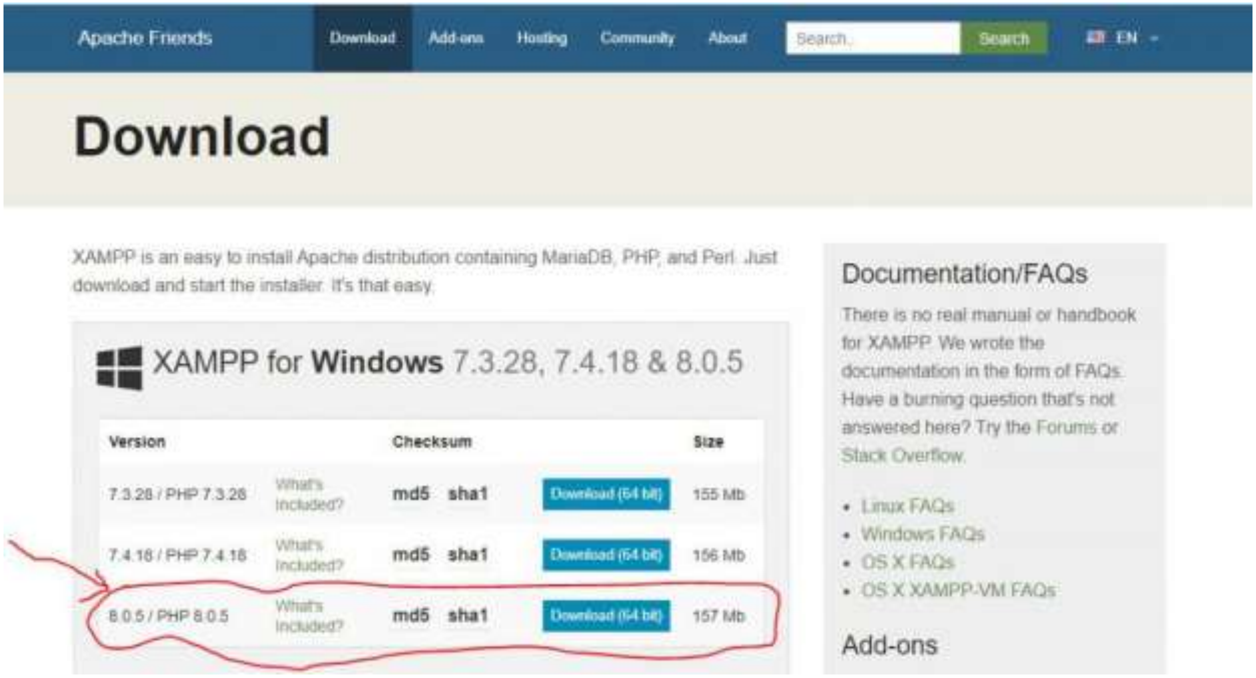
XAMPP is an easy to install Apache distribution containing MariaDB, PHP and Perl.

www.apachefriends.org › download_success

XAMPP Download Success

Add-ons. Bitnami provides a free all-in-one tool to install Drupal, Joomla!, WordPress and many other popular open source apps on top of **XAMPP**. Visit Bitnami ...

Step 3: - please select the OS which you are using Windows, Linux, and Mac OS X. and click on the latest version of XAMPP to Download.



Download

XAMPP is an easy to install Apache distribution containing MariaDB, PHP, and Perl. Just download and start the installer. It's that easy.

XAMPP for **Windows** 7.3.28, 7.4.18 & 8.0.5

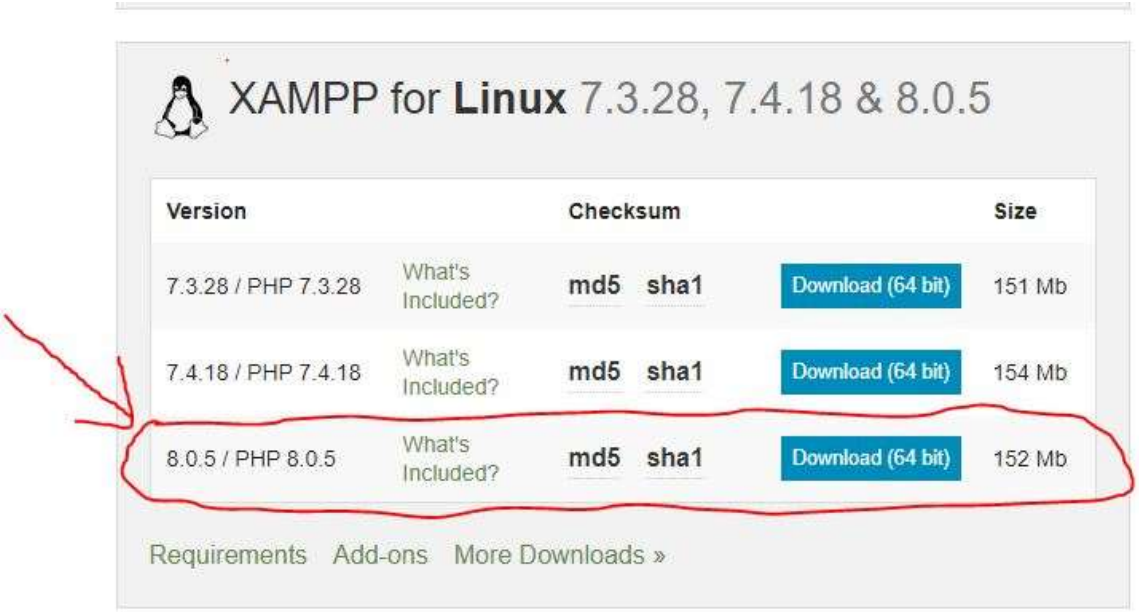
Version		Checksum			Size
7.3.28 / PHP 7.3.28	What's Included?	md5	sha1	Download (64 bit)	155 Mb
7.4.18 / PHP 7.4.18	What's Included?	md5	sha1	Download (64 bit)	156 Mb
8.0.5 / PHP 8.0.5	What's Included?	md5	sha1	Download (64 bit)	157 Mb

Documentation/FAQs

There is no real manual or handbook for XAMPP. We wrote the documentation in the form of FAQs. Have a burning question that's not answered here? Try the Forums or Stack Overflow.

- Linux FAQs
- Windows FAQs
- OS X FAQs
- OS X XAMPP-VM FAQs

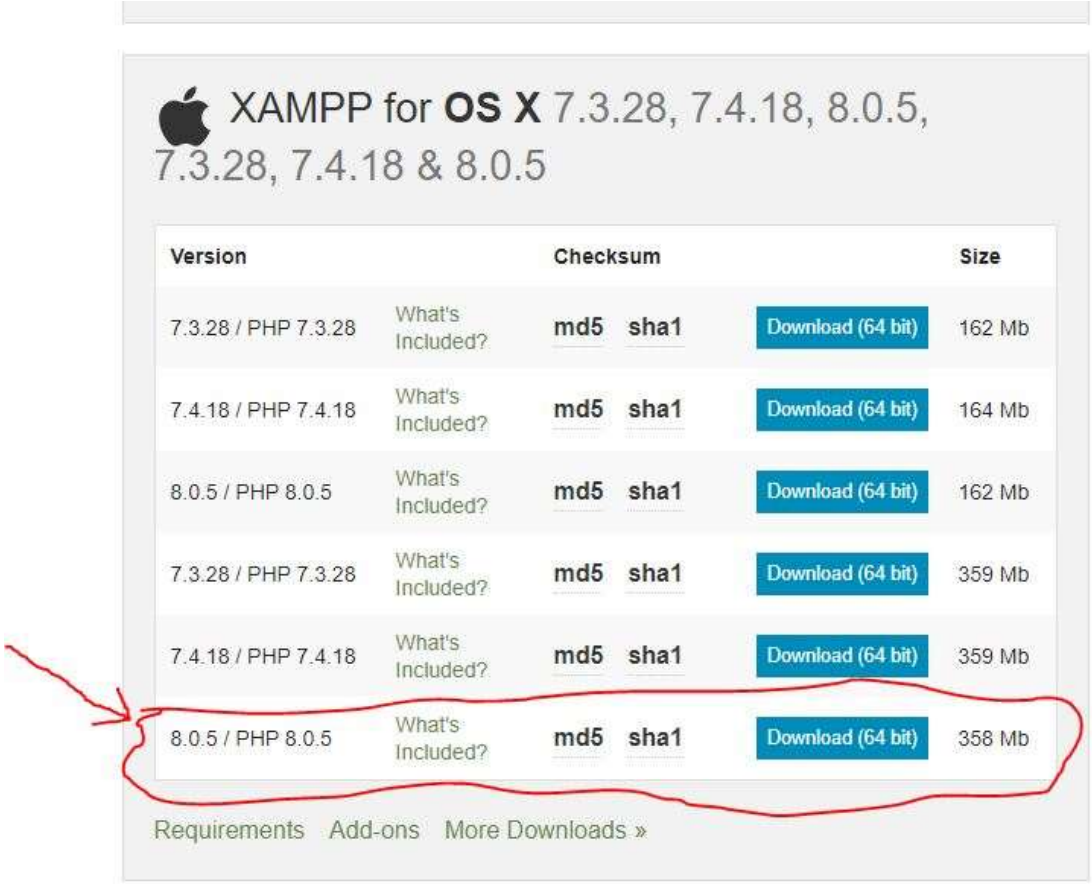
Add-ons



XAMPP for **Linux** 7.3.28, 7.4.18 & 8.0.5

Version		Checksum			Size
7.3.28 / PHP 7.3.28	What's Included?	md5	sha1	Download (64 bit)	151 Mb
7.4.18 / PHP 7.4.18	What's Included?	md5	sha1	Download (64 bit)	154 Mb
8.0.5 / PHP 8.0.5	What's Included?	md5	sha1	Download (64 bit)	152 Mb

Requirements Add-ons More Downloads »

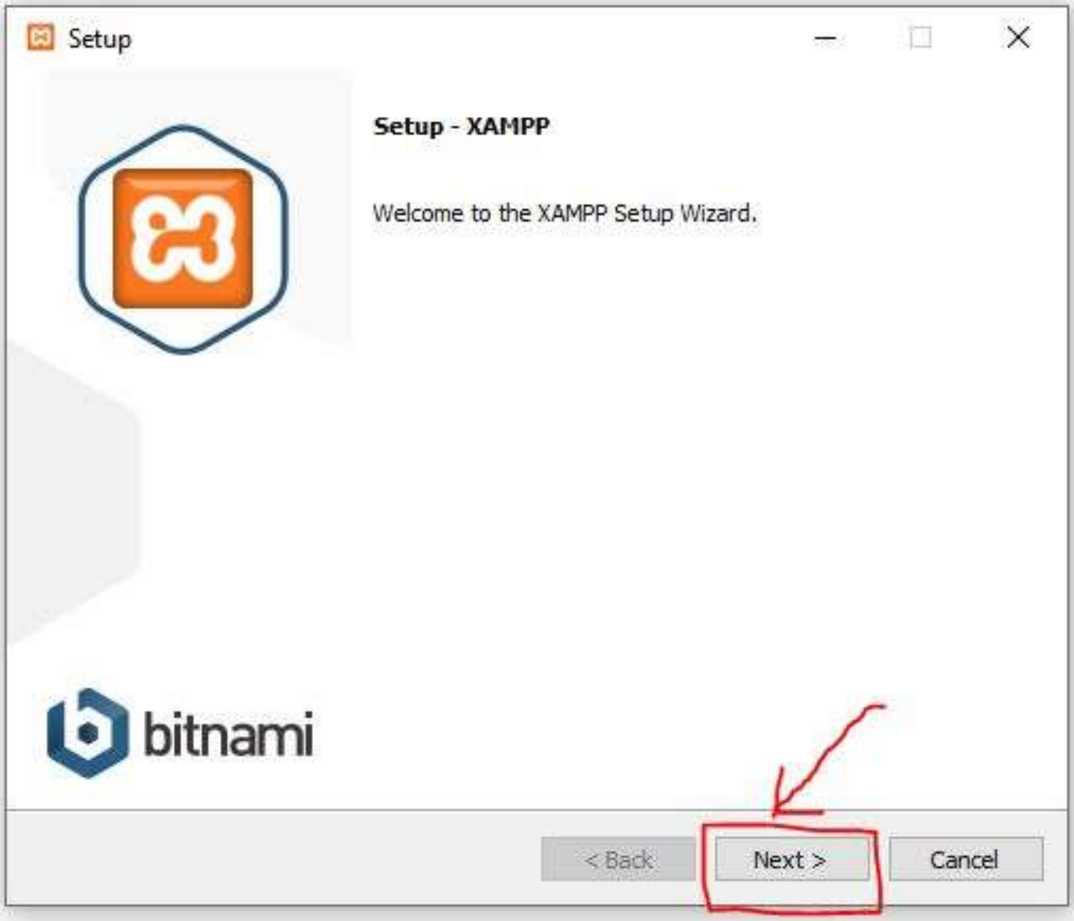


XAMPP for **OS X** 7.3.28, 7.4.18, 8.0.5, 7.3.28, 7.4.18 & 8.0.5

Version		Checksum			Size
7.3.28 / PHP 7.3.28	What's Included?	md5	sha1	Download (64 bit)	162 Mb
7.4.18 / PHP 7.4.18	What's Included?	md5	sha1	Download (64 bit)	164 Mb
8.0.5 / PHP 8.0.5	What's Included?	md5	sha1	Download (64 bit)	162 Mb
7.3.28 / PHP 7.3.28	What's Included?	md5	sha1	Download (64 bit)	359 Mb
7.4.18 / PHP 7.4.18	What's Included?	md5	sha1	Download (64 bit)	359 Mb
8.0.5 / PHP 8.0.5	What's Included?	md5	sha1	Download (64 bit)	358 Mb

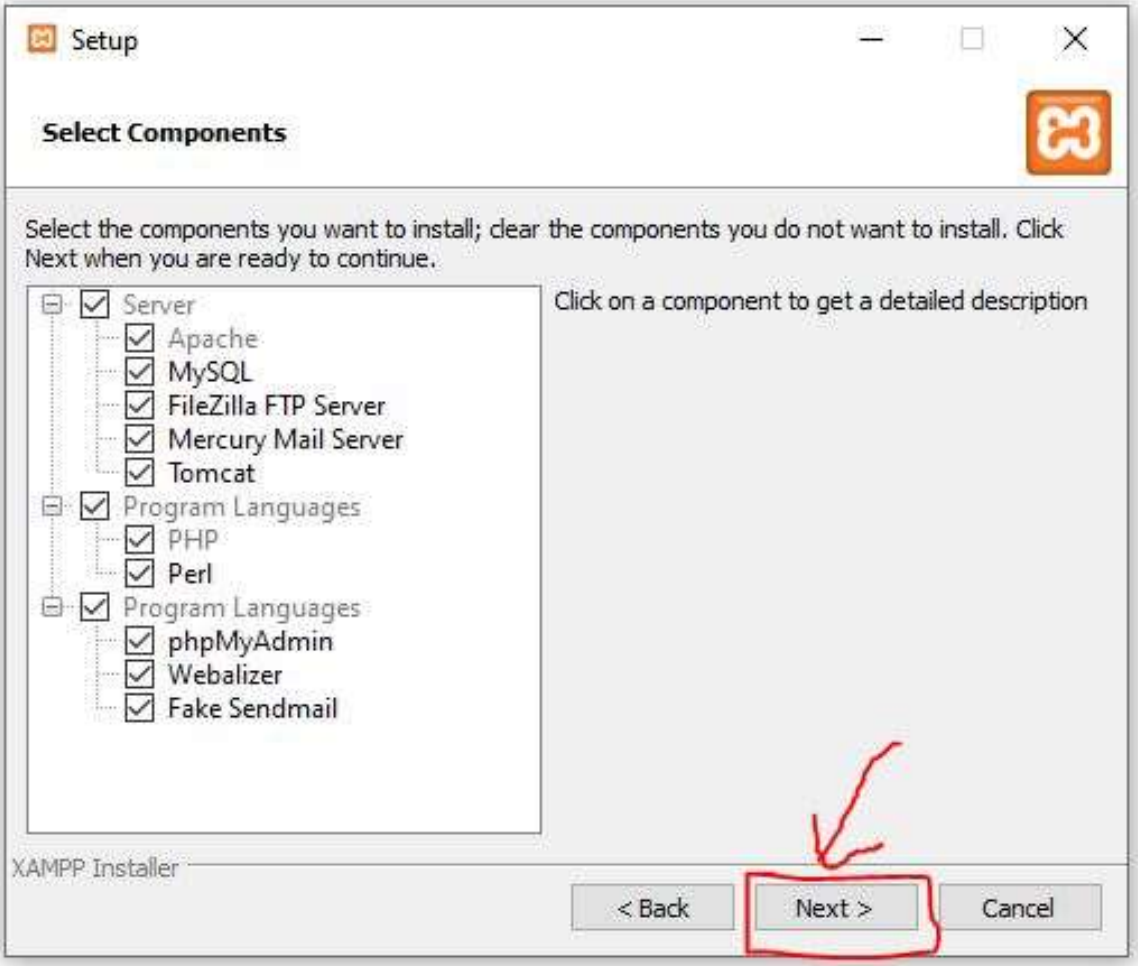
Requirements Add-ons More Downloads »

Step 4:- Now open the Downloaded File to Install XAMPP. Complete the Setup according to the given Image.



Setup - XAMPP

Welcome to the XAMPP Setup Wizard.



Setup

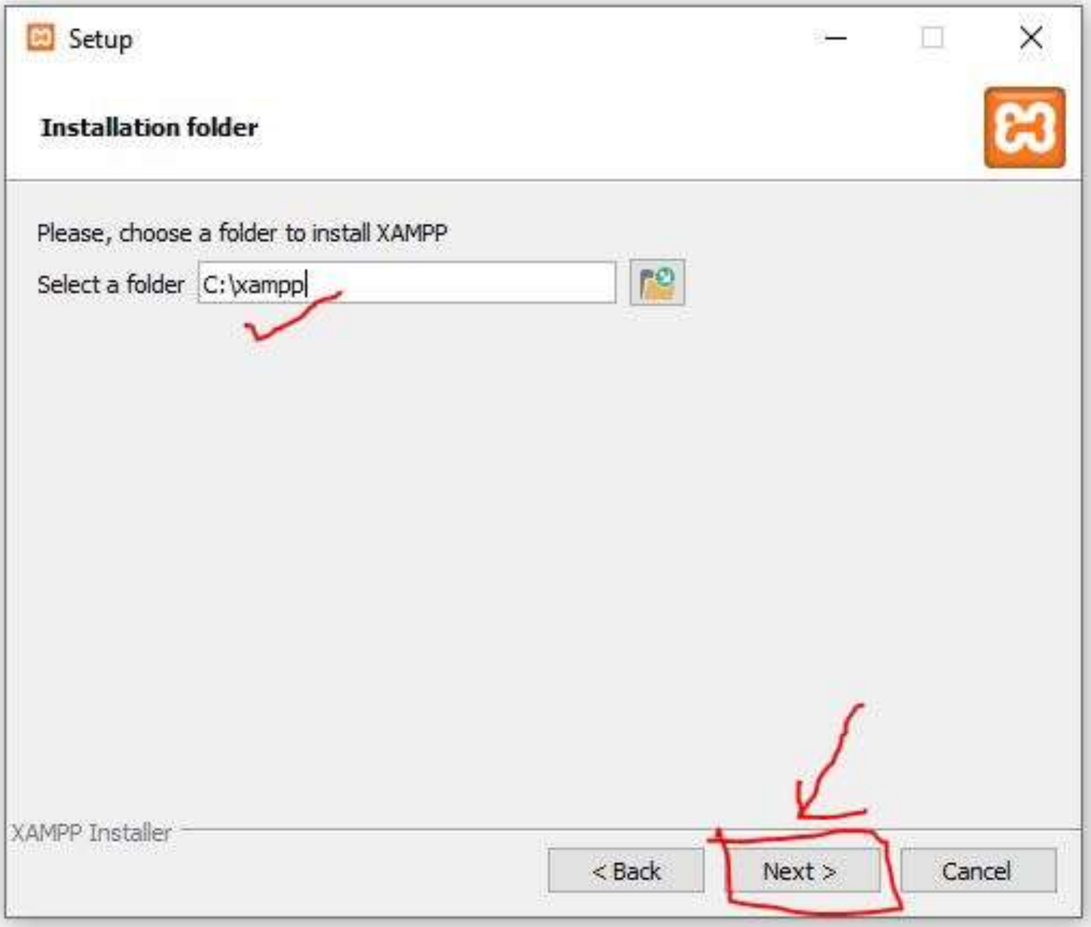
Select Components

Select the components you want to install; clear the components you do not want to install. Click Next when you are ready to continue.

- ☑ Server
  - ☑ Apache
  - ☑ MySQL
  - ☑ FileZilla FTP Server
  - ☑ Mercury Mail Server
  - ☑ Tomcat
- ☑ Program Languages
  - ☑ PHP
  - ☑ Perl
- ☑ Program Languages
  - ☑ phpMyAdmin
  - ☑ Webalizer
  - ☑ Fake Sendmail

Click on a component to get a detailed description

XAMPP Installer



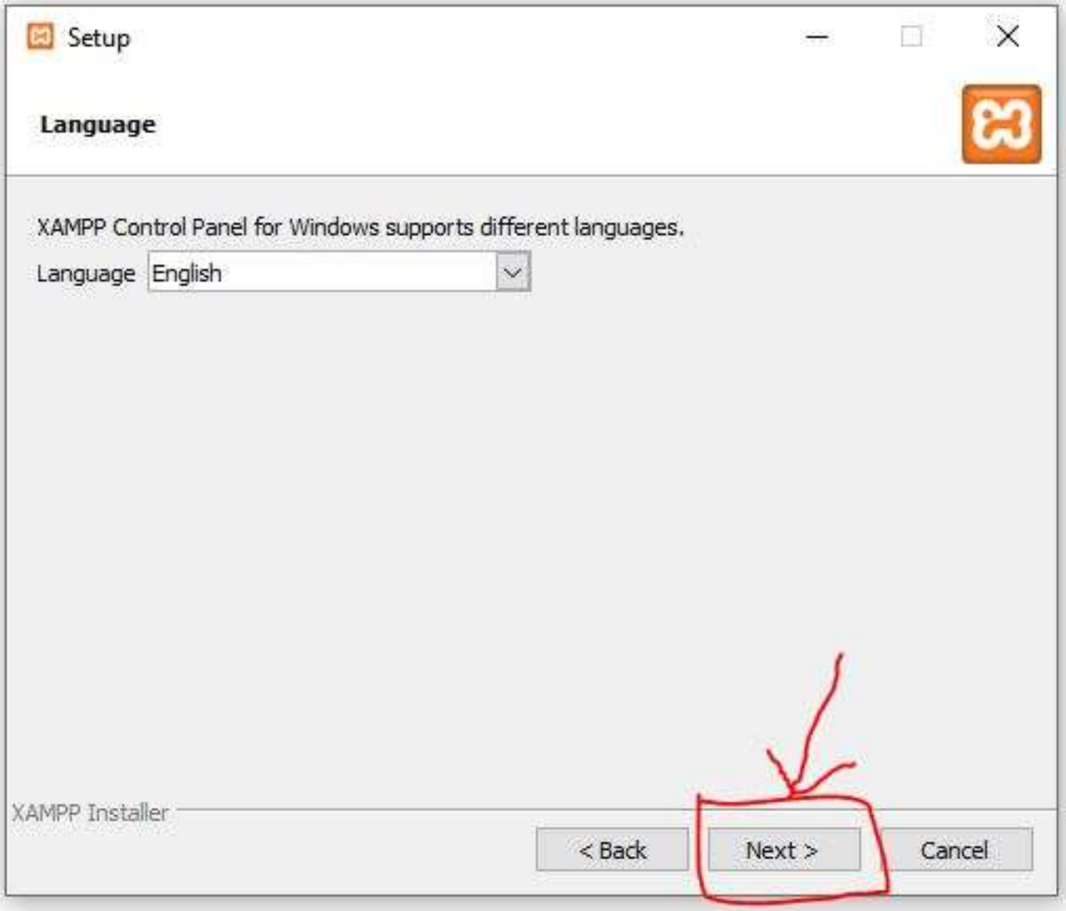
Setup

Installation folder

Please, choose a folder to install XAMPP

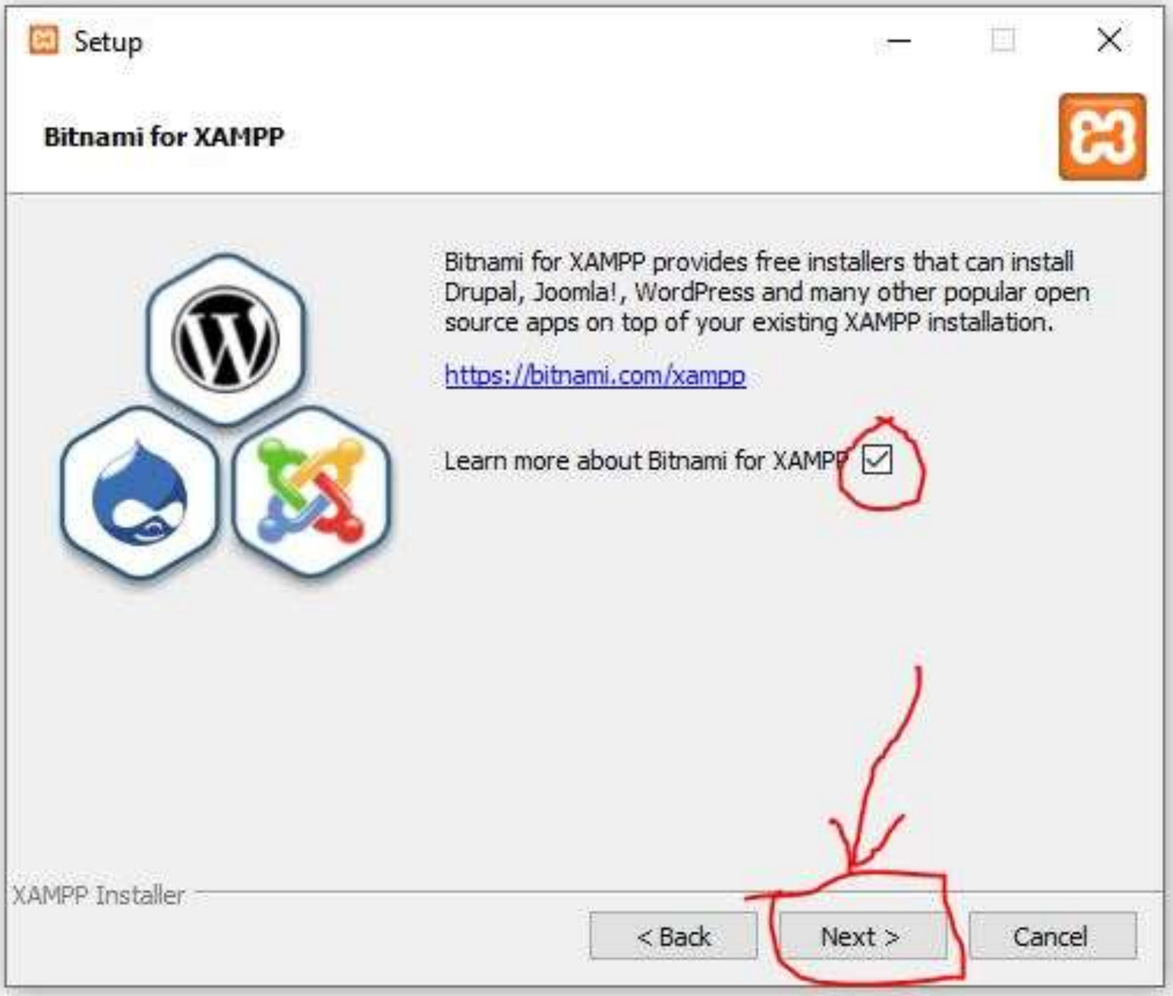
Select a folder C:\xampp

XAMPP Installer



XAMPP Control Panel for Windows supports different languages.

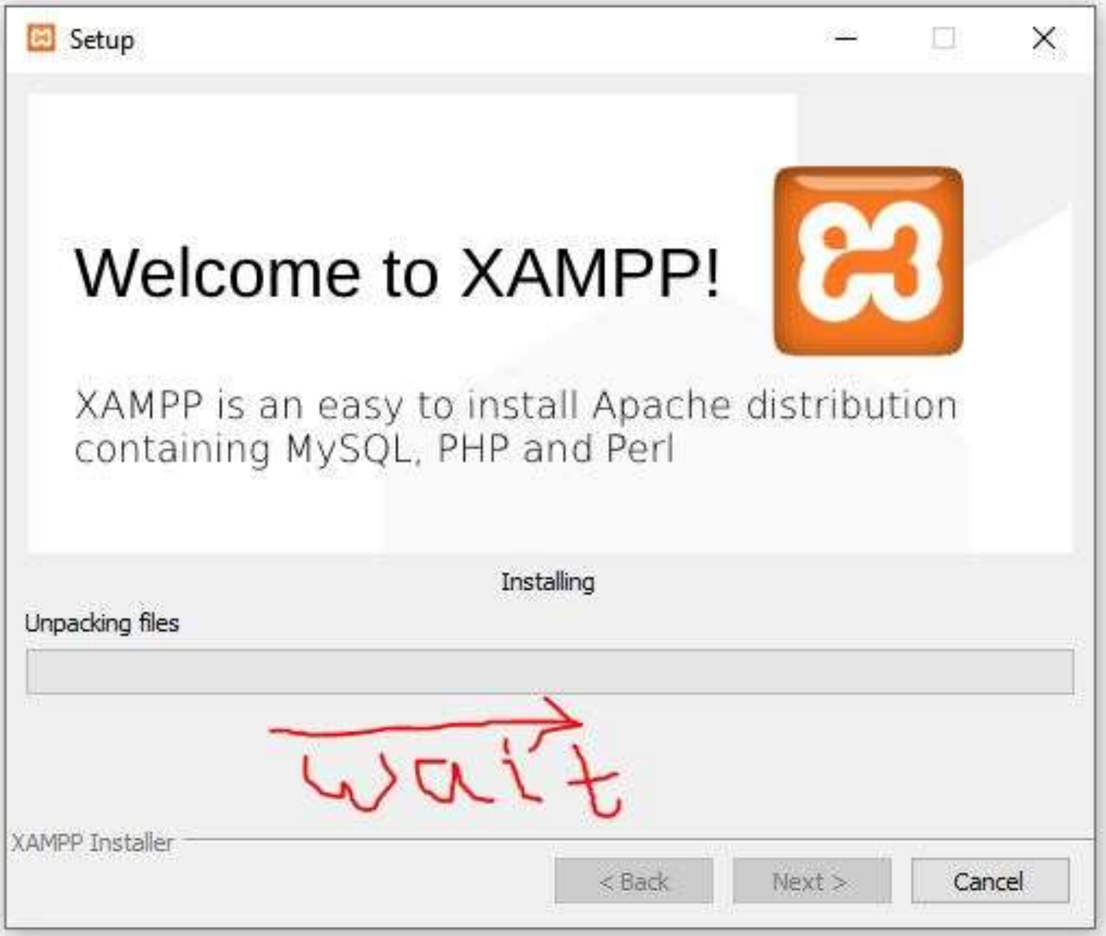
Language: English



Bitnami for XAMPP provides free installers that can install Drupal, Joomla!, WordPress and many other popular open source apps on top of your existing XAMPP installation.

https://bitnami.com/xampp

Learn more about Bitnami for XAMPP ☑



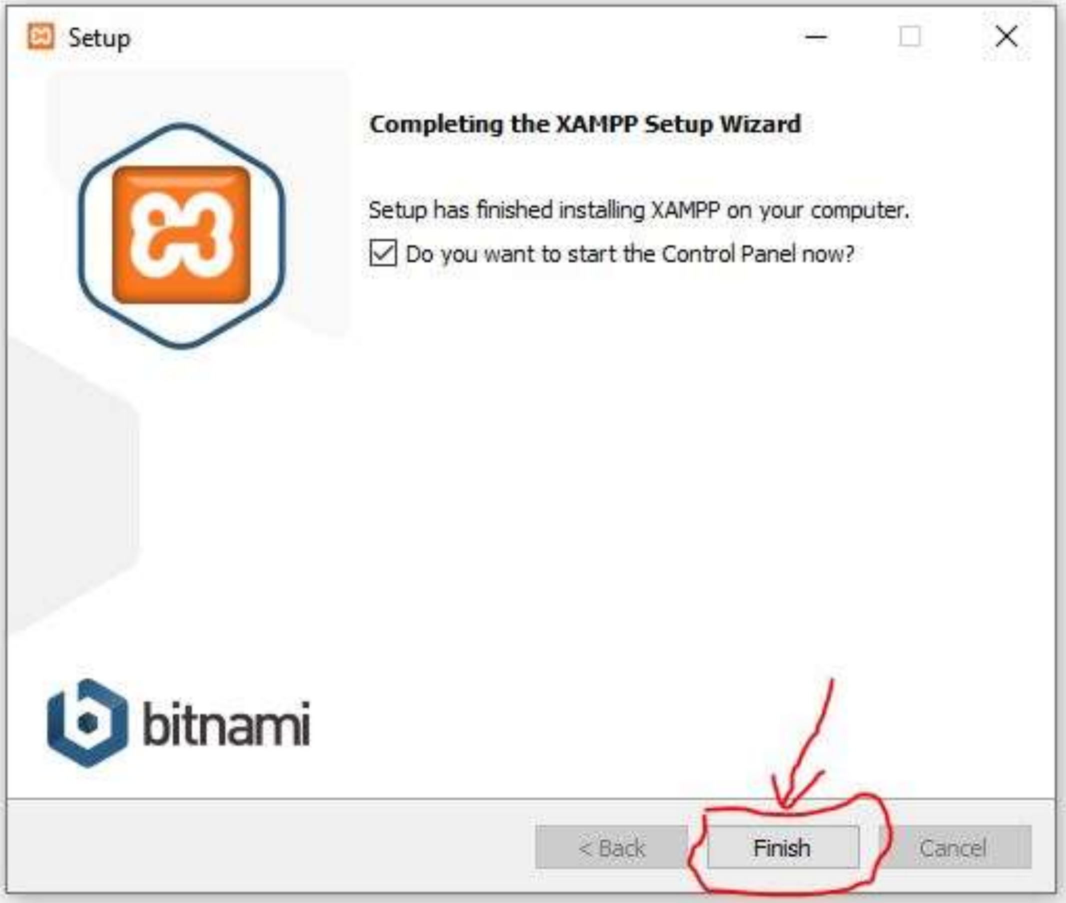
Welcome to XAMPP!

XAMPP is an easy to install Apache distribution containing MySQL, PHP and Perl

Installing

Unpacking files

wait

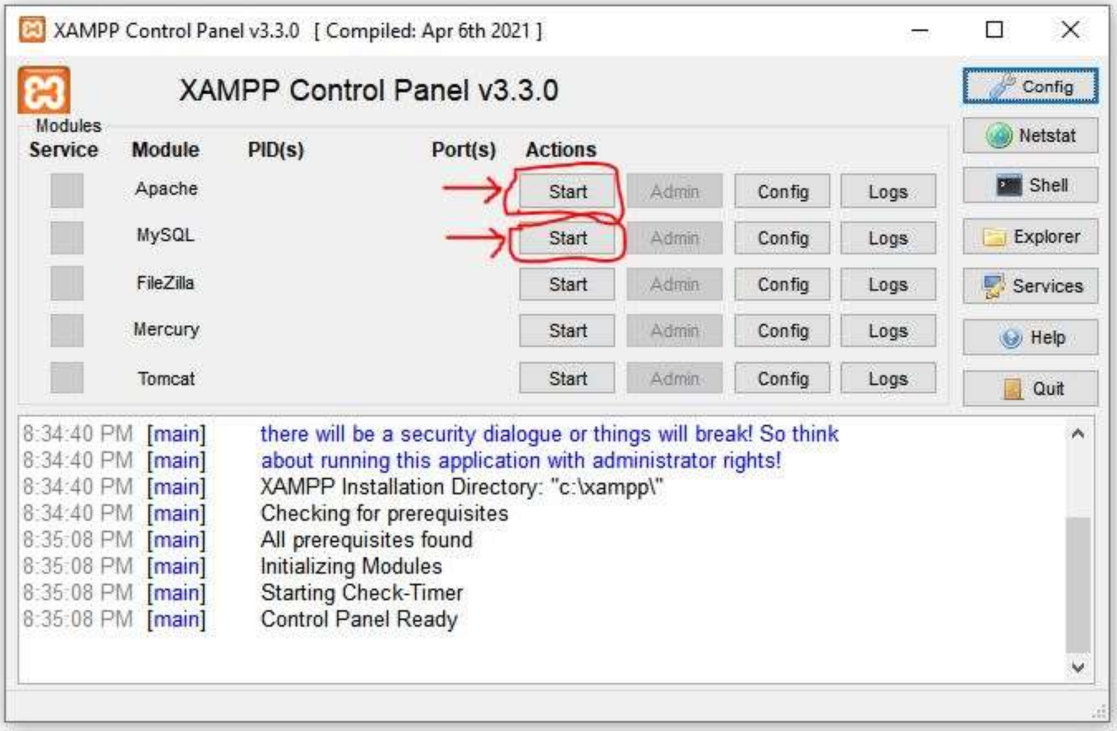


Completing the XAMPP Setup Wizard

Setup has finished installing XAMPP on your computer.

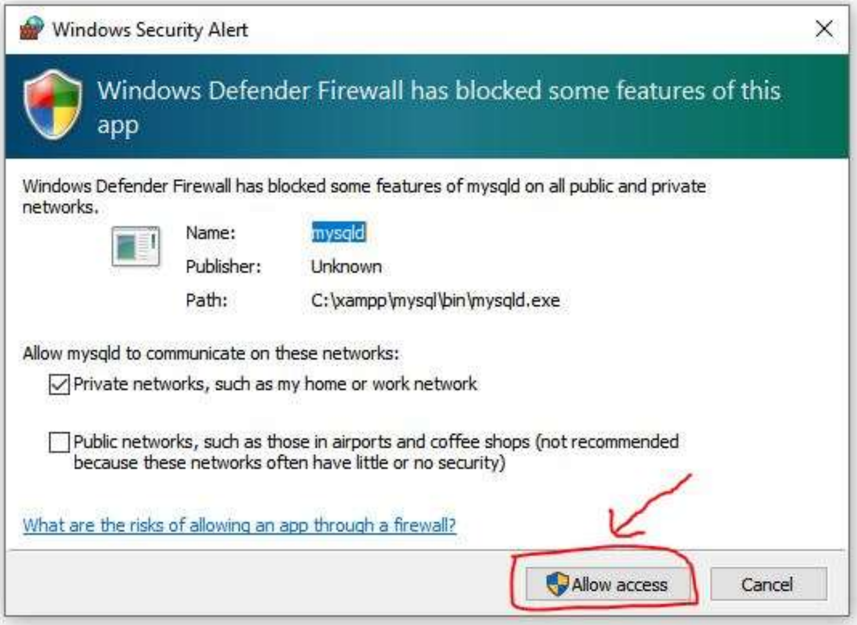
☑ Do you want to start the Control Panel now?

Step 5:- Then open the XAMPP software after installing it will look like this given Image. It Contains XAMPP Control Panel. Please click on the start button to Run the Server as given in picture.


XAMPP Control Panel v3.3.0 [Compiled: Apr 6th 2021]
XAMPP Control Panel v3.3.0
Config
Netstat
Shell
Explorer
Services
Help
Quit

Modules Service	Module	PID(s)	Port(s)	Actions			
□	Apache			Start	Admin	Config	Logs
□	MySQL			Start	Admin	Config	Logs
□	FileZilla			Start	Admin	Config	Logs
□	Mercury			Start	Admin	Config	Logs
□	Tomcat			Start	Admin	Config	Logs

```
8:34:40 PM  [main]   there will be a security dialogue or things will break! So think
8:34:40 PM  [main]   about running this application with administrator rights!
8:34:40 PM  [main]   XAMPP Installation Directory: "c:\xampp\"
8:34:40 PM  [main]   Checking for prerequisites
8:35:08 PM  [main]   All prerequisites found
8:35:08 PM  [main]   Initializing Modules
8:35:08 PM  [main]   Starting Check-Timer
8:35:08 PM  [main]   Control Panel Ready
```



Windows Defender Firewall has blocked some features of this app

Windows Defender Firewall has blocked some features of mysqld on all public and private networks.

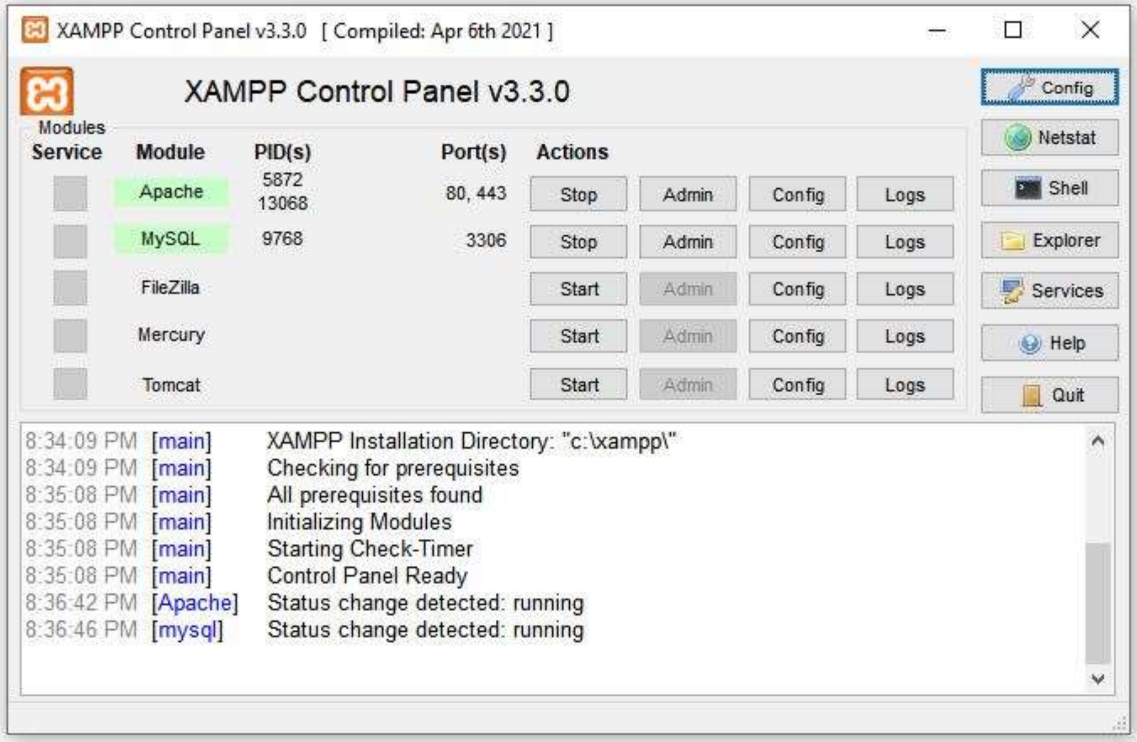
Name: mysqld
Publisher: Unknown
Path: C:\xampp\mysql\bin\mysqld.exe

Allow mysqld to communicate on these networks:

☑ Private networks, such as my home or work network

□ Public networks, such as those in airports and coffee shops (not recommended because these networks often have little or no security)

What are the risks of allowing an app through a firewall?



XAMPP Control Panel v3.3.0 [Compiled: Apr 6th 2021]

XAMPP Control Panel v3.3.0

Modules

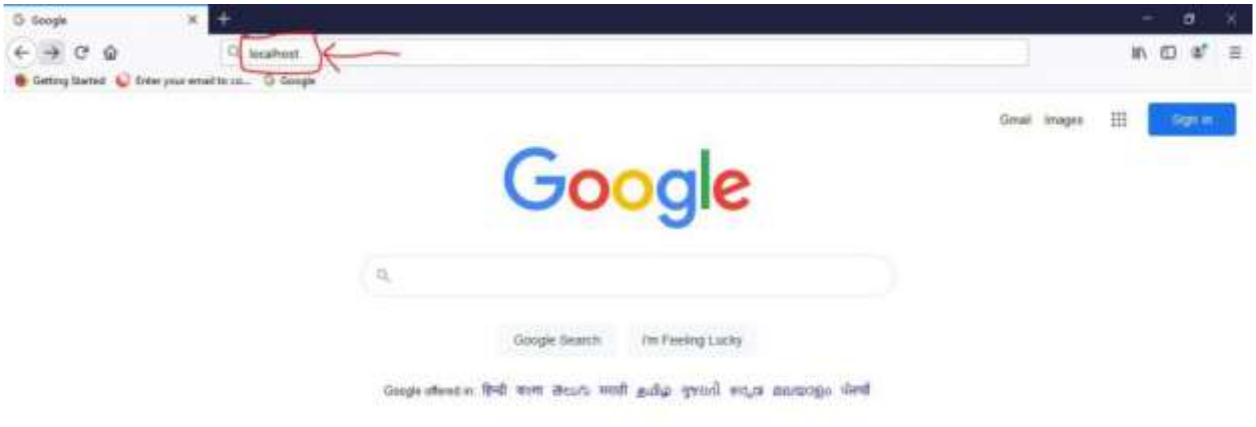
Service	Module	PID(s)	Port(s)	Actions			
	Apache	5872 13068	80, 443	Stop	Admin	Config	Logs
	MySQL	9768	3306	Stop	Admin	Config	Logs
	FileZilla			Start	Admin	Config	Logs
	Mercury			Start	Admin	Config	Logs
	Tomcat			Start	Admin	Config	Logs

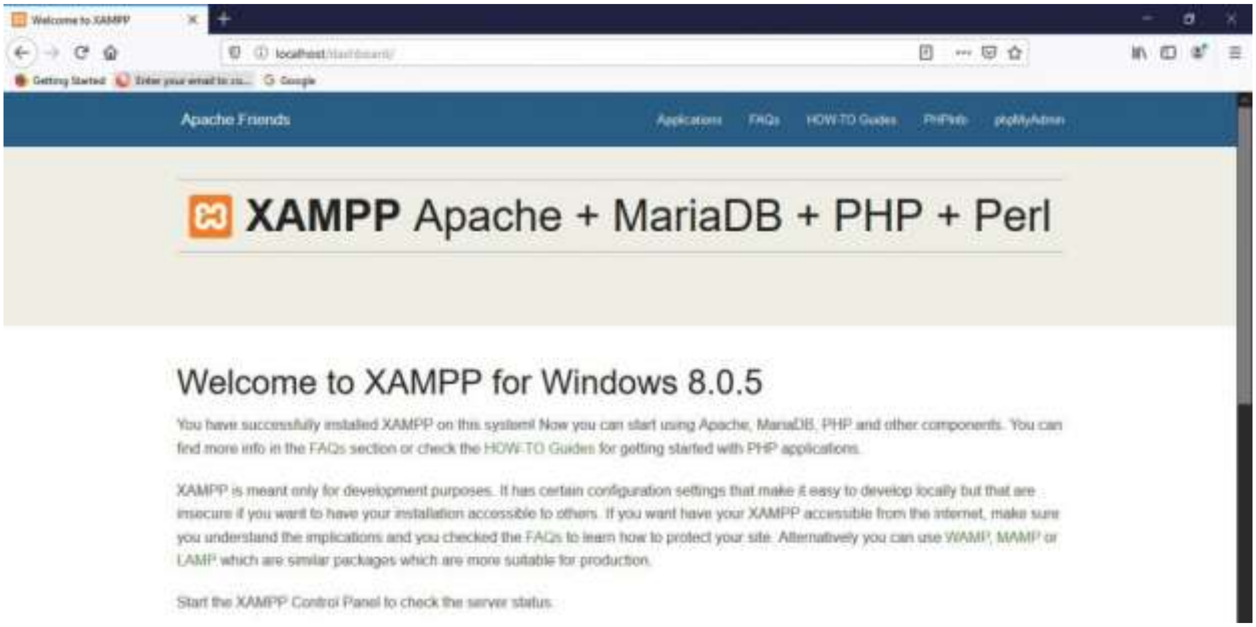
```
8:34:09 PM  [main]    XAMPP Installation Directory: "c:\xampp\"
8:34:09 PM  [main]    Checking for prerequisites
8:35:08 PM  [main]    All prerequisites found
8:35:08 PM  [main]    Initializing Modules
8:35:08 PM  [main]    Starting Check-Timer
8:35:08 PM  [main]    Control Panel Ready
8:36:42 PM  [Apache]  Status change detected: running
8:36:46 PM  [mysql]   Status change detected: running
```

Note:- Please allow the Access when you will get this allert to start the server.
The control panel contains various buttons like config, net stat, shell, explorer, services, help, quit.

- Config → It helps you to customise both the XAMPP and the individual components.
- Netstat → Allows you to see all the processes that are currently operating on your local machine.
- Shell → It enables opening UNIX shell.
- Explorer → It will launch Windows Explorer and open the XAMPP archive.
- Services → Assist in displaying all of the utilities that are actually active in the backend.
- Help → As the name suggests, it provides access to user forums.
- Quit → The XAMPP control panel will close when you press the button.

Step 6:- Go on your web browser and serch “LocalHost” to check the XAMPP software is running properly or not.
After getting this page you will make sure that you XAMPP software is running correctly.


Google
Getting Started
Enter your email to co...
Google
localhost
Gmail
Images
Sign in
Google
Google Search
I'm Feeling Lucky
Google offered in: हिन्दी বাংলা తెలుగు मराठी தமிழ் ગુજરાતી ಕನ್ನಡ മലയാളം ਪੰਜਾਬੀ



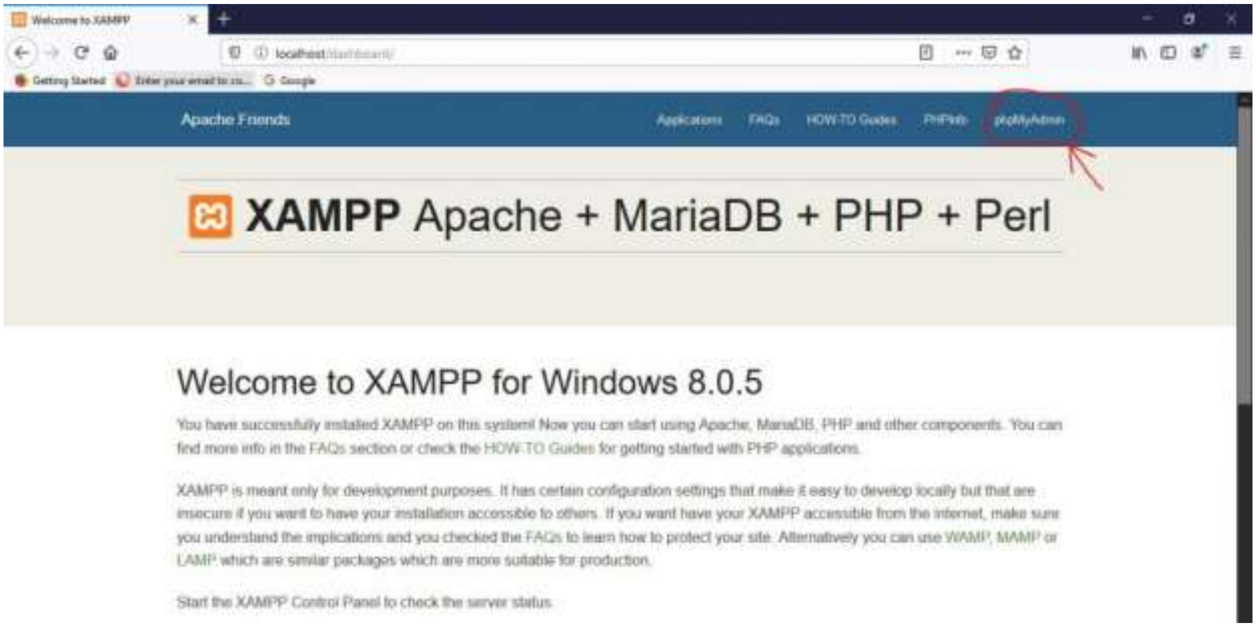
Welcome to XAMPP for Windows 8.0.5

You have successfully installed XAMPP on this system! Now you can start using Apache, MariaDB, PHP and other components. You can find more info in the FAQs section or check the HOW-TO Guides for getting started with PHP applications.

XAMPP is meant only for development purposes. It has certain configuration settings that make it easy to develop locally but that are insecure if you want to have your installation accessible to others. If you want have your XAMPP accessible from the internet, make sure you understand the implications and you checked the FAQs to learn how to protect your site. Alternatively you can use WAMP, MAMP or LAMP which are similar packages which are more suitable for production.

Start the XAMPP Control Panel to check the server status.

Step 7:- If you wanna open the phpMyAdmin then Click on this button and you will get the phpMyAdmin page.



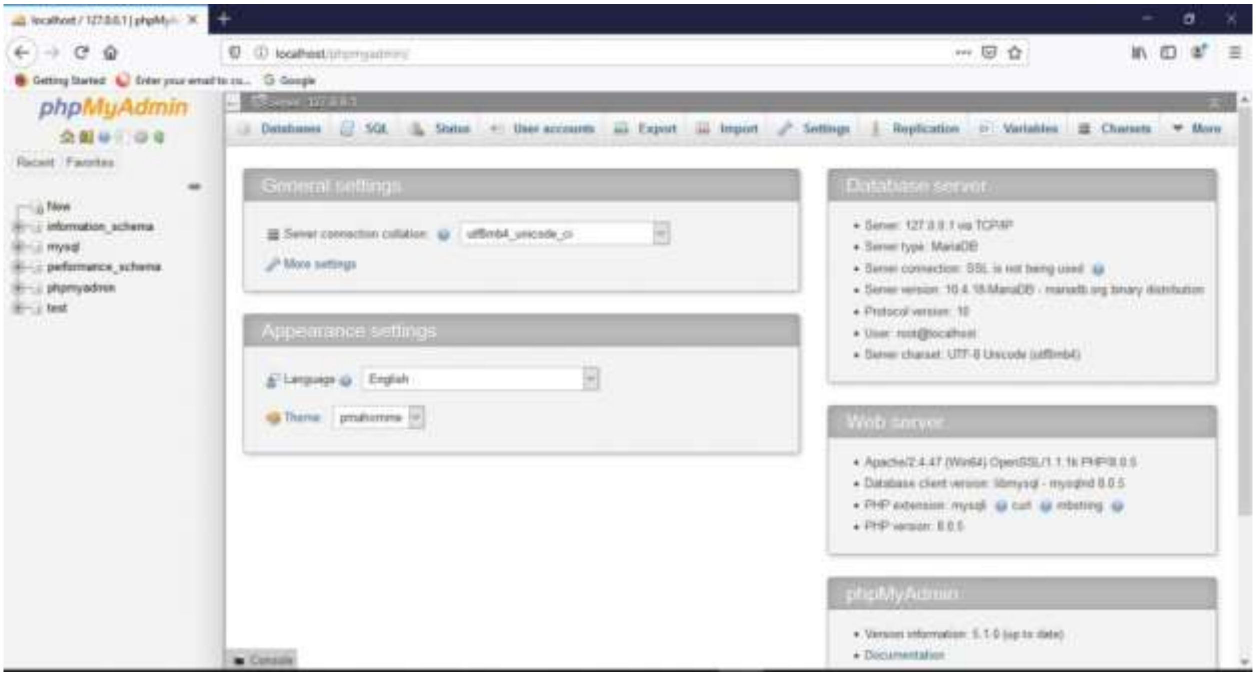
XAMPP Apache + MariaDB + PHP + Perl

Welcome to XAMPP for Windows 8.0.5

You have successfully installed XAMPP on this system! Now you can start using Apache, MariaDB, PHP and other components. You can find more info in the FAQs section or check the HOW-TO Guides for getting started with PHP applications.

XAMPP is meant only for development purposes. It has certain configuration settings that make it easy to develop locally but that are insecure if you want to have your installation accessible to others. If you want have your XAMPP accessible from the internet, make sure you understand the implications and you checked the FAQs to learn how to protect your site. Alternatively you can use WAMP, MAMP or LAMP which are similar packages which are more suitable for production.

Start the XAMPP Control Panel to check the server status.



General settings

Server connection collation: utf8mb4_unicode_ci

More settings

Appearance settings

Language: English

Theme: pmahomme

Database server

- Server: 127.0.0.1 via TCP/IP
- Server type: MariaDB
- Server connection: SSL is not being used
- Server version: 10.4.18-MariaDB - mariadb.org binary distribution
- Protocol version: 10
- User: root@localhost
- Server charset: UTF-8 Unicode (utf8mb4)

Web server

- Apache/2.4.47 (Win64) OpenSSL/1.1.1k PHP/8.0.5
- Database client version: libmysql - mysqlnd 8.0.5
- PHP extension: mysqli curl mbstring
- PHP version: 8.0.5

phpMyAdmin

- Version information: 5.1.0 (up to date)
- Documentation

References:

- https://www.devopsschool.com/blog/what-is-xampp-and-how-to-install-xampp/

Activity 2

Aim: Connect to MySQL database system and create new database

Learning outcome: Able to configure embedded databases with different web pages using XAMPP and MySQL.

Duration: 1 hour

List of Hardware/Software requirements:

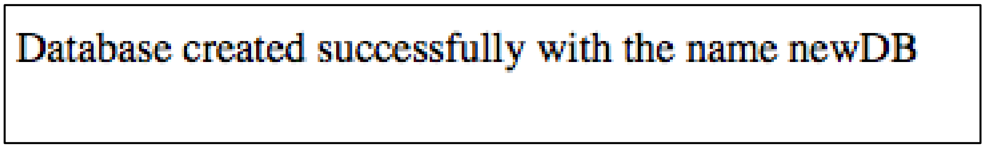
1. Laptop/Computer with Windows 10/11
2. XAMPP software

Program / Procedure:

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";


// Creating connection
$conn = mysqli_connect($servername, $username, $password);

// Checking connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}

// Creating a database named newDB
$sql = "CREATE DATABASE newDB";
if (mysqli_query($conn, $sql)) {
    echo "Database created successfully with the name newDB";
} else {
    echo "Error creating database: " . mysqli_error($conn);
}

// closing connection
mysqli_close($conn);
```

```
?>
```

Output



```
Database created successfully with the name newDB
```

References:

- https://www.geeksforgeeks.org/php-mysql-creating-database/

Activity 3

Aim: Create tables in new database and enter sample data

Learning outcome: Able to configure embedded databases with different web pages using XAMPP and MySQL.

Duration: 1 hour

List of Hardware/Software requirements:

1. Laptop/Computer with Windows 10/11
2. XAMPP software

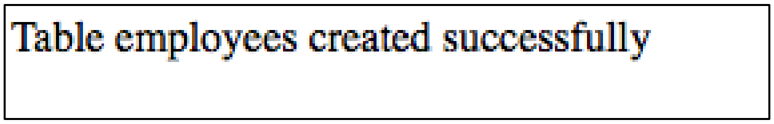
Program / Procedure :

Creating Table in Database

```
<?php

$servername = "localhost";

$username = "username";

$password = "password";

$dbname = "newDB";


// checking connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
   die("Connection failed: " . $conn->connect_error);
}

// sql code to create table
$sql = "CREATE TABLE employees(
        id INT(2)  PRIMARY KEY,
        firstname VARCHAR(30) NOT NULL,
        lastname VARCHAR(30) NOT NULL,
        email VARCHAR(50)
        )";
```

```
if ($conn->query($sql) === TRUE) {
    echo "Table employees created successfully";
} else {
    echo "Error creating table: " . $conn->error;
}

$conn->close();
?>
```

Output:



```
Table employees created successfully
```

Insert data into table

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "newDB";

// Create connection
$conn = mysqli_connect($servername, $username, $password, $dbname);
// Check connection
if (!$conn) {
  die("Connection failed: " . mysqli_connect_error());
}

$sql = "INSERT INTO employees (id, firstname, lastname, email)
VALUES (1, 'John', 'Doe', 'john@example.com')";

if (mysqli_query($conn, $sql)) {
  echo "New record created successfully";
} else {
  echo "Error: " . $sql . "<br>" . mysqli_error($conn);
}

mysqli_close($conn);
?>
```

Reference:

- https://www.w3schools.com/php/php_mysql_insert.asp

Activity 4

Aim: Import data into database table from CSV file

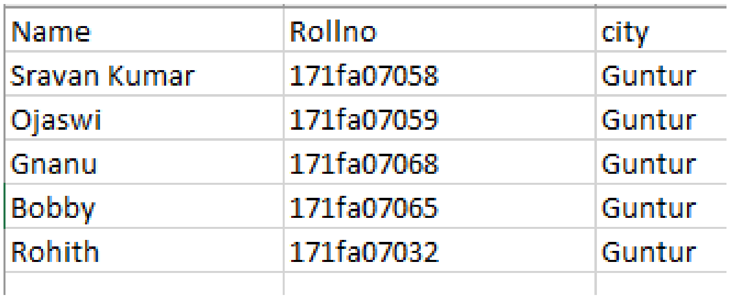
Learning outcome: Able to configure embedded databases with different web pages using XAMPP and MySQL.

Duration: 1 hour

List of Hardware/Software requirements:

1. Laptop/Computer with Windows 10/11
2. XAMPP software

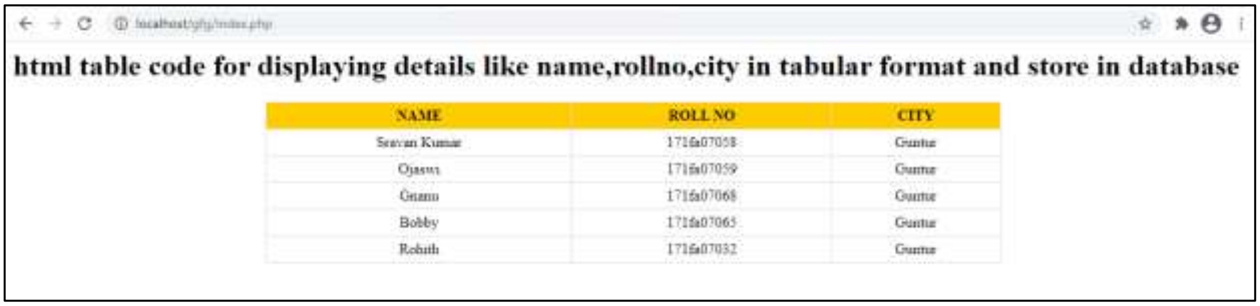
Program / Procedure:
detail.csv



Name	Rollno	city
Sravan Kumar	171fa07058	Guntur
Ojaswi	171fa07059	Guntur
Gnanu	171fa07068	Guntur
Bobby	171fa07065	Guntur
Rohith	171fa07032	Guntur

```
<?php

// Create connection
// Localhost is the server name,
// root is the username,
// password is empty
// database name is gfg
$db = new mysqli('localhost', 'root', '', 'gfg');

// Checking connection
if ($db->connect_errno) {
echo "Failed " . $db->connect_error;
exit();
}
?>

<h1>
        html table code for displaying
        details like name, rollno, city
        in tabular format and store in
        database
</h1>

<table align="center" width="800"
        border="1" style=
        "border-collapse: collapse;
        border:1px solid #ddd;"
        cellpadding="5"
        cellspacing="0">

        <thead>
                <tr bgcolor="#FFCC00">
                        <th>
                                <center>NAME</center>
                        </th>
                        <th>
                                <center>ROLL NO</center>
                        </th>
                        <th>
                                <center>CITY</center>
                        </th>

                </tr>
        </thead>
        <tbody>
                <?php

                // Get csv file
                if(($handle = fopen("detail.csv",
                                                "r")) !== FALSE) {
                        $n = 1;
                        while(($row = fgetcsv($handle))
                                                                !== FALSE) {

                                // SQL query to store data in
                                // database our table name is
                                // table2
                                $db->query('INSERT INTO table2
                                VALUES ("'.$row[0].'","'.$row[1].'",
                                "'".$row[2]."'")');

                                // row[0] = name
                                // row[1] = rollno
                                // row[2] = city
                                if($n>1) {
                                ?>
                                <tr>
                                    <td>
                                                <center>
                                                    <?php echo $row[0];?>
                                                </center>
                                    </td>
                                    <td>
                                                <center>
                                                    <?php echo $row[1];?>
                                                </center>
                                    </td>
                                    <td>
                                                <center>
                                                    <?php echo $row[2];?>
                                                </center>
                                    </td>
                                </tr>
                                    <?php
                                }

                                // Increment records
                                $n++;
                        }

                // Closing the file
                fclose($handle);
        }
        ?>
        </tbody>
</table>
```

Output:



html table code for displaying details like name,rollno,city in tabular format and store in database

NAME	ROLL NO	CITY
Sravan Kumar	171fa07058	Guntur
Ojaswi	171fa07059	Guntur
Gnanu	171fa07068	Guntur
Bobby	171fa07065	Guntur
Rohith	171fa07032	Guntur

Reference:

- https://www.geeksforgeeks.org/load-csv-data-into-mysql-server-using-php/

Activity 5

Aim: Read data from database table, all data, filtered data based on SQL queries

Learning outcome: Able to configure embedded databases with different web pages using XAMPP and MySQL.

Duration: 2 hour

List of Hardware/Software requirements:

1. Laptop/Computer with Windows 10/11
2. XAMPP software

Program /Procedure:

```
<?php
/* Attempt MySQL server connection. Assuming you are running MySQL server with default setting
(user 'root' with no password) */
$link = mysqli_connect("localhost", "root", "", "demo");
// Check connection
if($link === false)
{ die("ERROR: Could not connect. " . mysqli_connect_error()); }
// Attempt select query execution
$sql = "SELECT * FROM persons WHERE first_name='john'";
if($result = mysqli_query($link, $sql)){
    if(mysqli_num_rows($result) > 0){
        echo "<table>";
        echo "<tr>";
        echo "<th>id</th>";
        echo "<th>first_name</th>";
        echo "<th>last_name</th>";
        echo "<th>email</th>";
        echo "</tr>";
    while($row = mysqli_fetch_array($result)){
    echo "<tr>";
    echo "<td>" . $row['id'] . "</td>";
    echo "<td>" . $row['first_name'] . "</td>";
    echo "<td>" . $row['last_name'] . "</td>";
    echo "<td>" . $row['email'] . "</td>";
    echo "</tr>"; } echo "</table>";
    // Close result set
    mysqli_free_result($result); }
    else{ echo "No records matching your query were found."; } }
    else{ echo "ERROR: Could not able to execute $sql. " . mysqli_error($link); }
    // Close connection
    mysqli_close($link);
    ?>
```

Output:

```
+----+------------+-----------+---------------------+
| id | first_name | last_name | email               |
+----+------------+-----------+---------------------+
|  2 | John       | Rambo     | johnrambo@mail.com  |
|  4 | John       | Carter    | johncarter@mail.com |
+----+------------+-----------+---------------------+
```

Reference:

- https://www.tutorialrepublic.com/php-tutorial/php-mysql-where-clause.php

Activity 6

Aim: Update single record, and multiple records based on criteria

Learning outcome: Able to configure embedded databases with different web pages using XAMPP and MySQL.

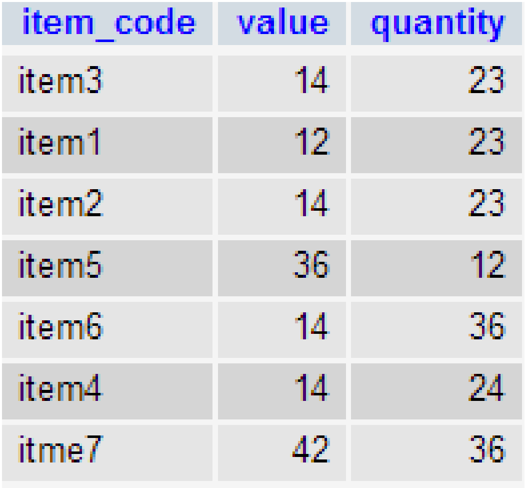
Duration: 2 hour

List of Hardware/Software requirements:

1. Laptop/Computer with Windows 10/11
2. XAMPP software

Program /Procedure:

Table Name : item Structure : item_code varchar(20), value int(11), quantity int(11) where item_code is the primary key. In the following rows of item table, 'value' column which is marked with red rectangle will be updated.



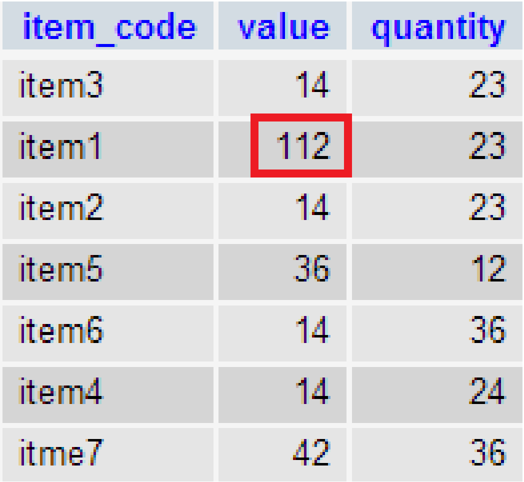
item_code	value	quantity
item3	14	23
item1	12	23
item2	14	23
item5	36	12
item6	14	36
item4	14	24
itme7	42	36

PHP Script

```
<?php

 $dbhost = 'localhost';

 $dbuser = 'root';

 $dbpass = '';
```

```
$connec = MySQL_connect($dbhost, $dbuser, $dbpass);

if(!$connec)

{

die('Could not connect: ' . MySQL_error());

}

$sql = "UPDATE item

SET value = '112'

WHERE item_code='item1'";

MySQL_select_db('MySQL');

$result = MySQL_query($sql, $connec);

if(!$result)

{

die('Could not update data: ' . MySQL_error());

}

echo "Data successfully updated...";

MySQL_close($connec);

?>
```

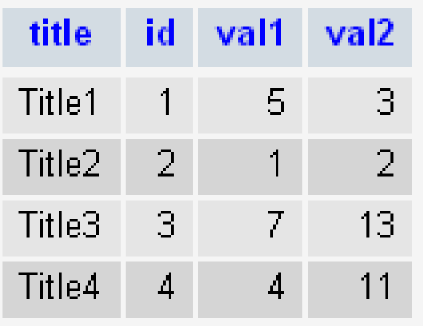
Sample Output:



item_code	value	quantity
item3	14	23
item1	112	23
item2	14	23
item5	36	12
item6	14	36
item4	14	24
itme7	42	36

Multiple Updates in MySQL

Sample table: table1



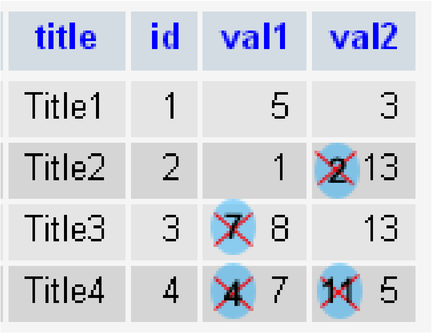
title	id	val1	val2
Title1	1	5	3
Title2	2	1	2
Title3	3	7	13
Title4	4	4	11

Code:

```
UPDATE table1 SET val1= CASE id
            WHEN 1 THEN 5
            WHEN 3 THEN 8
            WHEN 4 THEN 7
            ELSE val1
           END,
        val2= CASE id
            WHEN 2 THEN 13
```

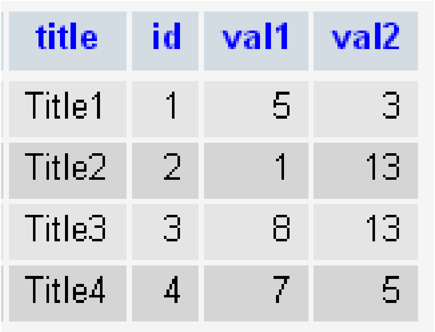
```
                WHEN 4 THEN 5

                ELSE val2

              END

        WHERE id IN (1, 2, 3, 4);
```

Pictorial presentation:



title	id	val1	val2
Title1	1	5	3
Title2	2	1	~~2~~ 13
Title3	3	~~7~~ 8	13
Title4	4	~~4~~ 7	~~11~~ 5

Sample Output:



title	id	val1	val2
Title1	1	5	3
Title2	2	1	13
Title3	3	8	13
Title4	4	7	5

Reference:

- https://www.w3resource.com/mysql/update-table/update-table.php

Activity 7

Aim: Delete selected records from table

Learning outcome: Able to configure embedded databases with different web pages using XAMPP and MySQL.

Duration: 1 hour

List of Hardware/Software requirements:

1. Laptop/Computer with Windows 10/11
2. XAMPP software

Program /Procedure:

Table: persons

```
+----+------------+-----------+----------------------+

| id | first_name | last_name | email                |

+----+------------+-----------+----------------------+

|  1 | Peter      | Parker    | peterparker@mail.com |

|  2 | John       | Rambo     | johnrambo@mail.com   |

|  3 | Clark      | Kent      | clarkkent@mail.com   |

|  4 | John       | Carter    | johncarter@mail.com  |

|  5 | Harry      | Potter    | harrypotter@mail.com |

+----+------------+-----------+----------------------+
```

```
<?php
/* Attempt MySQL server connection. Assuming you are running MySQL server with default setting
(user 'root' with no password) */
$link = mysqli_connect("localhost", "root", "", "demo");
// Check connection
if($link === false){
die("ERROR: Could not connect. " . mysqli_connect_error());
}
```

```
// Attempt delete query execution
$sql = "DELETE FROM persons WHERE first_name='John'";
if(mysqli_query($link, $sql)){
echo "Records were deleted successfully.";
} else{
echo "ERROR: Could not able to execute $sql. " . mysqli_error($link);
}
// Close connection
mysqli_close($link);
?>
```

Output:

```
+----+------------+-----------+----------------------+
| id | first_name | last_name | email                |
+----+------------+-----------+----------------------+
|  1 | Peter      | Parker    | peterparker@mail.com |
|  3 | Clark      | Kent      | clarkkent@mail.com   |
|  5 | Harry      | Potter    | harrypotter@mail.com |
+----+------------+-----------+----------------------+
```

Reference:

- https://www.tutorialrepublic.com/php-tutorial/php-mysql-delete-query.php

Activity 8

Aim: Work with SQL stored procedures

Learning outcome: Able to configure embedded databases with different web pages using XAMPP and MySQL.

Duration: 1 hour

List of Hardware/Software requirements:

1. Laptop/Computer with Windows 10/11
2. XAMPP software

Program /Procedure:

Create a new database named *test*. After you create the new database, make sure to add a user called *example* with password *example* to the database and give it read access.

```
CREATE DATABASE `test`;
```

Now create the table *users*:

```
DROP TABLE IF EXISTS `test`.`users`;

CREATE TABLE  `test`.`users` (

`users_id` int(10) unsigned NOT NULL auto_increment,

`first_name` varchar(100) NOT NULL,

`last_name` varchar(100) NOT NULL,

PRIMARY KEY  (`users_id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Before we create the stored procedures, lets put some dummy data in the *users* table. To do that just run the following query:

```
INSERT INTO `test`.`users` VALUES (null, 'Joey', 'Rivera'), (null, 'John', 'Doe');
```

Next create the first stored procedure *get_user*:

```
DELIMITER $$

DROP PROCEDURE IF EXISTS `test`.`get_user`$$

CREATE PROCEDURE  `test`.`get_user`

(

IN userId INT,

OUT firstName VARCHAR(100),

OUT lastName VARCHAR(100)

)

BEGIN

SELECT first_name, last_name
```

```
INTO firstName, lastName
FROM users
WHERE users_id = userId;
END $$
DELIMITER ;
```

Finally create the second and last stored procedure *get_users*:

```
DELIMITER $$
DROP PROCEDURE IF EXISTS `test`.`get_users`$$
CREATE PROCEDURE  `test`.`get_users`()
BEGIN
SELECT *
FROM users;
END $$
DELIMITER ;
```

The three tests will be:

1. A simple select statement
2. Calling stored procedure passing *IN* variable and retrieve *OUT* variables – *get_user*
3. Calling stored procedure with no parameters and returns a recordset – *get_users*

Below is the code to run all three tests with each of the database extensions:

```
<?php

// MYSQL

$mysql = mysql_connect('localhost', 'example', 'example');

mysql_select_db('test', $mysql);




print '<h3>MYSQL: simple select</h3>';

$rs = mysql_query( 'SELECT * FROM users;' );

while($row = mysql_fetch_assoc($rs))

{

debug($row);

}




print '<h3>MYSQL: calling sp with out variables</h3>';
```

```
$rs = mysql_query( 'CALL get_user(1, @first, @last)' );
$rs = mysql_query( 'SELECT @first, @last' );
while($row = mysql_fetch_assoc($rs))
{
debug($row);
}


print '<h3>MYSQL: calling sp returning a recordset - doesn\'t work</h3>';
$rs = mysql_query( 'CALL get_users()' );
while($row = mysql_fetch_assoc($rs))
{
debug($row);
}
function debug($o)
{
print '<pre>';
print_r($o);
print '</pre>';

}

?>
```

Output:

```
MYSQL: simple select

Array

(

    [users_id] => 1

    [first_name] => Joey

    [last_name] => Rivera

)

Array

(

    [users_id] => 2

    [first_name] => John

    [last_name] => Doe

)
```

```
MYSQL: calling sp with out variables

Array

(

    [@first] => Joey

    [@last] => Rivera

)



MYSQL: calling sp returning a recordset - doesn't work

Warning: mysql_fetch_assoc(): supplied argument is not a valid MySQL result resource in
***test.php on line 24
```

Reference:

- http://www.joeyrivera.com/2009/01/27/using-mysql-stored-procedures-with-php-mysqlmysqlipdo/

Activity 9

Aim: Perform transaction in database tables using commit and rollback operations

Learning outcome: Able to configure embedded databases with different web pages using XAMPP and MySQL.

Duration: 5 hours

List of Hardware/Software requirements:

1. Laptop/Computer with Windows 10/11
2. XAMPP software

Program /Procedure:

```
<?php

// trans.php
function begin(){
   mysql_query("BEGIN");
}

function commit(){
   mysql_query("COMMIT");
}

function rollback(){
   mysql_query("ROLLBACK");
}

mysql_connect("localhost","Dude1", "SuperSecret") or die(mysql_error());

mysql_select_db("bedrock") or die(mysql_error());

$query = "INSERT INTO employee (ssn,name,phone) values ('123-45-6789','Matt','1-800-555-1212')";

begin(); // transaction begins

$result = mysql_query($query);
```

```
if(!$result){
    rollback(); // transaction rolls back
    echo "transaction rolled back";
    exit;
}else{
    commit(); // transaction is committed
    echo "Database transaction was successful";
}

?>
```

Reference:

- https://stackoverflow.com/questions/2708237/php-mysql-transactions-examples

Activity 10

Aim: Demonstrate use of transaction control in relational databases

Learning outcome: Able to configure embedded databases with different web pages using XAMPP and MySQL.

Duration: 5 hour

List of Hardware/Software requirements:

1. Laptop/Computer with Windows 10/11
2. XAMPP software

Program /Procedure:

Using Savepoint and Rollback
Following is the table **class**,

id	**name**
1	Abhi
2	Adam
4	Alex

Let’s use some SQL queries on the above table and see the results.

```
INSERT INTO class VALUES(5, 'Rahul');
COMMIT;
UPDATE class SET name = 'Abhijit' WHERE id = '5';
SAVEPOINT A;
INSERT INTO class VALUES(6, 'Chris');
SAVEPOINT B;
INSERT INTO class VALUES(7, 'Bravo');
SAVEPOINT C;

SELECT * FROM class;
```

SELECT statement is used to show the data stored in the table.
The resultant table will look like,

id	name
1	Abhi
2	Adam
4	Alex
5	Abhijit
6	Chris
7	Bravo

Now let's use the ROLLBACK command to roll back the state of data to the **savepoint B**.

```
ROLLBACK TO B;
SELECT * FROM class;
```

Now our **class** table will look like,

id	name
1	Abhi
2	Adam
4	Alex
5	Abhijit
6	Chris

Now let's again use the ROLLBACK command to roll back the state of data to the **savepoint A**

```
ROLLBACK TO A;

SELECT * FROM class;
```

Now the table will look like,

id	name
1	Abhi
2	Adam
4	Alex

5	Abhijit

So now you know how the commands COMMIT, ROLLBACK and SAVEPOINT works.

Reference:

- https://www.studytonight.com/dbms/tcl-command.php

Activity 11

Aim: Perform user management activities in MySQL database, create, modify user, set privileges

Learning outcome: Able to configure embedded databases with different web pages using XAMPP and MySQL.

Duration: 5 hour

List of Hardware/Software requirements:

1. Laptop/Computer with Windows 10/11
2. XAMPP software

Program /Procedure:

MySQL | CREATE USER Statement

MySQL allows us to specify which user account can connect to a database server. The user account details in MySQL contains two information – username and host from which the user is trying to connect in the format *username@host-name*.

If the admin user is connecting through localhost then the user account will be *admin@localhost*.

MySQL stores the user account in the *user* grant table of the *mysql* database.

The **CREATE USER** statement in MySQL allows us to create new MySQL accounts or in other words, the **CREATE USER** statement is used to create a database account that allows the user to log into the MySQL database.

Syntax:

The syntax for the CREATE USER statement in MySQL is:

CREATE USER user_account IDENTIFIED BY password;

Parameters used:

1. **user_account:** It is the name that the user wants to give to the database account.The user_account should be in the format **'username'@'hostname'**
2. **password:**It is the password used to assign to the user_account.The password is specified in the IDENTIFIED BY clause.

Below are the different ways in which CREATE USER statement can be used:

- **Creating a new user**: For creating a new user "gfguser1" that connects to the MySQL database server from the localhost with the password "abcd", the CREATE USER statement should be used in the following way.

Syntax:

```
CREATE USER gfguser1@localhost IDENTIFIED BY 'abcd';
```

Note: The create user statement only creates a new user,it does not grant any permissions to the user.

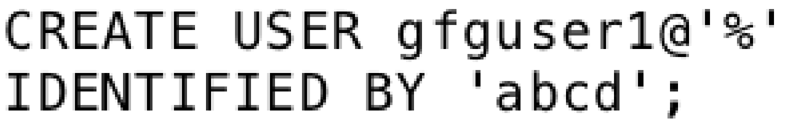
- **Creating more than one user**: For creating more than one new user that connects to the MySQL database server from the localhost, the CREATE USER statement should be used in the following way.
  Syntax:

```
CREATE USER

'gfguser2'@'localhost' IDENTIFIED BY 'efgh',

'gfguser3'@'localhost' IDENTIFIED BY 'uvxy';
```

The above code creates two new users with username “gfguser2” and “gfguser3” with passwords “efgh” and “uvxy” respectively.

- **Allowing a user account to connect from any host**: To allow a user account to connect from any host, the percentage (%) wildcard is used in the following way.

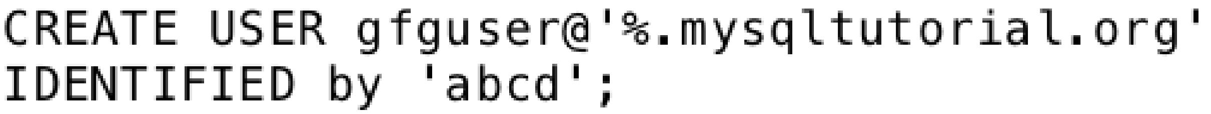
- **Syntax:**



```
CREATE USER gfguser1@'%'
IDENTIFIED BY 'abcd';
```

To allow the user account to connect to the database server from any subdomain of the “mysqltutorial.org” host, then the percentage wildcard % should be used as follows:

Syntax:

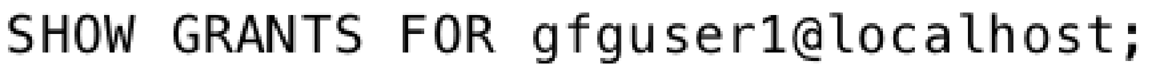


```
CREATE USER gfguser@'%.mysqltutorial.org'
IDENTIFIED by 'abcd';
```

- **Viewing permissions of an User Account**: The “Show Grants” statement is used to view the permissions of a user account.The show grants statement is used in the following way:
  Syntax:

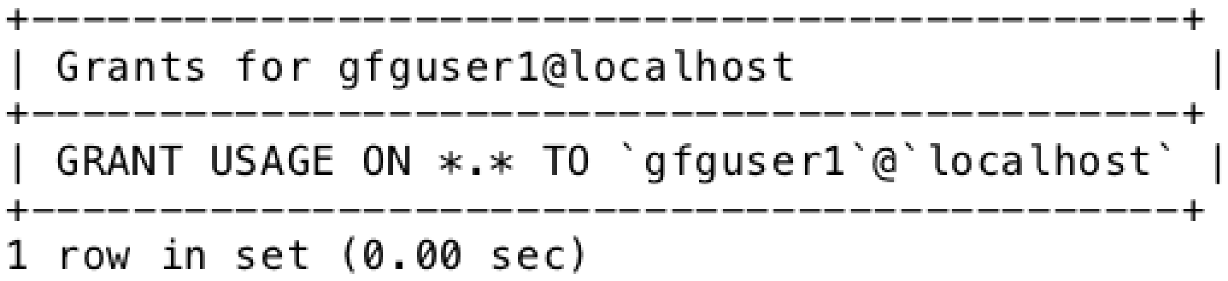
SHOW GRANTS FOR user-account;

Example:



```
SHOW GRANTS FOR gfguser1@localhost;
```

Output:



```
+--------------------------------------------+
| Grants for gfguser1@localhost               |
+--------------------------------------------+
| GRANT USAGE ON *.* TO `gfguser1`@`localhost` |
+--------------------------------------------+
1 row in set (0.00 sec)
```

Reference:

- https://www.geeksforgeeks.org/mysql-create-user-statement/

Learning Outcome

After completing this module, the student should be able to configure an embedded database with different web pages using MongoDB
To meet the learning outcome, a student has to complete the following activities

1. Install of MongoDB in the system (2Hrs)
2. Create data with the following Data types - String, Integer, Boolean, double, min/max keys, arrays, timestamp, object, Null, symbol, date, object ID, Binary data, Code, Regular Expression (3Hrs)
3. Insert Document in database (1Hr)
4. Update document in database (1Hr)
5. Delete document in database (1Hr)
6. Project document in document (2Hr)
7. Create a MongoDB query to display all the documents in the collection data (Trainee's data) (5Hrs)
8. Create a MongoDB query to display the fields id, trainee name, lab name, Certificate No., course title, course starting date, course ending date for all the documents in the collection trainee's data. (5 Hrs)
9. Create a MongoDB query to display the fields id, trainee name, lab name, Certificate No., course name, course starting date, course ending date for all the documents in the collection trainee's data, but excluding lab name (5 Hrs)
10. Create a MongoDB query to display all the trainees who attended course on PHP (2Hrs)
11. Create a MongoDB query to display the 1st batch trainees of PHP (2Hrs)
12. Create a MongoDB query to display the 2nd batch trainees of PHP (2Hrs)
13. Create a MongoDB query to find the course where maximum trainees attended (2Hrs)
14. Create a MongoDB query to find lab wise details of trainees (2Hrs)
15. Create a MongoDB query with course wise details of trainees (2Hrs)
16. Create MongoDB cluster in MongoDB Atlas cloud service (2Hrs)
17. Connect and use MongoDB Atlas on local system (2Hrs)
18. Get familiar with big data and its tools (2Hrs)

Activity 1

Aim: To install MongoDB in the computer system running with Ubuntu 18.04 LTS
Learning outcome: Able to configure embedded databases with different web pages using MongoDB.
Duration: 2 hour
List of Hardware/Software requirements:

1. Laptop/Computer with Linux OS - Ubuntu 18.04 LTS
2. MongoDB 4.2 Community Edition

Installation Steps of MongoDB Community Edition:

1) Import the public key used by the package management system.

From a terminal, issue the following command to import the MongoDB public GPG Key from https://www.mongodb.org/static/pgp/server-4.2.asc

```
wget -qO - https://www.mongodb.org/static/pgp/server-4.2.asc | sudo apt-key add –
```

The operation should respond with an OK.
However, if you receive an error indicating that gnupg is not installed, you can:

a) Install gnupg and its required libraries using the following command:

```
sudo apt-get install gnupg
```

b) Once installed, retry importing the key:

```
wget -qO - https://www.mongodb.org/static/pgp/server-4.2.asc | sudo apt-key add -
```

2) Create a /etc/apt/sources.list.d/mongodb-org-4.2.list file for MongoDB.
Create the list file using the command appropriate for your version of Debian:

```
    echo "deb http://repo.mongodb.org/apt/debian buster/mongodb-org/4.2 main" | sudo
tee/etc/apt/sources.list.d/mongodb-org-4.2.list
```

3) Reload local package database.
Issue the following command to reload the local package database:

```
sudo apt-get update
```

4) Install the MongoDB packages
To install the latest stable version, run command as following

```
sudo apt-get install -y mongodb-org
```

5) Start MongoDB
You can start the mongod process by issuing the following command:

```
sudo systemctl start mongod
```

6) **Begin using MongoDB.**

Run the following command

```
mongo
```

References:

- https://docs.mongodb.com/manual/tutorial/install-mongodb-on-debian/

Activity 2

Aim: Create MongoDB database with the following Data types – String, Integer, Boolean, double, min/max keys, arrays, timestamp, object, Null, symbol, date, object ID, Binary data, Code, Regular Expression

Learning outcome: Able to configure embedded databases with different web pages using MongoDB.

Duration: 3 hours

List of Hardware/Software requirements:

1. Laptop/Computer with Linux OS - Ubuntu 18.04 LTS
2. MongoDB 4.2 Community Edition
3. XAMPP- PHP 7
4. PHP-MongoDB driver 1.7.4

Create a Data:

a) Insert a Document without Specifying an _id Field

In the following example, the document passed to the insert() method does not contain the _id field:

```
db.products.insert( { item: "card", qty: 15 } )
```

Output:

```
{ "_id" : ObjectId("5063114bd386d8fadbd6b004"), "item" : "card", "qty" : 15 }
```

b) Insert Multiple Documents

The following example performs a bulk insert of three documents by passing an array of documents to the insert() method.

```
db.products.insert( [
{ _id: 11, item: "pencil", qty: 50, type: "no.2" },
{ item: "pen", qty: 20 },
{ item: "eraser", qty: 25 }
]
)
```

Output:

```
{ "_id" : 11, "item" : "pencil", "qty" : 50, "type" : "no.2" }
{ "_id" : ObjectId("51e0373c6f35bd826f47e9a0"), "item" : "pen", "qty" : 20 }
{ "_id" : ObjectId("51e0373c6f35bd826f47e9a1"), "item" : "eraser", "qty" : 25 }
```

We can either set the value of _id for each document or else _id value is assigned automatically.

Let’s take a look at the example below

```
db.student.insertMany( [
{name : "Alex", age : 19}, {name : "Albert" , age : 20}, {name : "Bob" , age : 19}
]
)
```

Output:

```
{
"acknowledged" : true, "insertedIds" : [
ObjectId("5b680cbc80847beb3aa3e837"), ObjectId("5b680cbc80847beb3aa3e838"),
ObjectId("5b680cbc80847beb3aa3e839")
]
}
```

Reference:

- https://docs.mongodb.com/manual/reference/method/db.collection.insert/

Activity 3

Aim: Insert Document in database

Learning outcome: Able to configure embedded databases with different web pages using MongoDB.

Duration: 1 hour

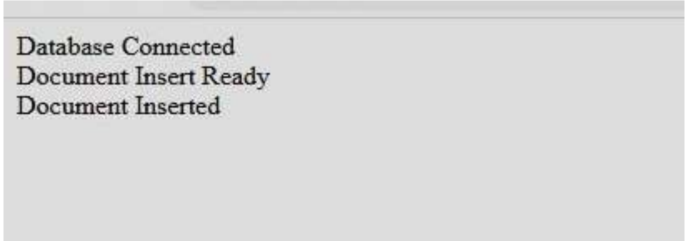
List of Hardware/Software requirements:

1. Laptop/Computer with Linux OS - Ubuntu 18.04 LTS
2. MongoDB 4.2 Community Edition
3. XAMPP- PHP 7
4. PHP-MongoDB driver 1.7.4

Program:

```
<?php
//connecting to database
if($con=new MongoDB\Driver\Manager("mongodb://localhost:27017")) echo "Database
Connected";
//creating document as array
$doc=array("_id"=>"124", "Name"=>"John", "Age"=>"23", "Location"=>"India");
//creating write database object
$single_insert=new MongoDB\Driver\BulkWrite();
if($single_insert->insert($doc))      //preparing query statement for insert echo "<br>Document
Insert Ready";
if($con->executeBulkWrite("mydb.mycol", $single_insert)) //executing write query echo
"<br>Document Inserted";
?>
```

Output:



```
Database Connected
Document Insert Ready
Document Inserted
```

Activity 4

Aim: Update Document in database

Learning outcome: Able to configure embedded databases with different web pages using MongoDB.

Duration: 1 hour

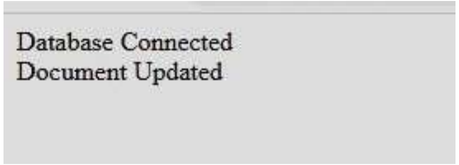
List of Hardware/Software requirements:

1. Laptop/Computer with Linux OS - Ubuntu 18.04 LTS
2. MongoDB 4.2 Community Edition
3. XAMPP- PHP 7
4. PHP-MongoDB driver 1.7.4

Program:

```
<?php
//connecting database
if($con=new MongoDB\Driver\Manager("mongodb://localhost:27017")) echo "Database Connected";
//creating write object
$single_update=new MongoDB\Driver\BulkWrite();
//creating query for update
$single_update->update(["_id"=>"124"], ["Name"=>"Raj", "Age"=>"26"], ["multi"=>false, "upsert"=>false]);
//executing query for update
if($con->executeBulkWrite("mydb.mycol", $single_update)) echo "<br>Document Updated";
?>
```

Output:



```
Database Connected
Document Updated
```

Activity 5

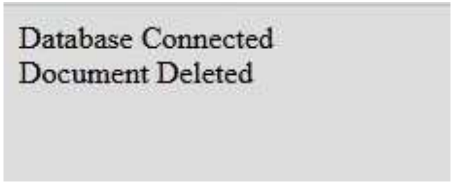
Aim: Delete Document in database
Learning outcome: Able to configure embedded databases with different web pages using MongoDB.
Duration: 1 hour
List of Hardware/Software requirements:

1. Laptop/Computer with Linux OS - Ubuntu 18.04 LTS
2. MongoDB 4.2 Community Edition
3. XAMPP- PHP 7
4. PHP-MongoDB driver 1.7.4

Program:

```
<?php
//connecting database
if($con=new MongoDB\Driver\Manager("mongodb://localhost:27017")) echo "Database Connected";
//creating object for write
$del=new MongoDB\Driver\BulkWrite();
//creating delete query
$del->delete(["Name"=>"John"], ["limit"=>0]);
//executing delete query
if($con->executeBulkWrite("mydb.mycol", $del)) echo "<br>Document Deleted";
?>
```

Output:



```
Database Connected
Document Deleted
```

Activity 6

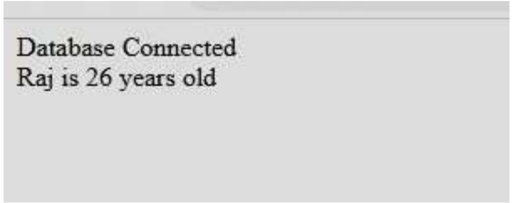
Aim: Project Document in database
Learning outcome: Able to configure embedded databases with different web pages using MongoDB.
Duration: 2 hours
List of Hardware/Software requirements:

1. Laptop/Computer with Linux OS - Ubuntu 18.04 LTS
2. MongoDB 4.2 Community Edition
3. XAMPP- PHP 7
4. PHP-MongoDB driver 1.7.4

Program:

```
<?php
//connecting to database
if($con=new MongoDB\Driver\Manager("mongodb://localhost:27017")) echo "Database
Connected";
$filter=["Name"=>"Raj"];     //defining projection filter
$option=[];     //leaving options blank
$read=new MongoDB\Driver\Query($filter, $option);     //creating query
$single_user=$con->executeQuery("mydb.mycol", $read);     //executing query
foreach($single_user as $user){   //foreach loop for traversing result displaying it
echo "<br>".$user->Name." is ".$user->Age." years old";
}
?>
```

Output:



```
Database Connected
Raj is 26 years old
```

Activity 7

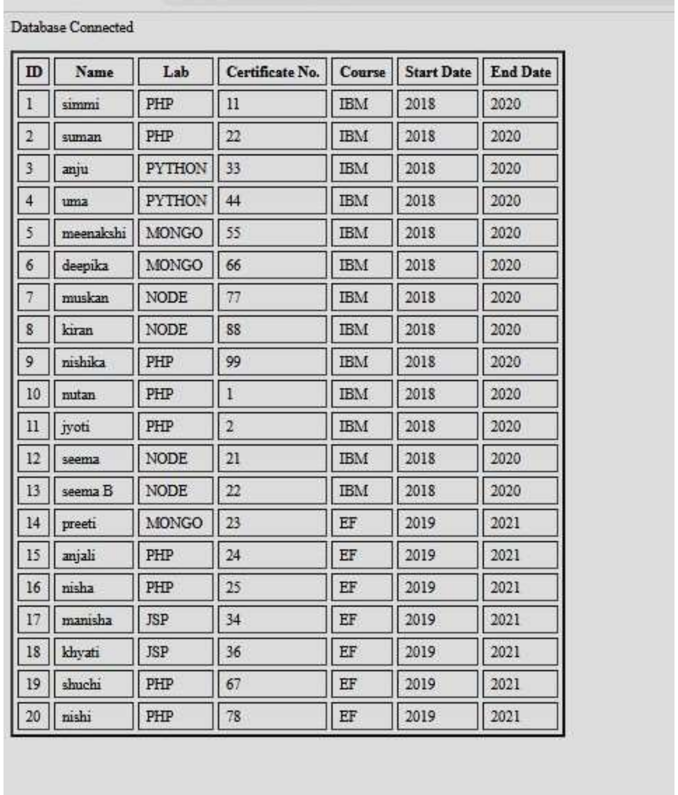
Aim: Create a MongoDB query to display all the documents in the collection data (Trainees **data)**
Learning outcome: Able to configure embedded databases with different web pages using MongoDB.
Duration: 5 hours
List of Hardware/Software requirements:

1. Laptop/Computer with Linux OS - Ubuntu 18.04 LTS
2. MongoDB 4.2 Community Edition
3. XAMPP- PHP 7
4. PHP-MongoDB driver 1.7.4

Program:

```
<?php
//connecting to database
if($con=new MongoDB\Driver\Manager("mongodb://localhost:27017")) echo "Database
Connected<br><br>";
$filter=[];       //no filter
$option=[];    //no options
$read=new MongoDB\Driver\Query($filter, $option);    //creating query
$single_user=$con->executeQuery("mydb3.mycol", $read); //executing query
//creating HTML table for displaying data echo "<table border=3 cellspacing=5
cellpadding=7><thead><th>ID<th>Name<th>Lab<th>Certificate No.<th>Course<th>Start
Date<th>End Date</thead>";
foreach($single_user as $user){   //parsing through results in loop echo "<tr>";
echo
"<td>".$user->_id."<td>".$user->name."<td>".$user->lab."<td>".$user->certNo."<td>".$user->c
ourse."<td>".$user->start."<td>".$user->end;
}        //end of loop for parsing echo "</table>"; //end of table
?>
```

Output:



Database Connected

ID	Name	Lab	Certificate No.	Course	Start Date	End Date
1	simmi	PHP	11	IBM	2018	2020
2	suman	PHP	22	IBM	2018	2020
3	anju	PYTHON	33	IBM	2018	2020
4	uma	PYTHON	44	IBM	2018	2020
5	meenakshi	MONGO	55	IBM	2018	2020
6	deepika	MONGO	66	IBM	2018	2020
7	muskan	NODE	77	IBM	2018	2020
8	kiran	NODE	88	IBM	2018	2020
9	nishika	PHP	99	IBM	2018	2020
10	nutan	PHP	1	IBM	2018	2020
11	jyoti	PHP	2	IBM	2018	2020
12	seema	NODE	21	IBM	2018	2020
13	seema B	NODE	22	IBM	2018	2020
14	preeti	MONGO	23	EF	2019	2021
15	anjali	PHP	24	EF	2019	2021
16	nisha	PHP	25	EF	2019	2021
17	manisha	JSP	34	EF	2019	2021
18	khyati	JSP	36	EF	2019	2021
19	shuchi	PHP	67	EF	2019	2021
20	nishi	PHP	78	EF	2019	2021

Activity 8

Aim: Create a MongoDB query to display the fields id, trainee name, lab name, Certificate No., course title, course starting date,course ending date for all the documents in the collection trainees data.

Learning outcome: Able to configure embedded databases with different web pages using MongoDB.

Duration: 5 hours

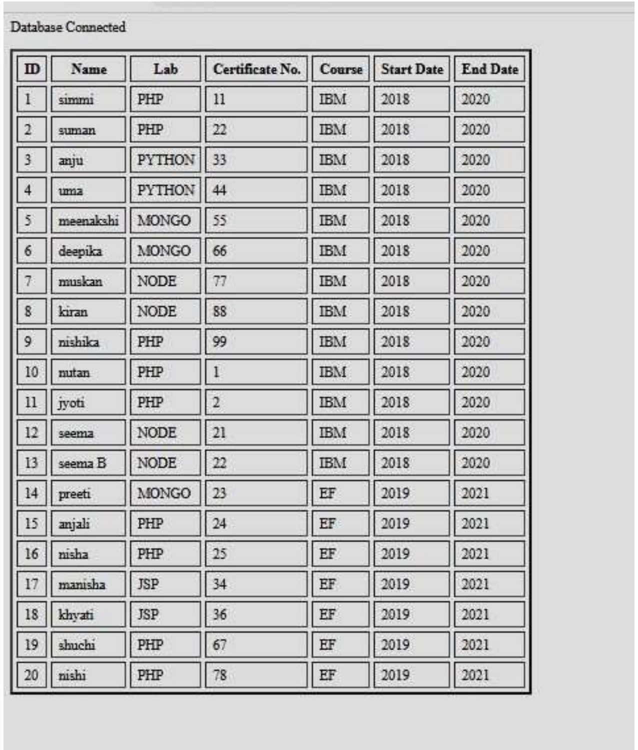
List of Hardware/Software requirements:

1. Laptop/Computer with Linux OS - Ubuntu 18.04 LTS
2. MongoDB 4.2 Community Edition
3. XAMPP- PHP 7
4. PHP-MongoDB driver 1.7.4

Program:

```
<?php
//connecting to database
if($con=new MongoDB\Driver\Manager("mongodb://localhost:27017")) echo "Database
Connected<br><br>";
//no filters & no options applied
$filter=[];
$option=[];
//creating query object
$read=new MongoDB\Driver\Query($filter, $option);
//executing query and receive results
$single_user=$con->executeQuery("mydb3.mycol", $read);
//create table view
echo "<table border=3 cellspacing=5 cellpadding=7><thead><th>ID<th>Name<th>Certificate
No.<th>Course<th>Start Date<th>End Date</thead>";
foreach($single_user as $user){    //parse through results using loop echo "<tr>";
echo
"<td>".$user->_id."<td>".$user->name."<td>".$user->certNo."<td>".$user->course."<td>".$user
->start."<td>".$user->end;
}        //end of loop
echo "</table>";      //end of table
?>
```

Output:



Database Connected

ID	Name	Lab	Certificate No.	Course	Start Date	End Date
1	simmi	PHP	11	IBM	2018	2020
2	suman	PHP	22	IBM	2018	2020
3	anju	PYTHON	33	IBM	2018	2020
4	uma	PYTHON	44	IBM	2018	2020
5	meenakshi	MONGO	55	IBM	2018	2020
6	deepika	MONGO	66	IBM	2018	2020
7	muskan	NODE	77	IBM	2018	2020
8	kiran	NODE	88	IBM	2018	2020
9	nishika	PHP	99	IBM	2018	2020
10	nutan	PHP	1	IBM	2018	2020
11	jyoti	PHP	2	IBM	2018	2020
12	seema	NODE	21	IBM	2018	2020
13	seema B	NODE	22	IBM	2018	2020
14	preeti	MONGO	23	EF	2019	2021
15	anjali	PHP	24	EF	2019	2021
16	nisha	PHP	25	EF	2019	2021
17	manisha	JSP	34	EF	2019	2021
18	khyati	JSP	36	EF	2019	2021
19	shuchi	PHP	67	EF	2019	2021
20	nishi	PHP	78	EF	2019	2021

Activity 9

Aim: Create a MongoDB query to display the fields id, trainee name, lab name, Certificate No., course name, course starting date, the course ending date for all the documents in the collection trainee’s data, but excluding lab name

Learning outcome: Able to configure embedded databases with different web pages using MongoDB.

Duration: 5 hours

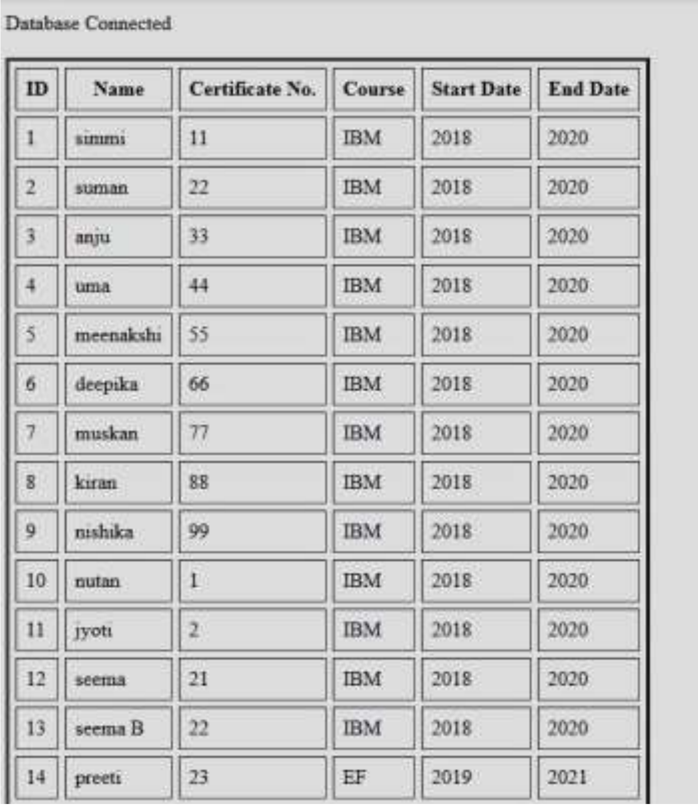
List of Hardware/Software requirements:

1. Laptop/Computer with Linux OS - Ubuntu 18.04 LTS
2. MongoDB 4.2 Community Edition
3. XAMPP- PHP 7
4. PHP-MongoDB driver 1.7.4

Program:

```
<?php
//connecting to database
if($con=new MongoDB\Driver\Manager("mongodb://localhost:27017")) echo "Database
Connected<br><br>";
$filter=[];        //no filters
$option=[ 'projection'=>['lab'=>0] ];          //options to eliminate lab column from result
$read=new MongoDB\Driver\Query($filter, $option);     //creating query
$single_user=$con->executeQuery("mydb3.mycol", $read); //executing query
//creating table view without column for lab
echo "<table border=3 cellspacing=5 cellpadding=7><thead><th>ID<th>Name<th>Certificate
No.<th>Course<th>Start Date<th>End Date</thead>";
foreach($single_user as $user){    //parsing results in loop echo "<tr>";
echo
"<td>".$user->_id."<td>".$user->name."<td>".$user->certNo."<td>".$user->course."<td>".$user
->start."<td>".$user->end;
}          end loop
echo "</table>";       //end table
?>
```

Output:



Database Connected

ID	Name	Certificate No.	Course	Start Date	End Date
1	simmi	11	IBM	2018	2020
2	suman	22	IBM	2018	2020
3	anju	33	IBM	2018	2020
4	uma	44	IBM	2018	2020
5	meenakshi	55	IBM	2018	2020
6	deepika	66	IBM	2018	2020
7	muskan	77	IBM	2018	2020
8	kiran	88	IBM	2018	2020
9	nishika	99	IBM	2018	2020
10	nutan	1	IBM	2018	2020
11	jyoti	2	IBM	2018	2020
12	seema	21	IBM	2018	2020
13	seema B	22	IBM	2018	2020
14	preeti	23	EF	2019	2021

Activity 10

Aim: Create a MongoDB query to display all the trainees who attended courses on PHP.

Learning outcome: Able to configure embedded databases with different web pages using MongoDB.

Duration: 2 hours

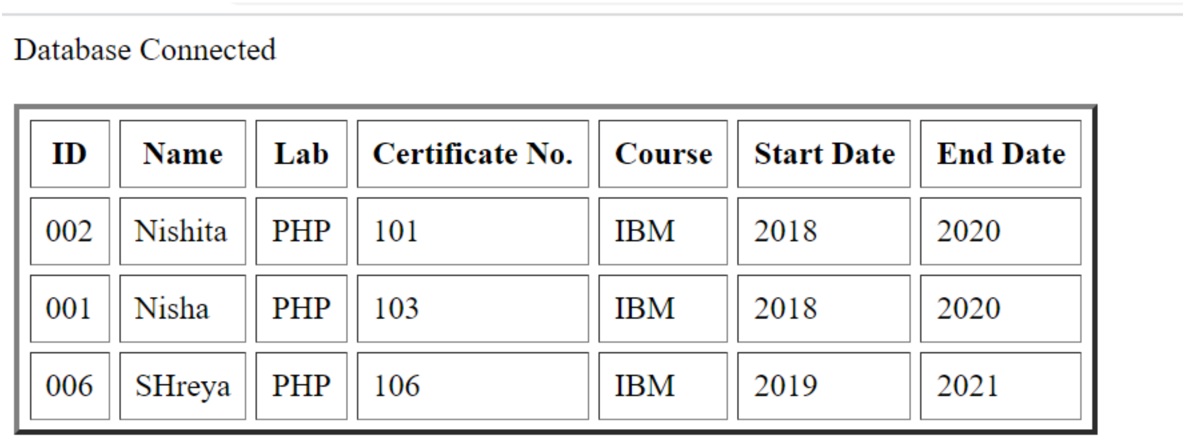
List of Hardware/Software requirements:

1. Laptop/Computer with Linux OS - Ubuntu 18.04 LTS
2. MongoDB 4.2 Community Edition
3. XAMPP- PHP 7
4. PHP-MongoDB driver 1.7.4

Program:

```
<?php

//connecting to database

if($con=new MongoDB\Driver\Manager("mongodb://localhost:27017")) echo "Database
Connected<br><br>";

$filter=[‘lab’=> ‘PHP’];        //no filters

$option=[];    //options to eliminate lab column from result

$read=new MongoDB\Driver\Query($filter, $option);    //creating query

$single_user=$con->executeQuery("mydb3.mycol", $read); //executing query

//creating table view without column for lab

echo "<table border=3 cellspacing=5 cellpadding=7><thead><th>ID<th>Name<th>Certificate
No.<th>Course<th>Start Date<th>End Date</thead>";

foreach($single_user as $user){    //parsing results in loop echo "<tr>";

echo

"<td>".$user->_id."<td>".$user->name."<td>".$user->certNo."<td>".$user->course."<td>".$user

->start."<td>".$user->end;
}        end loop

echo "</table>";      //end table

?>
```

Output:



Database Connected

ID	Name	Lab	Certificate No.	Course	Start Date	End Date
002	Nishita	PHP	101	IBM	2018	2020
001	Nisha	PHP	103	IBM	2018	2020
006	SHreya	PHP	106	IBM	2019	2021

Activity 11

Aim: Create a MongoDB query to display the 1st batch trainees of PHP

Learning outcome: Able to configure embedded databases with different web pages using MongoDB.

Duration: 2 hours

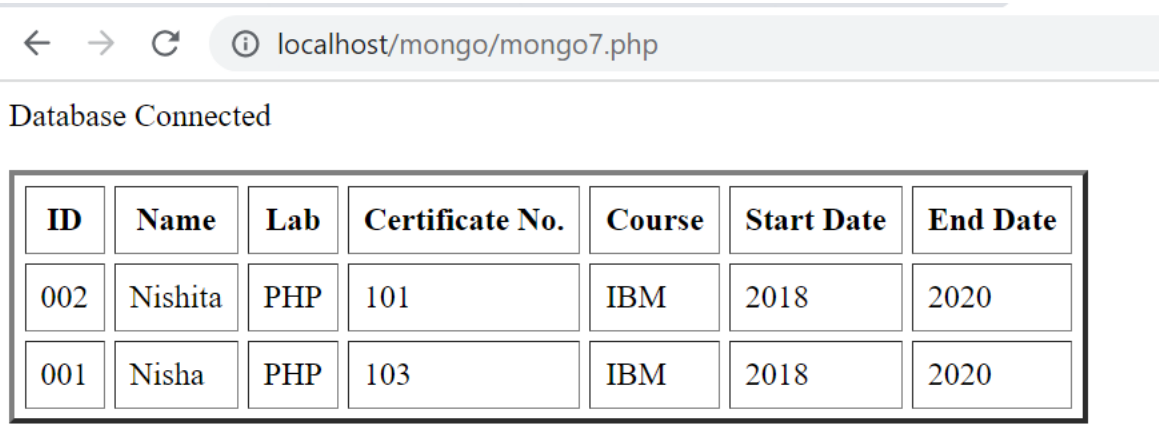
List of Hardware/Software requirements:

1. Laptop/Computer with Linux OS - Ubuntu 18.04 LTS
2. MongoDB 4.2 Community Edition
3. XAMPP- PHP 7
4. PHP-MongoDB driver 1.7.4

Program:

```
<?php

//connecting to database

if($con=new MongoDB\Driver\Manager("mongodb://localhost:27017")) echo "Database
Connected<br><br>";

$filter=[‘lab’=> ‘PHP’, ‘start’=> ‘2018’];     //no filters

$option=[];     //options to eliminate lab column from result

$read=new MongoDB\Driver\Query($filter, $option);     //creating query

$single_user=$con->executeQuery("mydb3.mycol", $read); //executing query

//creating table view without column for lab

echo "<table border=3 cellspacing=5 cellpadding=7><thead><th>ID<th>Name<th>Certificate
No.<th>Course<th>Start Date<th>End Date</thead>";

foreach($single_user as $user){    //parsing results in loop echo "<tr>";

echo

"<td>".$user->_id."<td>".$user->name."<td>".$user->certNo."<td>".$user->course."<td>".$user

->start."<td>".$user->end;

}         end loop

echo "</table>";       //end table

?>
```

Output:



localhost/mongo/mongo7.php

Database Connected

ID	Name	Lab	Certificate No.	Course	Start Date	End Date
002	Nishita	PHP	101	IBM	2018	2020
001	Nisha	PHP	103	IBM	2018	2020

Activity 12

Aim: Create a MongoDB query to display the 2nd batch trainees of PHP

Learning outcome: Able to configure embedded databases with different web pages using MongoDB.

Duration: 2 hours

List of Hardware/Software requirements:

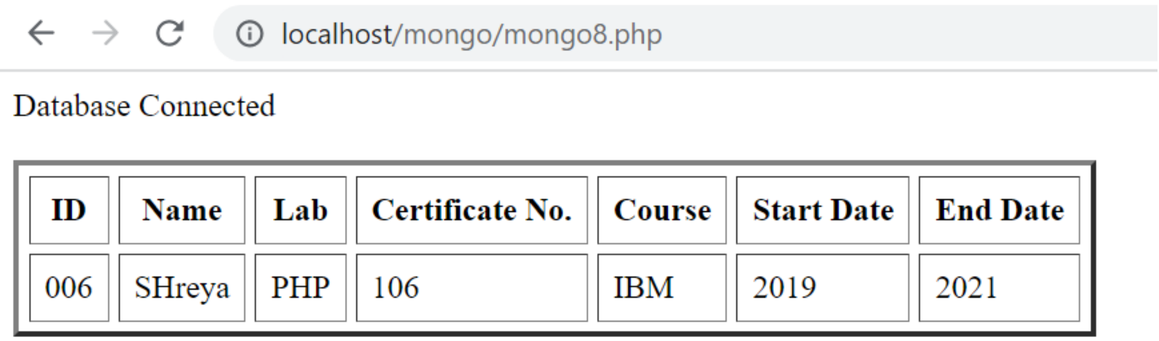
1. Laptop/Computer with Linux OS - Ubuntu 18.04 LTS
2. MongoDB 4.2 Community Edition
3. XAMPP- PHP 7
4. PHP-MongoDB driver 1.7.4

Program:

```
<?php

//connecting to database

if($con=new MongoDB\Driver\Manager("mongodb://localhost:27017")) echo "Database Connected<br><br>";

$filter=[‘lab’=> ‘PHP’, ‘start’=> ‘2019’];     //no filters

$option=[];     //options to eliminate lab column from result

$read=new MongoDB\Driver\Query($filter, $option);     //creating query
```

```
$single_user=$con->executeQuery("mydb3.mycol", $read); //executing query

//creating table view without column for lab

echo "<table border=3 cellspacing=5 cellpadding=7><thead><th>ID<th>Name<th>Certificate
No.<th>Course<th>Start Date<th>End Date</thead>";

foreach($single_user as $user){    //parsing results in loop echo "<tr>";

echo

"<td>".$user->_id."<td>".$user->name."<td>".$user->certNo."<td>".$user->course."<td>".$user

->start."<td>".$user->end;

}          end loop

echo "</table>";       //end table

?>
```

Output:



localhost/mongo/mongo8.php

Database Connected

ID	Name	Lab	Certificate No.	Course	Start Date	End Date
006	SHreya	PHP	106	IBM	2019	2021

References:

- HTML Introduction - https://www.w3schools.com/
- https://www.tutorialspoint.com/mongodb/mongodb_php.htm

Activity 13

Aim: Create a MongoDB query to find the course where maximum trainees attended

Learning outcome: Able to configure embedded databases with different web pages using MongoDB.

Duration: 2 hours

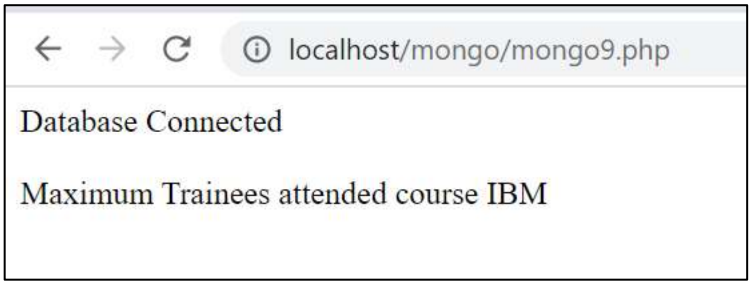
List of Hardware/Software requirements:

1. Laptop/Computer with Linux OS - Ubuntu 18.04 LTS
2. MongoDB 4.2 Community Edition
3. XAMPP- PHP 7
4. PHP-MongoDB driver 1.7.4

Program:

```
<?php

//connecting to database

if($con=new MongoDB\Driver\Manager("mongodb://localhost:27017"))

echo "Database Connected<br><br>";

//no filter or options

$filter=[ ];

$option=[ ];

$ibm=0; //counetr for IBM course

$ef=0; //counter for EF course

$read=new MongoDB\Driver\Query($filter, $option); //create query

$single_user=$con->executeQuery("mydb3.mycol", $read); //execute query

foreach($single_user as $user){ //loop for parsing all data

if($user->course=='IBM')

$ibm++; //counter update if course is IBM

else $ef++; //counter update if course is EF
}
if($ibm>$ef) echo('Maximum Trainees attended course IBM'); //display if IBM is max
else echo('Maximum Trainees attended course EF'); //display if EF is max
?>
```

Output:



```
Database Connected

Maximum Trainees attended course IBM
```

Activity 14

Aim: Create a MongoDB query to find lab wise details of trainees.

Learning outcome: Able to configure embedded databases with different web pages using MongoDB.

Duration: 2 hours

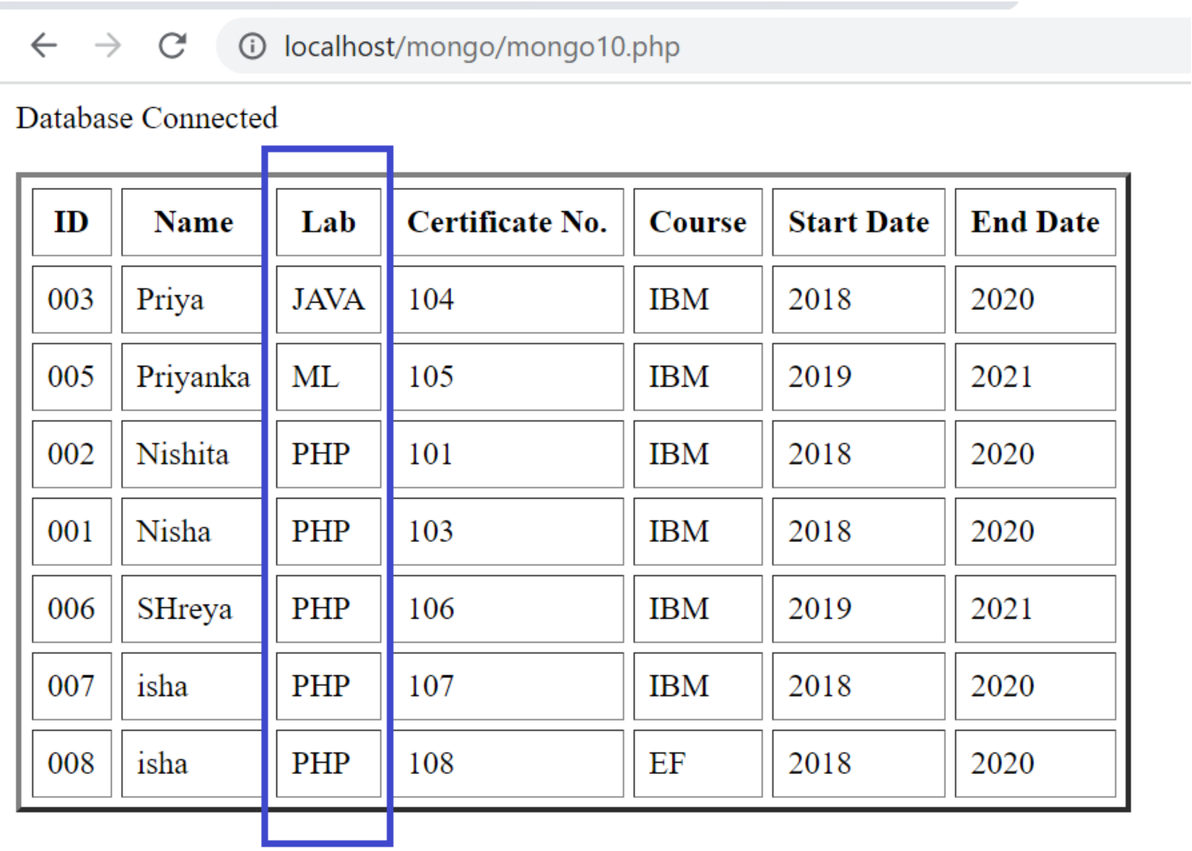
List of Hardware/Software requirements:

1. Laptop/Computer with Linux OS - Ubuntu 18.04 LTS
2. MongoDB 4.2 Community Edition
3. XAMPP- PHP 7
4. PHP-MongoDB driver 1.7.4

Program:

```
<?php

//connecting to database

if($con=new MongoDB\Driver\Manager("mongodb://localhost:27017"))

echo "Database Connected<br><br>";

$filter=[ ]; //no filters

$option=[ 'sort'=>['lab'=>1] ]; //sorting options on course name

$read=new MongoDB\Driver\Query($filter, $option); //create query

$single_user=$con->executeQuery("mydb3.mycol", $read); //execute query

//create table view

echo "<table border=3 cellspacing=5

cellpadding=7><thead><th>ID<th>Name<th>Lab<th>Certificate No.<th>Course<th>Start

Date<th>End Date</thead>";

foreach($single_user as $user){ //parsing results though loop

echo "<tr>";
echo

"<td>".$user->_id."<td>".$user->name."<td>".$user->lab."<td>".$user->certNo."<td>".$user->course."<td>".$user->start."<td>".$user->end;

} //end loop

echo "</table>"; //end table

?>
```

Output:



localhost/mongo/mongo10.php

Database Connected

ID	Name	Lab	Certificate No.	Course	Start Date	End Date
003	Priya	JAVA	104	IBM	2018	2020
005	Priyanka	ML	105	IBM	2019	2021
002	Nishita	PHP	101	IBM	2018	2020
001	Nisha	PHP	103	IBM	2018	2020
006	SHreya	PHP	106	IBM	2019	2021
007	isha	PHP	107	IBM	2018	2020
008	isha	PHP	108	EF	2018	2020

Activity 15

Aim: Create a MongoDB query to find Course wise details of trainees.

Learning outcome: Able to configure embedded databases with different web pages using MongoDB.

Duration: 2 hours

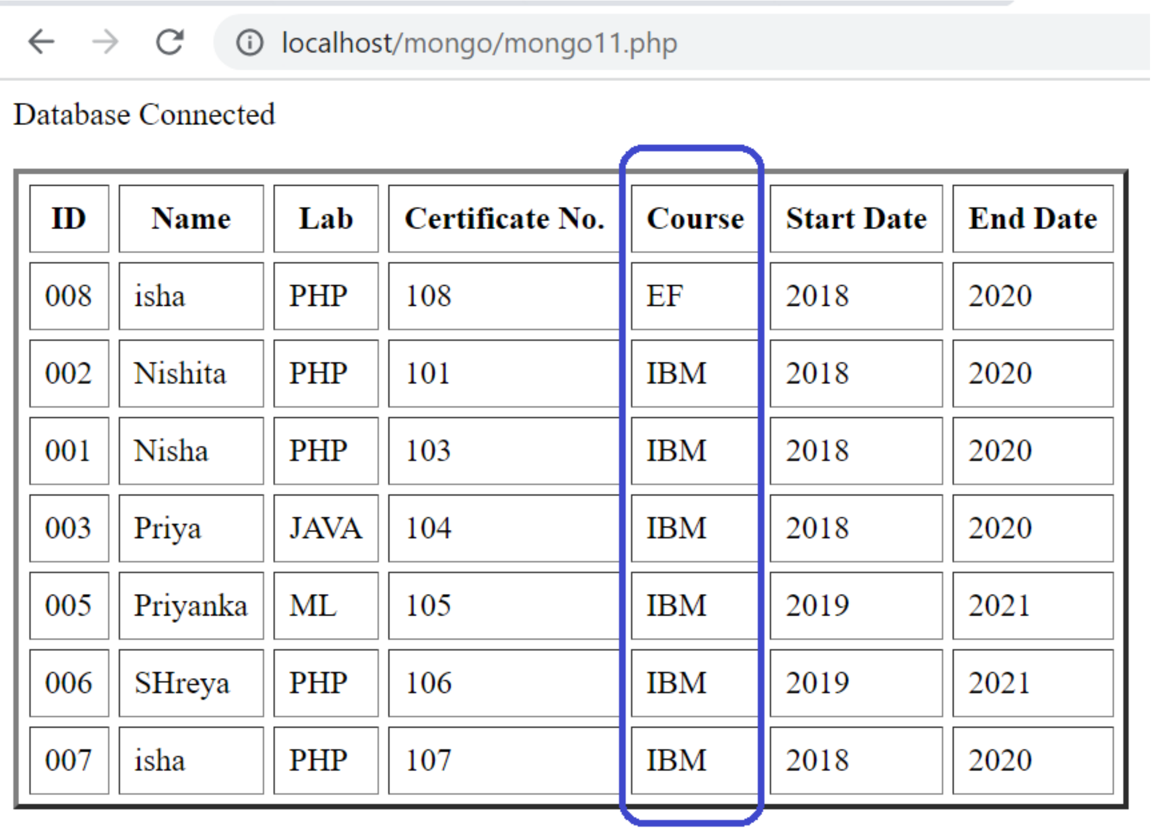
List of Hardware/Software requirements:

1. Laptop/Computer with Linux OS - Ubuntu 18.04 LTS
2. MongoDB 4.2 Community Edition
3. XAMPP- PHP 7
4. PHP-MongoDB driver 1.7.4

Program:

```
<?php

//connecting to database

if($con=new MongoDB\Driver\Manager("mongodb://localhost:27017"))

echo "Database Connected<br><br>";

$filter=[ ]; //no filters

$option=[ 'sort'=>['course'=>1] ]; //sorting options on course name

$read=new MongoDB\Driver\Query($filter, $option); //create query

$single_user=$con->executeQuery("mydb3.mycol", $read); //execute query

//create table view

echo "<table border=3 cellspacing=5

cellpadding=7><thead><th>ID<th>Name<th>Lab<th>Certificate No.<th>Course<th>Start

Date<th>End Date</thead>";

foreach($single_user as $user){ //parsing results though loop

echo "<tr>";
echo

"<td>".$user->_id."<td>".$user->name."<td>".$user->lab."<td>".$user->certNo."<td>".$user->course."<td>".$user->start."<td>".$user->end;

} //end loop

echo "</table>"; //end table

?>
```

Output:



Database Connected

ID	Name	Lab	Certificate No.	Course	Start Date	End Date
008	isha	PHP	108	EF	2018	2020
002	Nishita	PHP	101	IBM	2018	2020
001	Nisha	PHP	103	IBM	2018	2020
003	Priya	JAVA	104	IBM	2018	2020
005	Priyanka	ML	105	IBM	2019	2021
006	SHreya	PHP	106	IBM	2019	2021
007	isha	PHP	107	IBM	2018	2020

Activity 16

Aim: Create MongoDB cluster in MongoDB Atlas cloud service

Learning outcome: Able to create MongoDB cluster on MongoDB Atlas cloud platform.

Duration: 2 hours

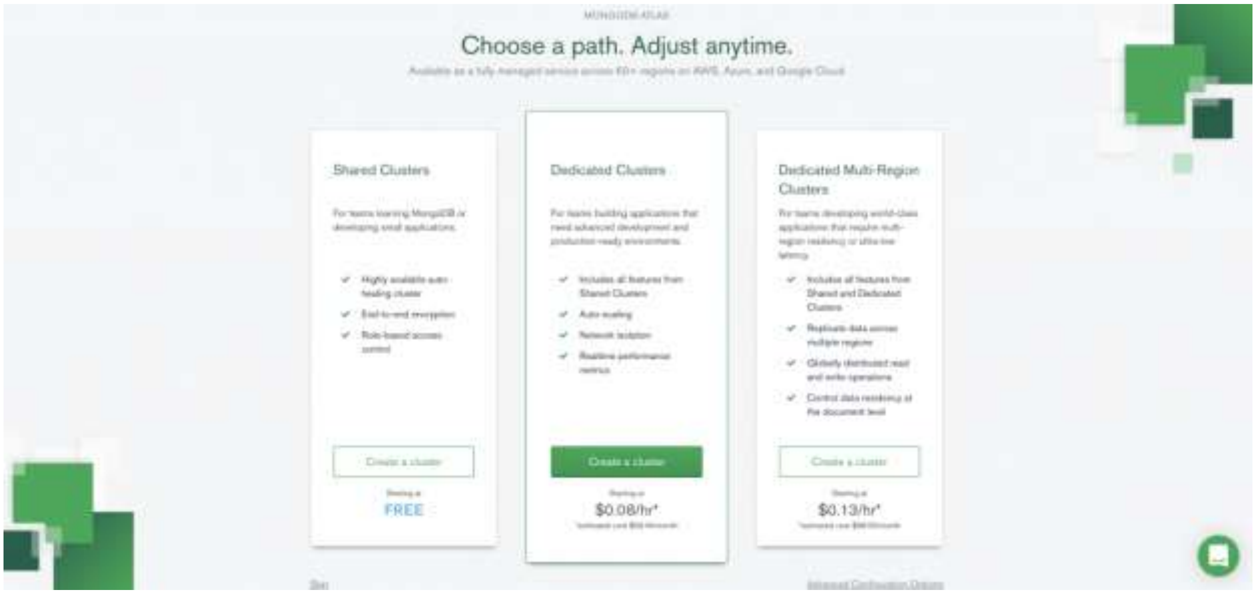
List of Hardware/Software requirements:

1. Laptop/Computer with Linux OS - Ubuntu 18.04 LTS / Windows
2. MongoDB 4.2 Community Edition
3. XAMPP- PHP 7
4. PHP-MongoDB driver 1.7.4
5. MongoDB Atlas cloud account

Procedure/ code:

Set up a free MongoDB Atlas cluster

1. Go to the MongoDB Atlas landing page.
2. Fill in the required information (email address, first name, last name, and password).
3. Click the **terms of service and privacy policy links**, which should open on a new tab.
4. If you want to continue with the registration, select the **I agree to the terms of service and privacy policy** check box.
5. Click the **Get started free** at the bottom of the form.
6. The website will ask to choose a cluster. Choose **Starter Clusters** and click on **Create a cluster**.
7. You should also receive a welcome email, which confirms that you're registered with MongoDB Atlas and includes a link for logging onto the service.



MONGODB ATLAS

Choose a path. Adjust anytime.

Available as a fully managed service across 60+ regions on AWS, Azure, and Google Cloud

Shared Clusters

For teams learning MongoDB or developing small applications.

- ✓ Highly available auto-healing cluster
- ✓ End-to-end encryption
- ✓ Role-based access control

Create a cluster

Starting at FREE

Dedicated Clusters

For teams building applications that need advanced development and production-ready environments.

- ✓ Includes all features from Shared Clusters
- ✓ Auto-scaling
- ✓ Network isolation
- ✓ Realtime performance metrics

Create a cluster

Starting at $0.08/hr*

*estimated cost [illegible]

Dedicated Multi-Region Clusters

For teams developing world-class applications that require multi-region resiliency or ultra-low latency.

- ✓ Includes all features from Shared and Dedicated Clusters
- ✓ Replicate data across multiple regions
- ✓ Globally distributed read and write operations
- ✓ Control data residency at the document level

Create a cluster

Starting at $0.13/hr*

*estimated cost [illegible]

Skip

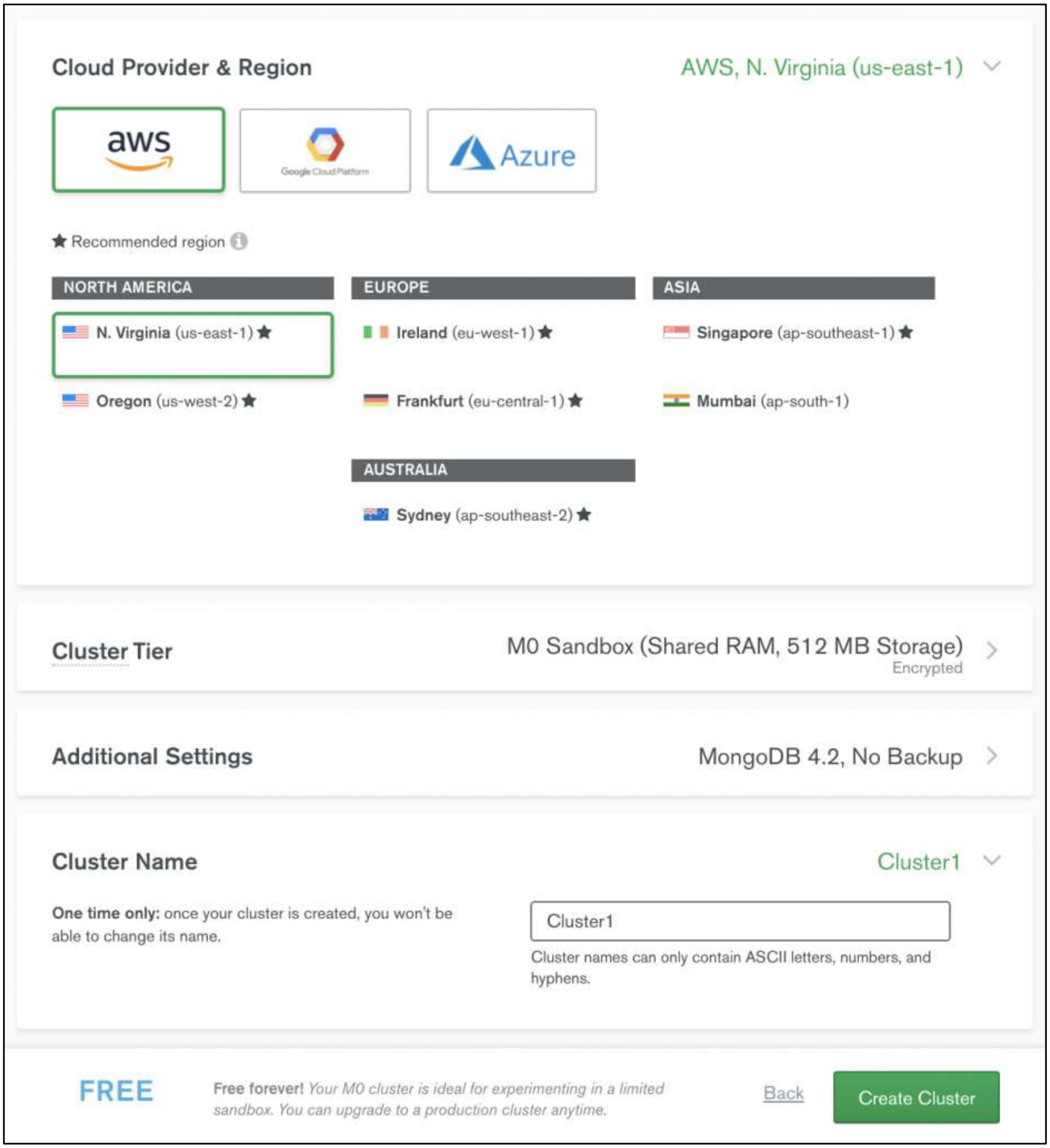
Advanced Configuration Options

7. In the **Cloud Provider & Region** section, the **aws** option should be selected as the default provider, but you can select any provider. All three platforms support the free tier.

8. Beneath the list of providers, select a region.

9. Expand the **Cluster Tier** section and ensure that **M0 Sandbox** is selected. This is the free M0 service level.

10. Expand the **Cluster Name** section and type **Cluster1** in the text box.



Cloud Provider & Region — AWS, N. Virginia (us-east-1)

★ Recommended region

NORTH AMERICA
- N. Virginia (us-east-1) ★
- Oregon (us-west-2) ★

EUROPE
- Ireland (eu-west-1) ★
- Frankfurt (eu-central-1) ★

ASIA
- Singapore (ap-southeast-1) ★
- Mumbai (ap-south-1)

AUSTRALIA
- Sydney (ap-southeast-2) ★

Cluster Tier — M0 Sandbox (Shared RAM, 512 MB Storage)
Encrypted

Additional Settings — MongoDB 4.2, No Backup

Cluster Name — Cluster1

One time only: once your cluster is created, you won't be able to change its name.

Cluster1

Cluster names can only contain ASCII letters, numbers, and hyphens.

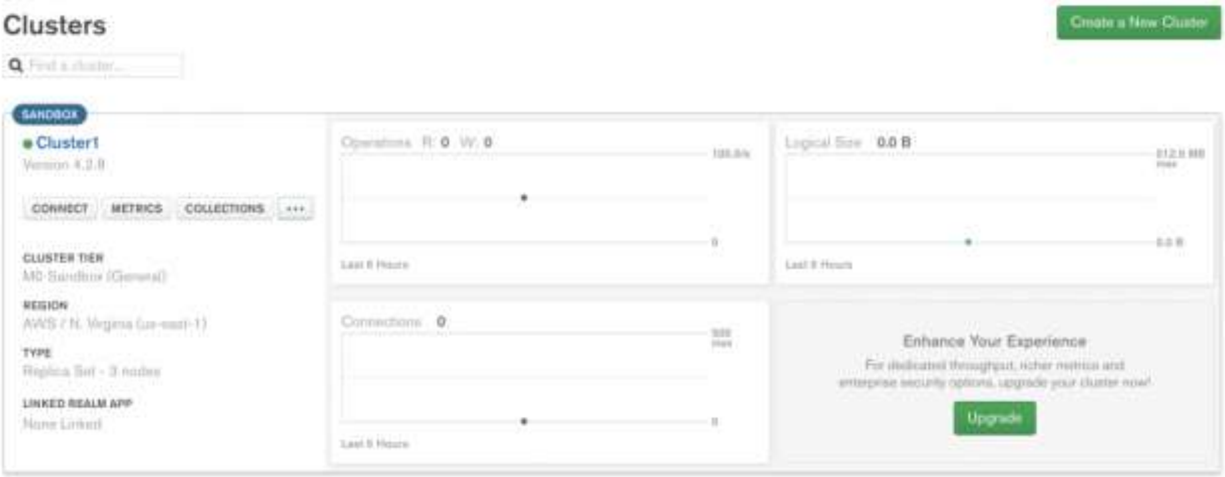
FREE

Free forever! *Your M0 cluster is ideal for experimenting in a limited sandbox. You can upgrade to a production cluster anytime.*

11. Click the **Create Cluster** button at the bottom of the web page.

12. You should then receive a message stating that your cluster is being created.
When the process is complete, you'll be taken to the Clusters page, which includes a listing for your new cluster, as shown in the following figure.

Output:



Cluster1
Version 4.2.8

CLUSTER TIER
M0 Sandbox (General)

REGION
AWS / N. Virginia (us-east-1)

TYPE
Replica Set - 3 nodes

LINKED REALM APP
None Linked

Operations R: 0 W: 0

Logical Size 0.0 B

Connections 0

Enhance Your Experience

For dedicated throughput, richer metrics and enterprise security options, upgrade your cluster now!

The cluster information includes the service level, cloud provider, and region, along with details about operations, connections, and logical size, all of which currently show zero amounts.

Reference:

- https://studio3t.com/knowledge-base/articles/mongodb-atlas-tutorial/
- https://cloud.mongodb.com/v2/60a4bd5bf765a20087add3a7#clusters/edit?from=ctaClusterHeader

Activity 17

Aim: Connect and use MongoDB Atlas on local system

Learning outcome: configure MongoDB Atlas to connect to the cluster you created.

Duration: 2 hours

List of Hardware/Software requirements:

1. Laptop/Computer with Linux OS - Ubuntu 18.04 LTS / Windows
2. MongoDB 4.2 Community Edition
3. XAMPP- PHP 7
4. PHP-MongoDB driver 1.7.4
5. MongoDB Atlas cloud account

Procedure/ code:

Configure IP address and connection string

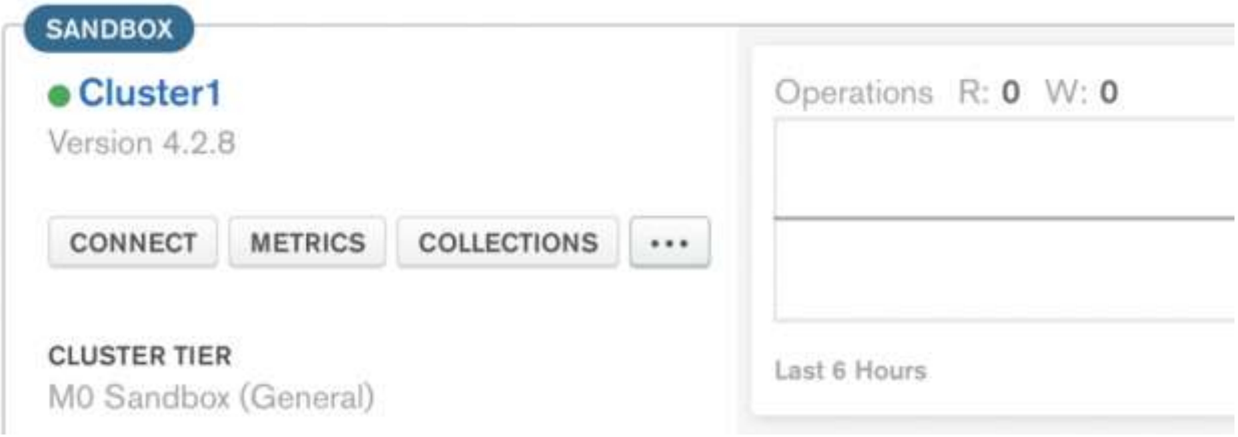
you will configure MongoDB Atlas to connect to the cluster you created.

For this, you will need the IP address of the device that will connect to the service.

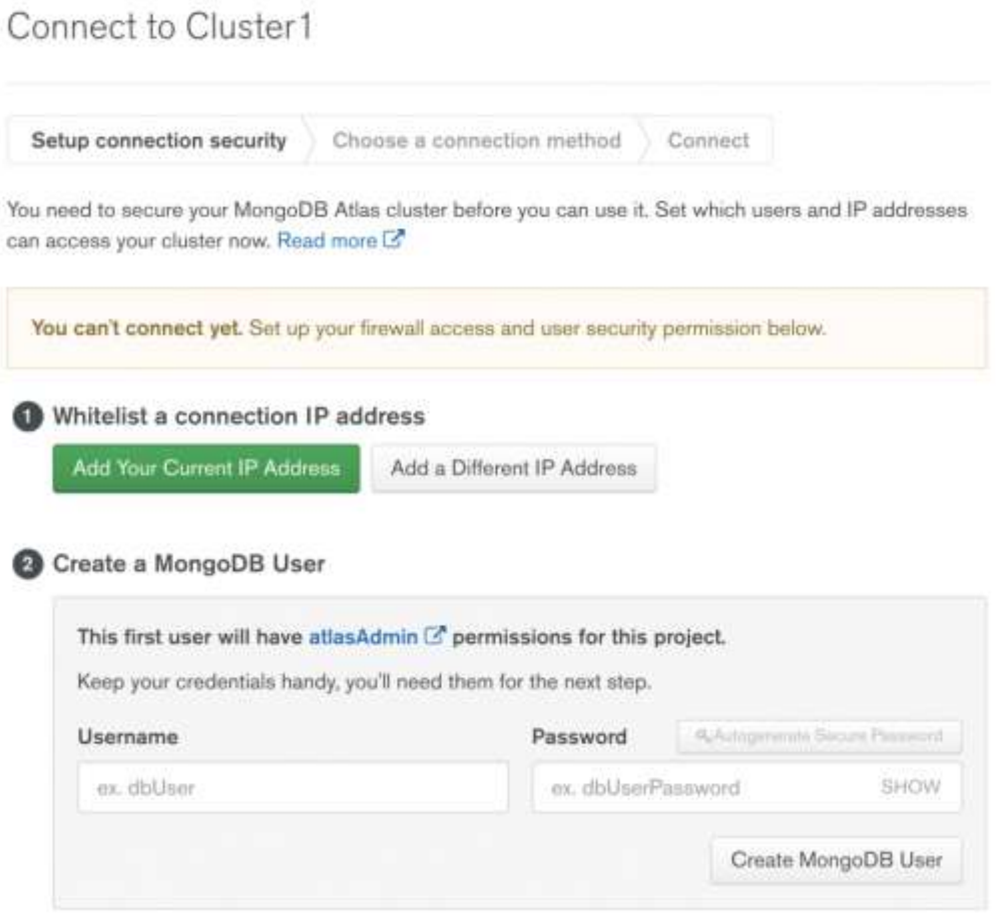
If you plan to connect to MongoDB Atlas on the same device where you're setting up the service, MongoDB Atlas can find the local IP address automatically.

As part of this exercise, you'll also set up an administrator account for accessing the cluster.

1. In the **Cluster1** section of the **Clusters** page, click the **CONNECT** button in the left pane.



The **Connect to Cluster1** dialog box appears, showing the two steps you must take to configure your connection.



Connect to Cluster1

Setup connection security
Choose a connection method
Connect

You need to secure your MongoDB Atlas cluster before you can use it. Set which users and IP addresses can access your cluster now. Read more

You can't connect yet. Set up your firewall access and user security permission below.

1 Whitelist a connection IP address

Add Your Current IP Address
Add a Different IP Address

2 Create a MongoDB User

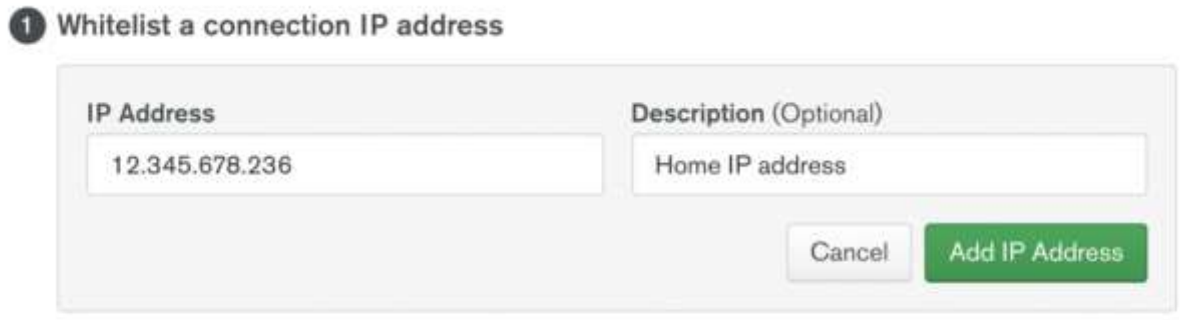
This first user will have atlasAdmin permissions for this project.

Keep your credentials handy, you'll need them for the next step.

Username: ex. dbUser

Password: ex. dbUserPassword

2. For security reasons, MongoDB Atlas blocks all outside connections by default. In order to connect, you must first whitelist your IP address. In the Step 1 section, do one of the following:



1 Whitelist a connection IP address

IP Address: 12.345.678.236

Description (Optional): Home IP address

- If you plan to connect from the computer you’re currently using, click **Add Your Current IP Address** and then click **Add IP Address**.
- If you plan to connect from a different computer, click **Add a Different IP Address**, type the IP address, and then click **Add IP Address**. You can edit the IP address, add IP addresses, or delete them at any time in case you change devices.

3. In the Step 2 section, type **admin** in the **Username** text box (or whatever name you want to use), and then type a password in the **Password** text box.

To make it easier to connect to MongoDB Atlas from Studio 3T, your password should include only alphanumeric characters, that is, letters and numbers only with no special characters.

If you use special characters, you will need to encode them when creating a connection string for accessing the MongoDB service.

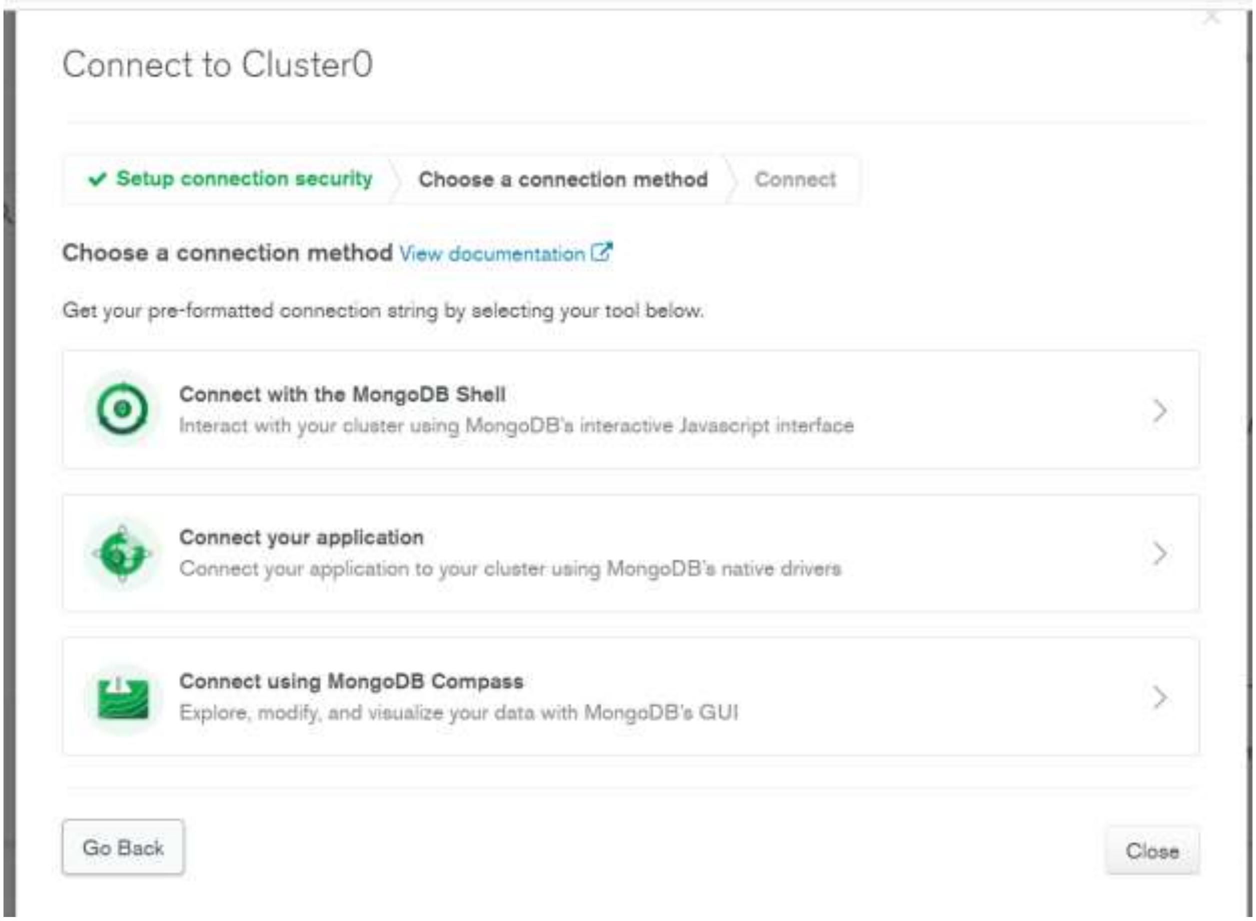
Note: To generate a password automatically, click the **Autogenerate Secure Password** button. A password will be generated that includes only alphanumeric characters. Be sure to save the password somewhere secure.

4. Click the **Create MongoDB User** button.

5. The next task is to generate the connection string or Uniform Resource Identifier (URI).

To start this process, click the **Choose a connection method** button. The Connect to Cluster1 dialog box reappears, providing a different set of options.

6. Click **Connect Your Application**. You’ll again be presented with two steps.



Connect to Cluster0

Choose a connection method View documentation

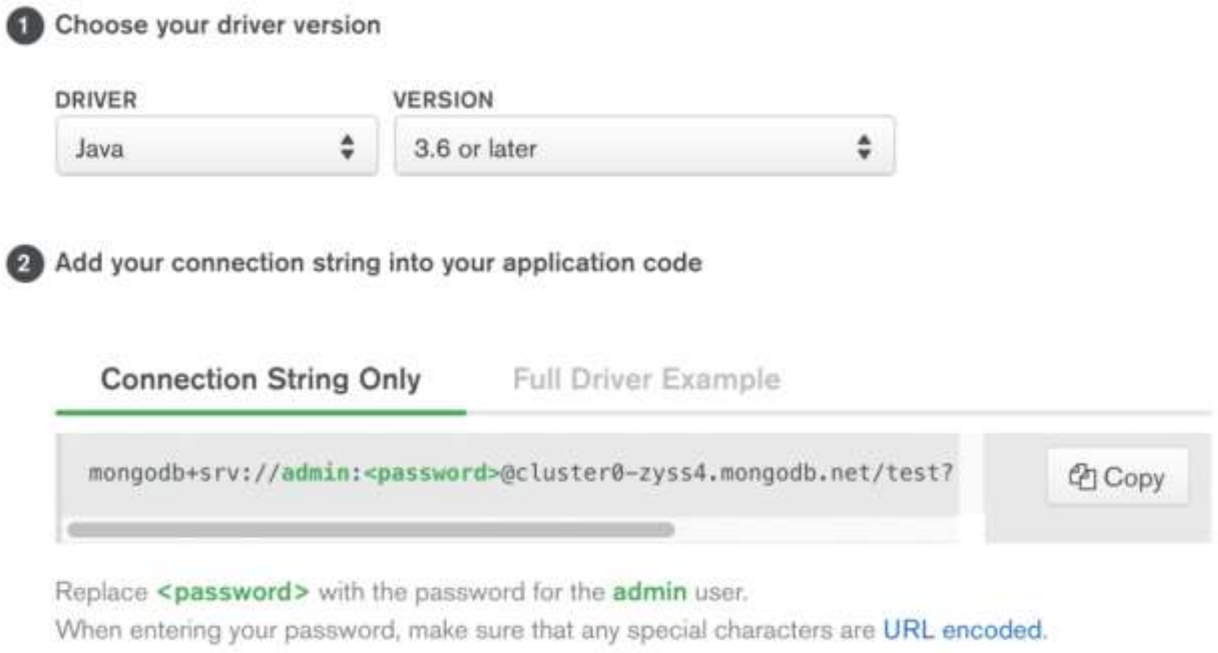
Get your pre-formatted connection string by selecting your tool below.

Connect with the MongoDB Shell
Interact with your cluster using MongoDB's interactive Javascript interface

Connect your application
Connect your application to your cluster using MongoDB's native drivers

Connect using MongoDB Compass
Explore, modify, and visualize your data with MongoDB's GUI

7. In the Step 1 section, select **Java** from the **DRIVER** drop-down list, and select **3.6 or later** from the **VERSION** drop-down list.



1 Choose your driver version
DRIVER
Java
VERSION
3.6 or later
2 Add your connection string into your application code
Connection String Only
Full Driver Example
Copy

```
mongodb+srv://admin:<password>@cluster0-zyss4.mongodb.net/test?
```

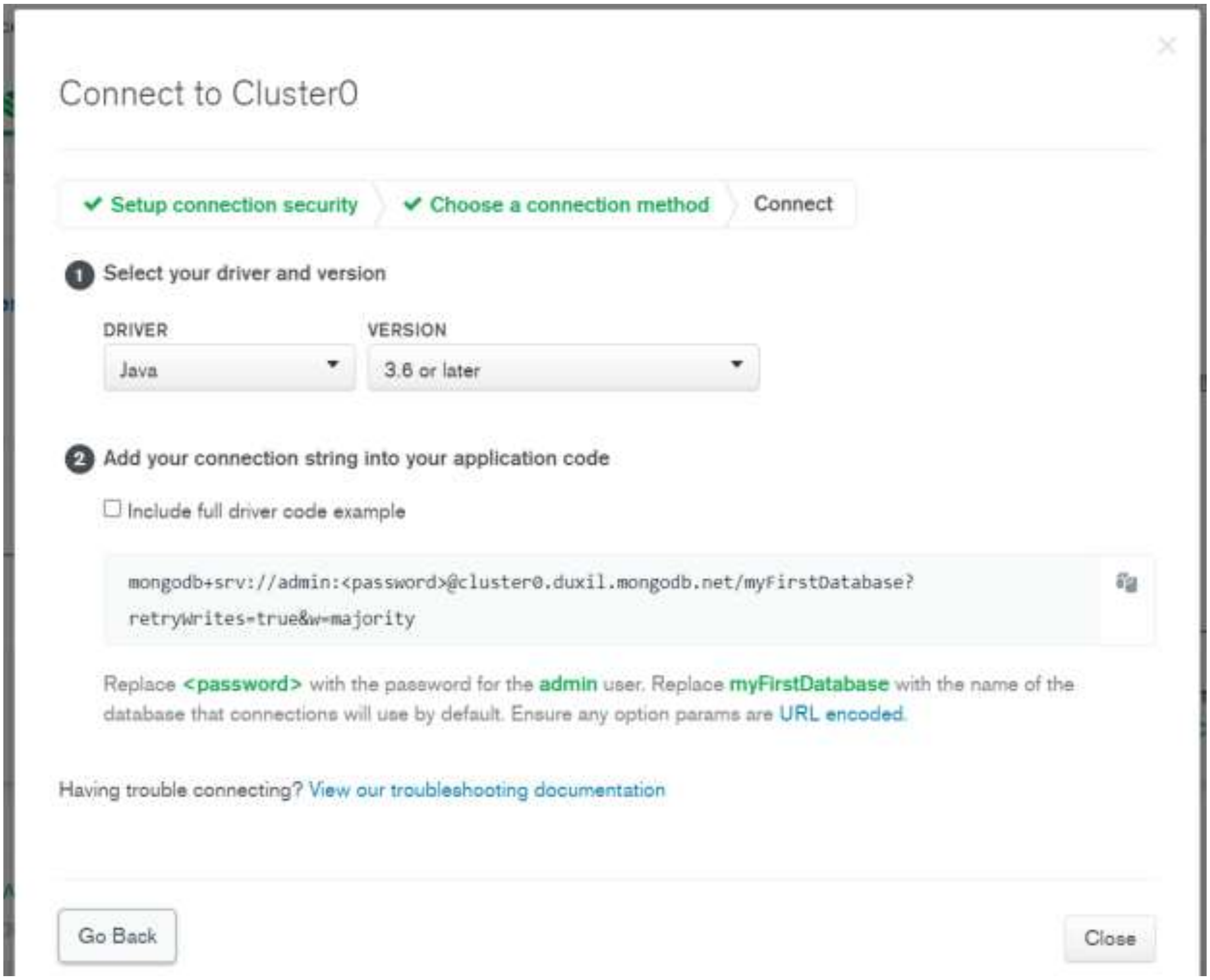
Replace <password> with the password for the admin user.
When entering your password, make sure that any special characters are URL encoded.

8. In the Step 2 section, click **Copy** to copy the connection string to your clipboard, and then paste the connection string to a safe location.

When you use the connection string or URI, you must replace the placeholder with the password you created for the administrator account. **Don't forget to remove the <> as well.**

9.Click **Close** to close the Connect to Cluster1 dialog box, and then sign out of the MongoDB Atlas service.

Output:



Connect to Cluster0

1 Select your driver and version

DRIVER: Java

VERSION: 3.6 or later

2 Add your connection string into your application code

□ Include full driver code example

```
mongodb+srv://admin:<password>@cluster0.duxil.mongodb.net/myFirstDatabase?
retryWrites=true&w=majority
```

Replace <password> with the password for the admin user. Replace myFirstDatabase with the name of the database that connections will use by default. Ensure any option params are URL encoded.

Having trouble connecting? View our troubleshooting documentation

Reference:

- https://studio3t.com/knowledge-base/articles/mongodb-atlas-tutorial/

Activity 18

Aim: Get familiar with big data and its tools

Learning outcome: Get familiar with big data and its tools

Duration: 2 hours

List of Hardware/Software requirements:

1. Laptop/Computer with Linux OS - Ubuntu 18.04 LTS / Windows
2. Big Data Tools

Procedure:

Big data is simply too large and complex data that cannot be dealt with using traditional data processing methods.
Big Data requires a set of tools and techniques for analysis to gain insights from it.

There are a number of big data tools available in the market such as Hadoop which helps in storing and processing large data, Spark helps in-memory calculation, Storm helps in faster processing of unbounded data, Apache Cassandra provides high availability and scalability of a database, MongoDB provides cross-platform capabilities, so there are different functions of every Big Data tool.
Analyzing and processing Big Data is not an easy task. Big Data is one big problem and to deal with it you need a set of great big data tools that will not only solve this problem but also help you in producing substantial results.

What are the best Big Data Tools?

Here is the list of top 10 big data tools –

- Apache Hadoop
- Apache Spark
- Flink
- Apache Storm
- Apache Cassandra
- MongoDB
- Kafka

- Tableau
- RapidMiner
- R Programming

Big Data is an essential part of almost every organization these days and to get significant results through Big Data Analytics a set of tools is needed at each phase of data processing and analysis. There are a few factors to be considered while opting for the set of tools i.e., the size of the datasets, pricing of the tool, kind of analysis to be done, and many more.

With the exponential growth of Big Data, the market is also flooded with its various tools. These tools used in big data help in bringing out better cost efficiency and thus increases the speed of analysis.

Apache Hadoop

Apache Hadoop is one of the most popularly used tools in the Big Data industry.

Hadoop is an open-source framework from Apache and runs on commodity hardware. It is used to store process and analyze Big Data.

Hadoop is written in Java. Apache Hadoop enables parallel processing of data as it works on multiple machines simultaneously. It uses clustered architecture. A Cluster is a group of systems that are connected via LAN.

It consists of 3 parts-

1. **Hadoop Distributed File System (HDFS) –** It is the storage layer of Hadoop.
2. **Map-Reduce –** It is the data processing layer of Hadoop.
3. **YARN –** It is the resource management layer of Hadoop.

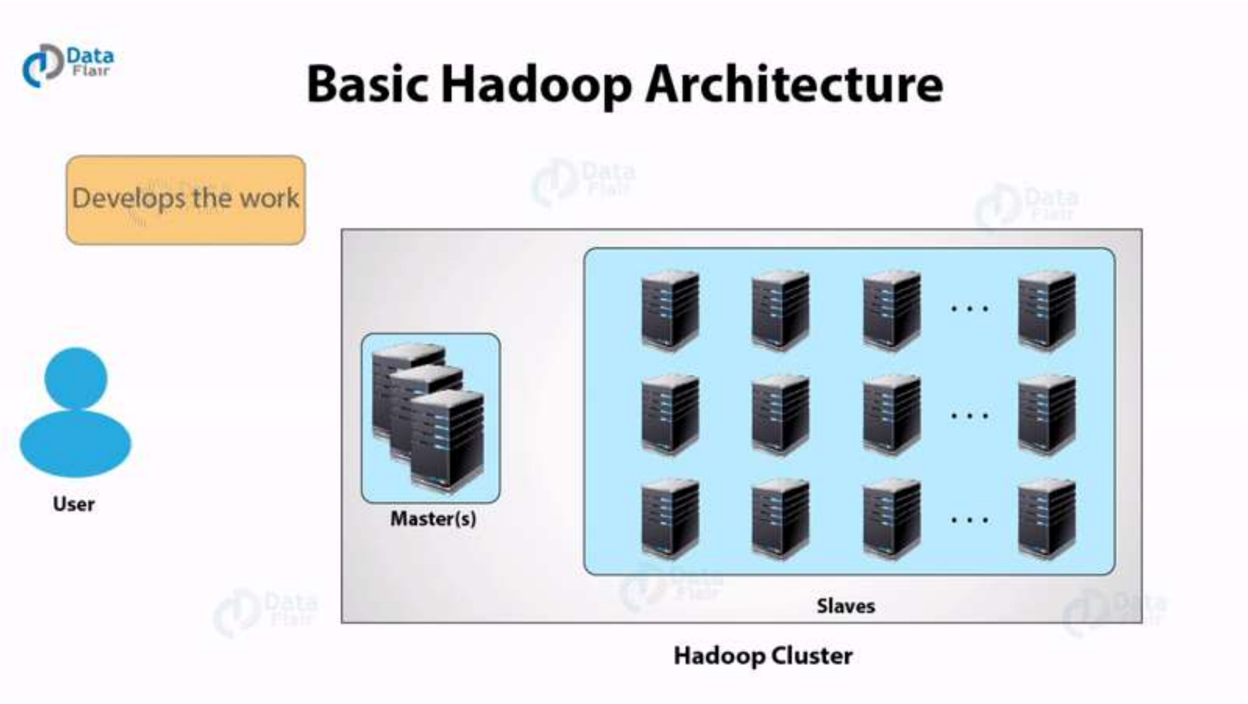


Image Source: https://data-flair.training/blogs/top-big-data-tools/

Limitation of Apache Hadoop:

- Hadoop does not support real-time processing. It only supports batch processing.
- Hadoop cannot do in-memory calculations.

Apache Spark

Apache Spark can be considered as the successor of Hadoop as it overcomes the drawbacks of it. Spark, unlike Hadoop, supports both real-time as well as batch processing. It is a general-purpose clustering system.
It also supports in-memory calculations, which makes it 100 times faster than Hadoop. This is made possible by reducing the number of read/write operations into the disk.

It provides more flexibility and versatility as compared to Hadoop since it works with different data stores such as HDFS, OpenStack and Apache Cassandra.

It offers high-level APIs in Java, Python, Scala and R. Spark also offers a substantial set of high-level tools including Spark SQL for structured data processing, MLlib for machine learning, GraphX for graph data set processing, and Spark Streaming. It also consists of 80 high-level operators for efficient query execution.

Apache Storm

Apache Storm is an open-source big data tool, distributed real-time and fault-tolerant processing system. It efficiently processes unbounded streams of data.

By unbounded streams, we refer to the data that is ever-growing and has a beginning but no defined end.

The biggest advantage of Apache Storm is that it can be used with any of the programming languages and it further supports JSON based protocols.

The processing speed of Storm is very high. It is easily scalable and also fault-tolerant. It is much easier to use.

On the other hand, it guarantees the processing of each data set. It's processing speed is rapid and a standard observed was as high as a million tuples processed per second on each node.

Apache Cassandra

Apache Cassandra is a distributed database that provides high availability and scalability without compromising performance efficiency. It is one of the best big data tools that can accommodate all types of data sets namely structured, semi-structured, and unstructured.

It is the perfect platform for mission-critical data with no single point of failure and provides fault tolerance on both commodity hardware and cloud infrastructure.

Cassandra works quite efficiently under heavy loads. It does not follow master-slave architecture so all nodes have the same role. Apache Cassandra supports the ACID (Atomicity, Consistency, Isolation, and Durability) properties.

MongoDB

MongoDB is an open-source data analytics tool, NoSQL database that provides cross-platform capabilities. It is exemplary for a business that needs fast-moving and real-time data for taking decisions.

MongoDB is perfect for those who want data-driven solutions. It is user-friendly as it offers easier installation and maintenance. MongoDB is reliable as well as cost-effective.

It is written in C, C++, and JavaScript. It is one of the most popular databases for Big Data as it facilitates the management of unstructured data or the data that changes frequently.

MongoDB uses dynamic schemas. Hence, you can prepare data quickly. This allows in reducing the overall cost. It executes on MEAN software stack, NET applications and, Java platform. It is also flexible in cloud infrastructure.

But a certain downfall in the processing speed has been noticed for some use-cases.

Apache Flink

Apache Flink is an Open-source data analytics tool distributed processing framework for bounded and unbounded data streams. It is written in Java and Scala. It provides high accuracy results even for late-arriving data.

Flink is a stateful and fault-tolerant i.e. it has the ability to recover from faults easily. It provides high-performance efficiency at a large scale, performing on thousands of nodes.

It gives a low-latency, high throughput streaming engine and supports event time and state management.

Kafka

Apache Kafka is an open-source platform that was created by LinkedIn in the year 2011.

Apache Kafka is a distributed event processing or streaming platform which provides high throughput to the systems. It is efficient enough to handle trillions of events a day. It is a streaming platform that is highly scalable and also provides great fault tolerance.

The streaming process includes publishing and subscribing to streams of records alike to the messaging systems, storing these records durably, and then processing these records. These records are stored in groups called topics.

Apache Kafka offers high-speed streaming and guarantees zero downtime.

Tableau

Tableau is one of the best data visualization and software solution tools in the Business Intelligence industry. It's a tool that unleashes the power of your data.

It turns your raw data into valuable insights and enhancing the decision-making process of the businesses.

Tableau offers a rapid data analysis process and resulted in visualizations are in the form of interactive dashboards and worksheets.

It works in synchronization with other **Big Data tools** such as Hadoop.

Tableau offered the capabilities of data blending are best in the market. It provides an efficient real-time analysis.

Tableau is not only bound to the technology industry but is a crucial part of some other industries as well. This software doesn’t require any technical or programming skills to operate.

RapidMiner

RapidMiner is a cross-platform tool that provides a robust environment for Data Science, Machine Learning and Data Analytics procedures. It is an integrated platform for the complete Data Science lifecycle starting from data prep to machine learning to predictive model deployment.

It offers various licenses for small, medium, and large proprietary editions. Apparently, it also offers a free edition that permits only 1 logical processor and up to 10,000 data rows.

RapidMiner is an open-source tool that is written in java. RapidMiner offers high efficiency even when integrated with APIs and cloud services. It provides some robust Data Science tools and algorithms.

R Programming

R is an open-source programming language and is one of the most comprehensive statistical analysis languages.

It is a multi-paradigm programming language that offers a dynamic development environment. As it is an open-source project and thousands of people have contributed to the development of the R.

R is written in C and Fortran. It is one of the most widely used statistical analysis tools as it provides a vast package ecosystem.

It facilitates the efficient performance of different statistical operations and helps in generating the results of data analysis in graphical as well as text format. The graphics and charting benefits it provides are unmatchable.

Reference:

- https://data-flair.training/blogs/top-big-data-tools/

Learning outcome

Able to design and develop dynamic websites with PHP.

After achieving this learning outcome, a student will be able to design and develop dynamic websites with PHP. In order to achieve this learning outcome, a student has to complete the following:

1. Capturing Form Data Dealing with Multi-value field Generating File uploaded form (2 Hrs)
2. Redirecting a form after submission (2 Hrs)
3. Write a PHP script to get the PHP version and configuration information (30 mins)
4. Write a PHP script to display the strings (1 Hr)
5. Create a simple HTML form and accept the user name and display the name through PHP echo statement (1 Hrs)
6. Write a e PHP script to display string, values within a table (2 Hrs)
7. Write a PHP script to count lines in a file (30 mins)
8. Write a PHP function to test whether a number is greater than 30, 20 or 10 using ternary operator (1 Hr)
9. Write a script which will display the string (1 Hr)
10. Write a PHP script which will display the array (2 Hrs)
11. Write a PHP script to sorting (2 Hrs)
12. Write a PHP script to calculate and display average temperature, five lowest and highest temperatures in given data (2 Hr)
13. Write a program to calculate and print the factorial of a number using a for loop (2 Hr)
14. Write a PHP script using nested for loop (1 Hrs)
15. Write a PHP program to generate and display the first n lines of a Floyd (2 Hrs)
16. Write a function to calculate the factorial of a number (2 Hr)
17. Write a function to check a number is prime or not(1Hr)
18. Write a function to reverse a string.
19. Write a PHP function that checks whether a passed string is a palindrome or not?
20. Write a simple PHP class which displays the given string.
21. Write a PHP Calculator class which will accept two values as arguments, then add them,

subtract them,

multiply them together, or divide them on request.

22. Write a PHP script to: - a) transform a string all uppercase letters. b) transform a string all lowercase letters. c) make a string's first character uppercase. d) make a string's first character of all the words uppercase.
23. Create a form in PHP and apply validations.
24. Create a date and time from a number of parameters in mktime().
25. Create a date and time from the strtotime () function.
26. Create more dates/times from strtotime.
27. Output the dates for the next six Saturdays.
28. Create and retrieve a cookie.
29. Modify a cookie value (2Hrs).
30. Delete a cookie.
31. Check if cookies are enabled.
32. Select data with MySQLi(Object-oriented).
33. Select data with MySQLi(Object-oriented) and put result in an HTML table.
34. Select data with MySQLi(Procedural).
35. Select data with PDO (+Prepared statements).

Activity 1

Aim: Capturing Form Data Dealing with Multi-value field Generating File uploaded form

Learning outcome: Able to design and develop dynamic websites with PHP.

Duration: 2 hours

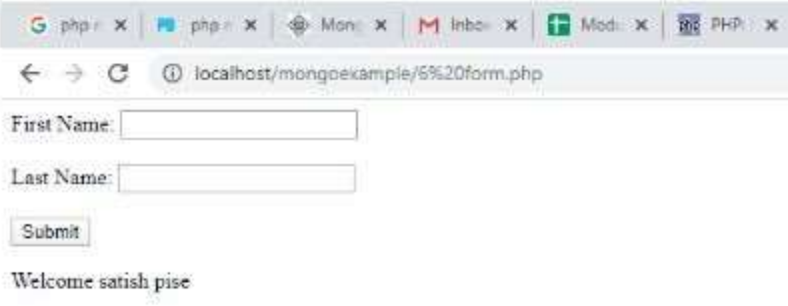
List of Hardware/Software requirements:

1. Laptop/Computer with Linux OS - Ubuntu 18.04 LTS/Windows
2. Apache 7.4 or above

Program:

```
//6 form.php example
<html>
<body>
<form action="6 form.php" method="post">
First Name: <input type="text" name="name"><br><br>
Last Name: <input type="text" name="lname"><br><br>
<input type="submit">
</form>
</body>
</html>
<html>
<body>
Welcome <?php echo $_POST["name"]." "; ?>
<?php echo $_POST["lname"]; ?>
</body>
</html>
```

Output/Results snippet:



localhost/mongoexample/5%20form.php

First Name:

Last Name:

Submit

Welcome satish pise

References:

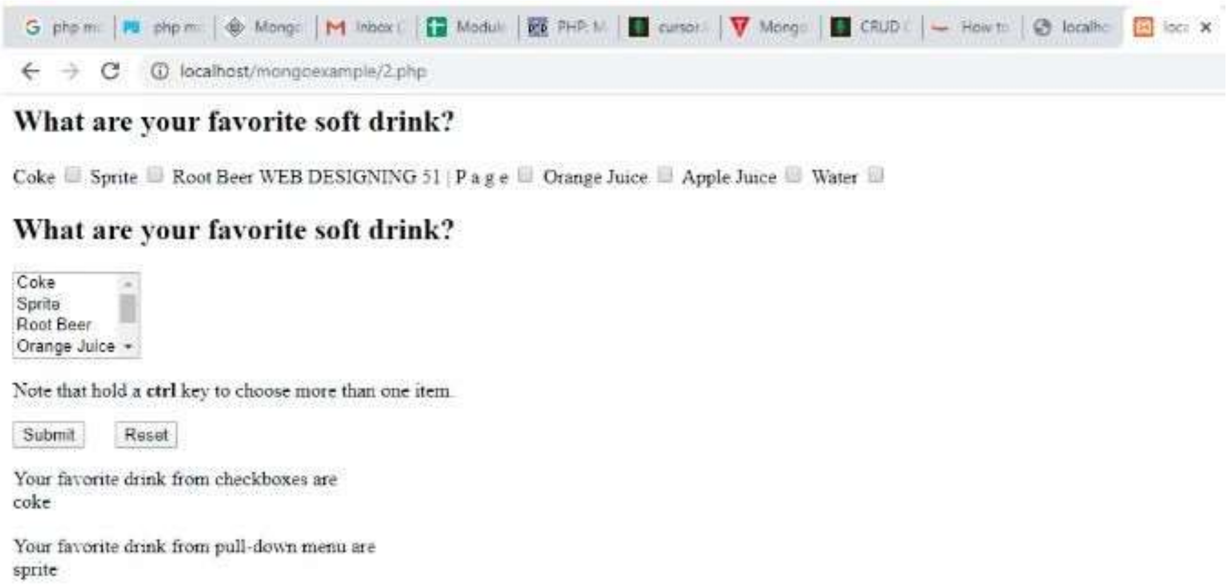
- HTML Introduction - https://www.w3schools.com/
- PHP - https://www.w3schools.com/

// Example 2 multi field

```
<form action="2.php" method="post">

<h2>What are your favorite soft drink?</h2>

<label>Coke</label>

<input type="checkbox" name="drink[]" value="coke" />

<label>Sprite</label>

<input type="checkbox" name="drink[]" value="sprite" />

<label>Root Beer</label>

<input type="checkbox" name="drink[]" value="root beer" />

<label>Orange Juice</label>

<input type="checkbox" name="drink[]" value="orange juice" />

<label>Apple Juice</label>

<input type="checkbox" name="drink[]" value="apple juice" />

<label>Water</label>

<input type="checkbox" name="drink[]" value="water" />
```

```
<h2>What are your favorite soft drink?</h2>
<select name="favor[]" size = "4" multiple = "multiple">
<option value="coke">Coke</option>
<option value="sprite">Sprite</option>
<option value="root beer">Root Beer</option>
<option value="orange juice">Orange Juice</option>
<option value="apple juice">Apple Juice</option>
<option value="water">Water</option>
</select>
<p>Note that hold a <b>ctrl</b> key to choose more than one item.</p>
<input type="submit" name="submit" id="moveRight" value="Submit" />
<input type="reset" name="reset" value="Reset" style="margin-left: 20px;display:inline;" />
</form>
<?php
$drinklist=$_POST["drink"]; //assign an array to a local array
$favorlist=$_POST["favor"];
echo "Your favorite drink from checkboxes are <br />";
foreach($drinklist as $drink)
echo $drink . "<br /> ";
echo "<br />";
echo "Your favorite drink from pull-down menu are <br /> ";
foreach($favorlist as $favor)
echo $favor. "<br />";
?>
```

Output/Results snippet:



localhost/mongoexample/2.php

What are your favorite soft drink?

Coke □ Sprite □ Root Beer WEB DESIGNING 51 | P a g e □ Orange Juice □ Apple Juice □ Water □

What are your favorite soft drink?

Coke
Sprite
Root Beer
Orange Juice

Note that hold a **ctrl** key to choose more than one item.

Submit Reset

Your favorite drink from checkboxes are
coke

Your favorite drink from pull-down menu are
sprite

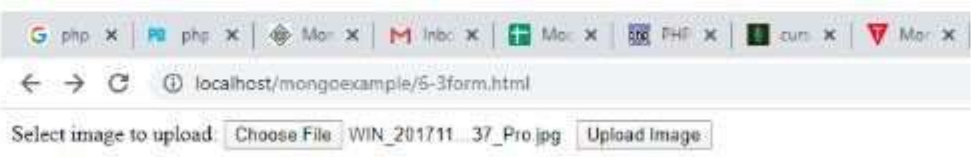
References:

- HTML Introduction - https://www.w3schools.com/
- PHP - https://www.w3schools.com/

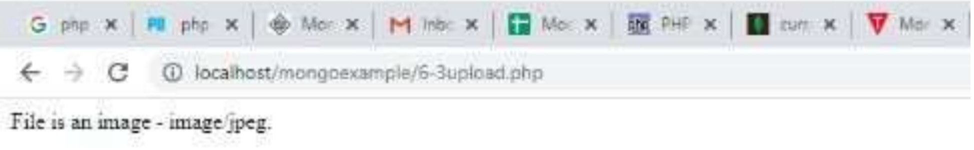
```
// File Upload example html form
<!DOCTYPE html>
<html>
<body>
<form action="6-3upload.php" method="post" enctype="multipart/form-
   data"> Select image to upload:
   <input type="file" name="fileToUpload" id="fileToUpload">
   <input type="submit" value="Upload Image" name="submit">
</form>
</body>
</html>
```

```
//File upload php file
<?php
$target_dir = "uploads/";
$target_file = $target_dir . basename($_FILES["fileToUpload"]["name"]);
$uploadOk = 1;
$imageFileType = strtolower(pathinfo($target_file,PATHINFO_EXTENSION));
// Check if image file is a actual image or fake image
if(isset($_POST["submit"])) {
  $check = getimagesize($_FILES["fileToUpload"]["tmp_name"]);
  if($check !== false) {
    echo "File is an image - " . $check["mime"] . ".";
    $uploadOk = 1;
  } else {
    echo "File is not an image.";
    $uploadOk = 0;
}
}?>
```

Output/Results snippet:



Select image to upload: Choose File WIN_201711...37_Pro.jpg Upload Image



File is an image - image/jpeg.

References:

- HTML Introduction - https://www.w3schools.com/
- PHP - https://www.w3schools.com/

Activity 2

Aim: Redirecting a form after submission

Learning outcome: Able to design and develop dynamic websites with PHP.

Duration: 2 hours

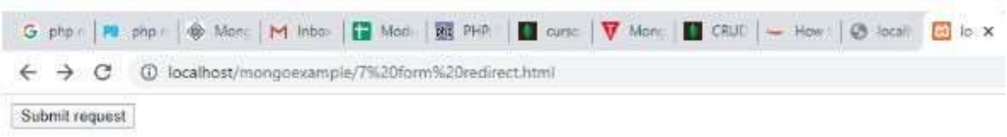
List of Hardware/Software requirements:

1. Laptop/Computer with Linux OS - Ubuntu 18.04 LTS/Windows
2. Apache 7.4 or above

Program:

```
<html>
  <body>
    <form action="https://google.com" method="POST">
      <input type="hidden" name="fullname" value="john" />
      <input type="hidden" name="address" value="street 2, 32 ave" />
      <input type="submit" value="Submit request" />
    </form>
  </body>
</html>
```

Output/Results snippet:



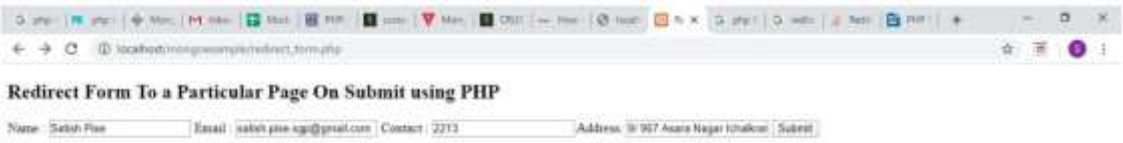
localhost/mongoexample/7%20form%20redirect.html

```
//PHP redirection – redirect_form.php
<!DOCTYPE html>
<html>
```

```
<head>
<title>Redirect Form To a Particular Page On Submit - Demo Preview</title>
<meta content="noindex, nofollow" name="robots">
<link href='css/redirect_form.css' rel='stylesheet' type='text/css'> <!--== Include CSS File Here ==-->
</head>
<body>
<div class="main">
<div class="first">
<h2>Redirect Form To a Particular Page On Submit using PHP</h2>
<form action="redirect_form.php" id="#form" method="post" name="#form">
<label>Name :</label>
<input id="name" name="name" placeholder='Your Name' type='text'>
<label>Email :</label>
<input id="email" name="email" placeholder='Valid Email Address' type='text'>
<label>Contact :</label>
<input id="contact" name="contact" placeholder='Contact' type='text'>
<label>Address:</label>
<input id="address" name="address" placeholder='Address' type='text' value="">
<input id='btn' name="submit" type='submit' value='Submit'>
<!---- Including PHP File Here -------- >
<?php
include "redirect.php";
?>
</form>
</div>
</div>
</body></html>
```

```
//Redirect.php
<?php
if(isset($_POST['submit'])){
// Fetching variables of the form which travels in URL
$name = $_POST['name'];
$email = $_POST['email'];
$contact = $_POST['contact'];
$address = $_POST['address'];
if($name !=''&& $email !=''&& $contact !=''&& $address !='')
{
// To redirect form on a particular page
header("Location:https://www.formget.com/app/");
}
else{
?><span><?php echo "Please fill all fields.....!!!!!!!!!!!!";?></span> <?php
}
}
?>
```

Output/Results snippet:



localhost/mongoexample/redirect_form.php

Redirect Form To a Particular Page On Submit using PHP

Name : Satish Pise Email : satish.pise.sgp@gmail.com Contact : 2213 Address: 9/ 967 Asara Nagar Ichalkar Submit

References:

- HTML Introduction - https://www.w3schools.com/
- PHP - https://www.w3schools.com/

Activity 3

Aim: Write a PHP script to get the PHP version and configuration information

Learning outcome: Able to design and develop dynamic websites with PHP.

Duration: 30 mins

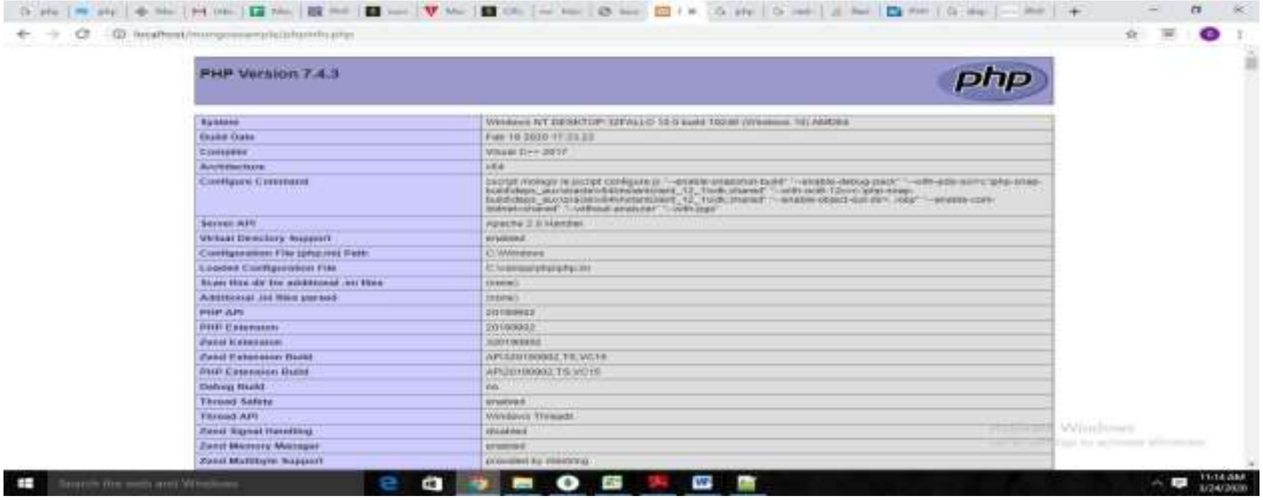
List of Hardware/Software requirements:

1. Laptop/Computer with Linux OS - Ubuntu 18.04 LTS/Windows
2. Apache 7.4 or above

Program:

```
<?php

phpinfo();

?>
```

Output/Results snippet:



localhost/mongoexample/phpinfo.php

PHP Version 7.4.3

System	Windows NT DESKTOP-[illegible] 10.0 build 10240 (Windows 10) AMD64
Build Date	Feb 18 2020 17:23:22
Compiler	Visual C++ 2017
Architecture	x64
Configure Command	cscript /nologo /e:jscript configure.js "--enable-snapshot-build" "--enable-debug-pack" "--with-pdo-oci=c:\php-snap-build\deps_aux\oracle\x64\instantclient_12_1\sdk,shared" "--with-oci8-12c=c:\php-snap-build\deps_aux\oracle\x64\instantclient_12_1\sdk,shared" "--enable-object-out-dir=../obj/" "--enable-com-dotnet=shared" "--without-analyzer" "--with-pgo"
Server API	Apache 2.0 Handler
Virtual Directory Support	enabled
Configuration File (php.ini) Path	C:\Windows
Loaded Configuration File	C:\xampp\php\php.ini
Scan this dir for additional .ini files	(none)
Additional .ini files parsed	(none)
PHP API	20190902
PHP Extension	20190902
Zend Extension	320190902
Zend Extension Build	API320190902,TS,VC15
PHP Extension Build	API20190902,TS,VC15
Debug Build	no
Thread Safety	enabled
Thread API	Windows Threads
Zend Signal Handling	disabled
Zend Memory Manager	enabled
Zend Multibyte Support	provided by mbstring

References:

- HTML Introduction - https://www.w3schools.com/
- PHP - https://www.w3schools.com/

Activity 4

Aim: Write a PHP script to display the strings (Notepad)

Learning outcome: Able to design and develop dynamic websites with PHP.

Duration: 1 hour

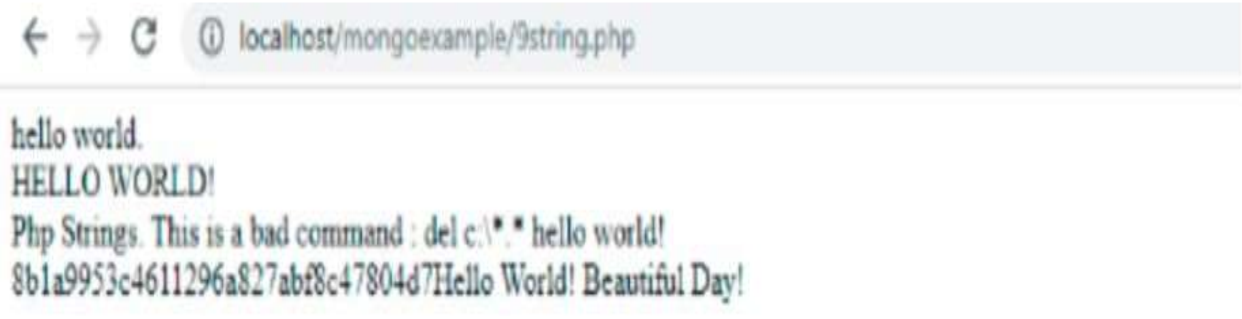
List of Hardware/Software requirements:

1. Laptop/Computer with Linux OS - Ubuntu 18.04 LTS/Windows
2. Apache 7.4 or above

Program:

```
<?php

echo strtolower("Hello WORLD.<br>");

echo strtoupper("Hello WORLD!<br>");

echo "Php Strings."."\n";

echo "This is a bad command : del c:\\*.*"."\n"; echo

lcfirst("Hello world!<br>");

$str = "Hello"; echo

md5($str);

$arr = array('Hello','World!','Beautiful','Day!');

echo join(" ",$arr);

?>
```

Output/Results snippet:



```
hello world.
HELLO WORLD!
Php Strings. This is a bad command : del c:\*.* hello world!
8b1a9953c4611296a827abf8c47804d7Hello World! Beautiful Day!
```

References:

- PHP - https://www.w3schools.com/
- https://www.tutorialrepublic.com/faq/how-to-create-a-new-line-in-php.php

Activity 5

Aim: Create a simple HTML form and accept the user name and display the name through PHP echo statement

Learning outcome: Able to design and develop dynamic websites with PHP.

Duration: 1 hour

List of Hardware/Software requirements:

1. Laptop/Computer with Linux OS - Ubuntu 18.04 LTS/Windows
2. Apache 7.4 or above

Program:

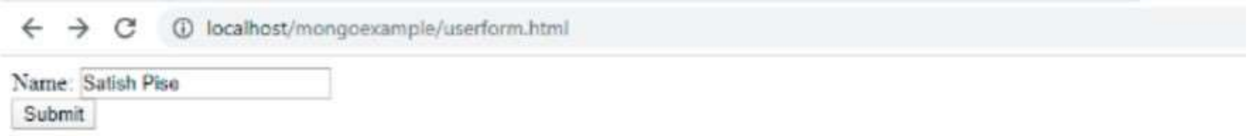
Userform.html

```
<html>
<body>
<form action="10formphp.php" method="post">
Name: <input type="text" name="name"><br>
<input type="submit">
</form> </body>
</html>
```

Formphp.php

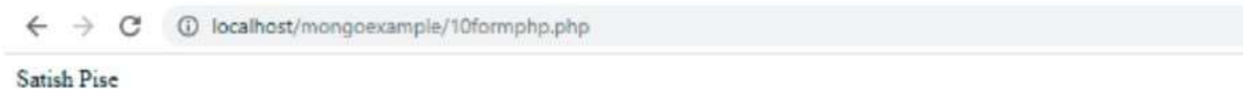
```
<?php
echo $_POST["name"];
?>
```

Output/Results snippet:



localhost/mongoexample/userform.html

Name: Satish Pise



localhost/mongoexample/10formphp.php

```
Satish Pise
```

References:

- PHP - https://www.w3schools.com/
- https://www.tutorialrepublic.com/faq/how-to-create-a-new-line-in-php.php

Activity 6

Aim: Write a PHP script to display string, values within a table

Learning outcome: Able to design and develop dynamic websites with PHP

Duration: 2 Hrs.

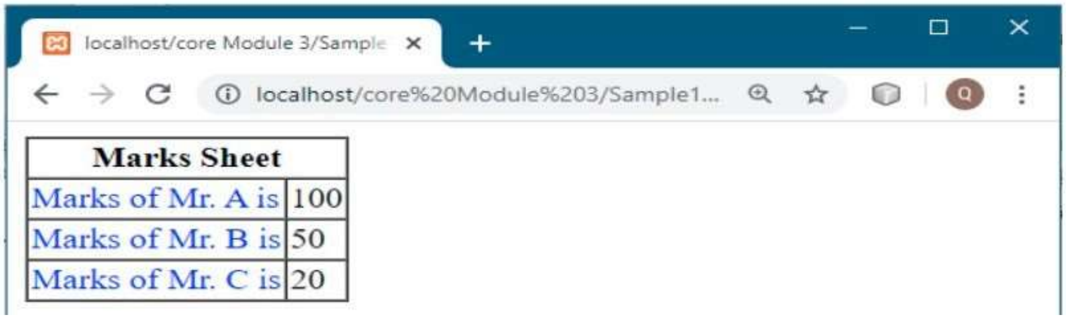
List of Hardware/Software requirements:

1. Laptop/Computer with Linux OS - Ubuntu 18.04 LTS/Windows
2. Notepad++
3. Xampp Server.

Program 1:

```
<?php
$a=100;
$b=50;
$c=20;
echo "<table border=1 cellspacing=0 cellpading=0>
<!— column which spans 2 -->
<tr> <th colspan= 2> Marks Sheet </th></tr>
<tr> <td><font color=blue>Marks of Mr. A is</td>
<td>$a</font></td></tr>
<tr> <td><font color=blue>Marks of Mr. B is</td>
<td>$b</font></td></tr>
<tr> <td><font color=blue>Marks of Mr. C is</td>
<td>$c</font></td></tr>
</table>";
?>
```

Output/Results snippet:

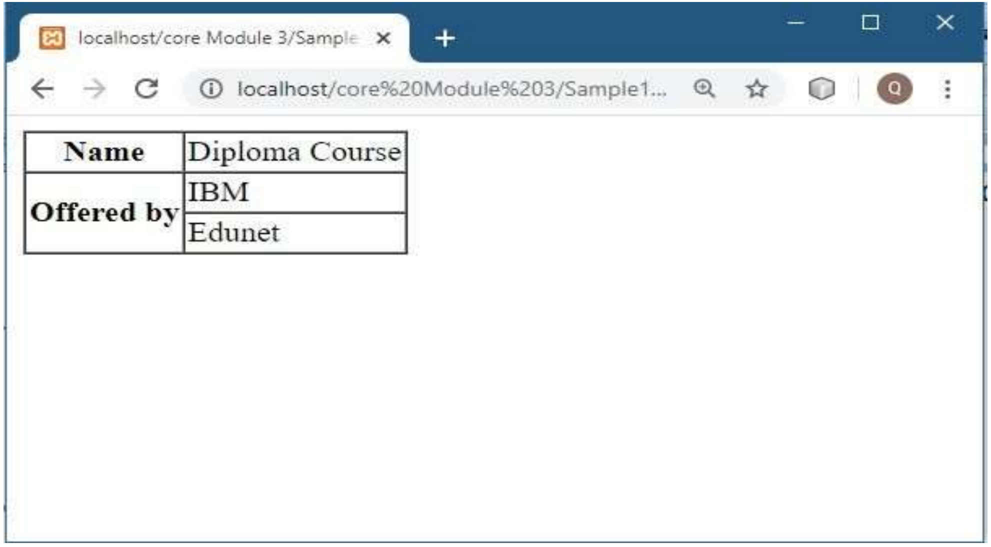

localhost/core Module 3/Sample
localhost/core%20Module%203/Sample1...

Marks Sheet	
Marks of Mr. A is	100
Marks of Mr. B is	50
Marks of Mr. C is	20

Program 2:

```
<?php
echo "<table border=1 cellspacing=0 cellpading=0>
 <tr>
    <th>Name</th>
    <td>Diploma Course</td>
  </tr>
  <tr>
<!— row which spans 2 -->
<th rowspan=2>Offered by</th>
    <td>IBM</td>
  </tr>
  <tr>
    <td>Edunet</td>
  </tr>";
?>
```

Output/Results snippet:



localhost/core Module 3/Sample
localhost/core%20Module%203/Sample1...
Q

Name	Diploma Course
Offered by	IBM
	Edunet

Reference:

- https://www.w3resource.com/php-exercises/php-basic-exercise-13.php

Activity 7

Aim: Write a PHP script to count lines in a file

Learning outcome: Able to design and develop dynamic websites with PHP

Duration: 30 mins

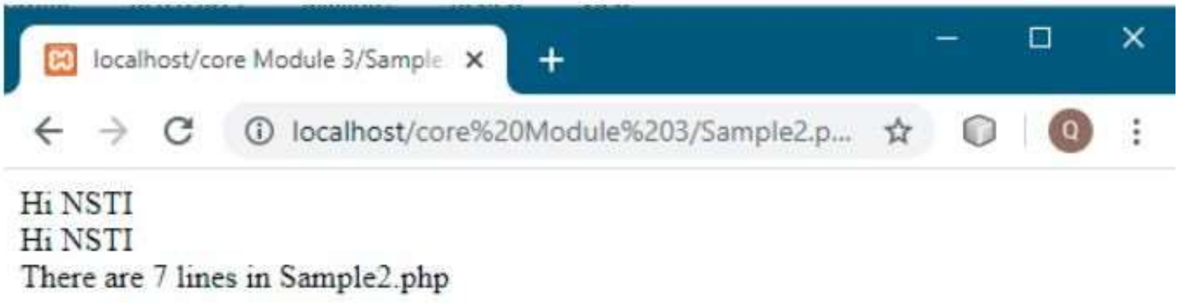
List of Hardware/Software requirements:

1. Laptop/Computer with Linux OS - Ubuntu 18.04 LTS/Windows
2. Notepad++
3. Xampp Server

Program:

```
<?php
// nl2br is used for newlines.
 echo nl2br("Hi NSTI \n");
echo nl2br("Hi NSTI \n");
$file = basename($_SERVER['PHP_SELF']);
$no_of_lines = count(file($file));
echo "There are $no_of_lines lines in $file"."\n";
?>
```

Output/Results snippet:


localhost/core Module 3/Sample
localhost/core%20Module%203/Sample2.p...

```
Hi NSTI
Hi NSTI
There are 7 lines in Sample2.php
```

Reference:

- https://www.w3resource.com/php-exercises/php-basic-exercise-16.php

Activity 8

Aim: Write a PHP function to test whether a number is greater than 30, 20 or 10 using ternary operator

Learning outcome: Able to design and develop dynamic websites with PHP

Duration: 1 Hrs.

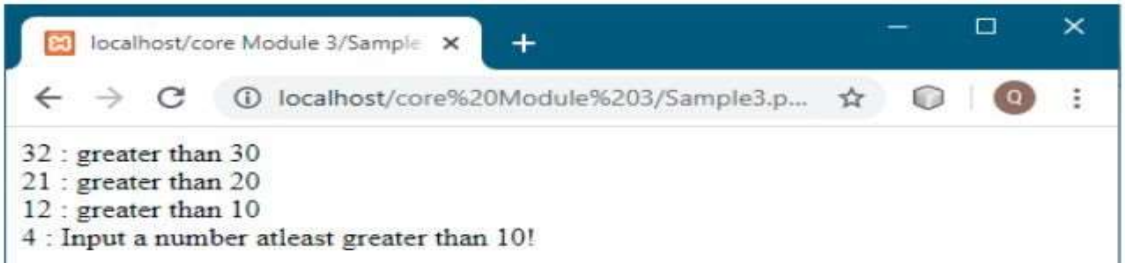
List of Hardware/Software requirements:

1. Laptop/Computer with Linux OS - Ubuntu 18.04 LTS/Windows
2. Notepad++
3. Xampp Server

Program 1:

```
<?php
// Declaring a function called Test
function trinary_Test($n){
$r = $n > 30
? nl2br("greater than 30 \n")
: ($n > 20
? nl2br("greater than 20 \n")
: ($n >10
? nl2br("greater than 10 \n")
: "Input a number atleast greater than 10!"));
echo $n." : ".$r."\n";
}
// Calling the function Test
trinary_Test(32);
 trinary_Test(21); trinary_Test(12);
 trinary_Test(4);
?>
```

Output/Results snippet:

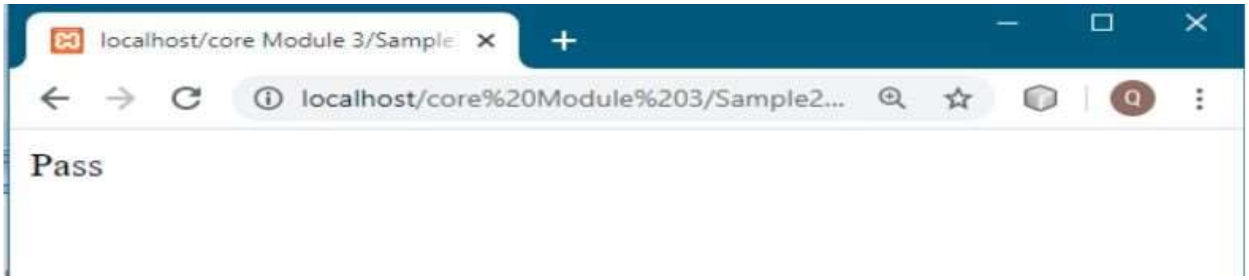

localhost/core Module 3/Sample
localhost/core%20Module%203/Sample3.p...

```
32 : greater than 30
21 : greater than 20
12 : greater than 10
4 : Input a number atleast greater than 10!
```

Program 2: Example of Ternary operators

```
<?php
$marks=50;
// using the print in php to display the output
print ($marks>=40) ? "Pass" : "Fail";
?>
```

Output/Results snippet:


localhost/core Module 3/Sample
localhost/core%20Module%203/Sample2...

```
Pass
```

Reference:

- https://www.w3resource.com/php-exercises/php-basic-exercise-21.php

Activity 9

Aim: Write a script which will display the string.

Learning outcome: Able to design and develop dynamic websites with PHP

Duration: 1 hour

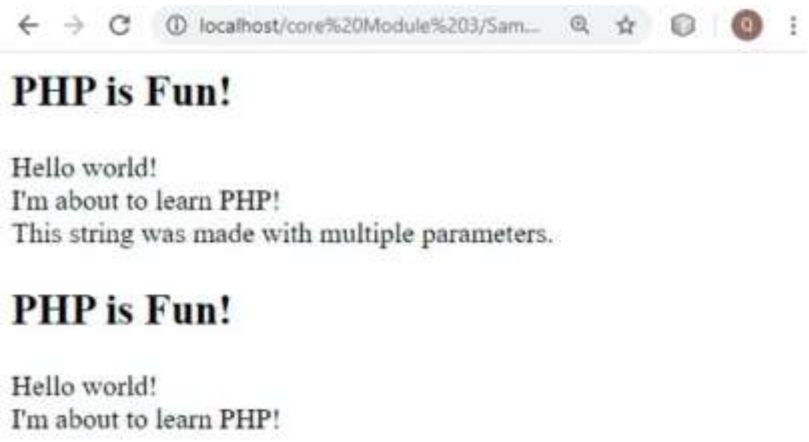
List of Hardware/Software requirements:

1. Laptop/Computer with Linux OS - Ubuntu 18.04 LTS/Windows
2. Notepad++
3. Xampp Server

Program:

```
<?php
    // using the echo in php to display the output echo
"<h2>PHP is Fun!</h2>";
echo "Hello world!<br>";
echo "I'm about to learn PHP!<br>";
echo "This ", "string ", "was ", "made ", "with multiple parameters.";
    // using the echo in php to display the output print
"<h2>PHP is Fun!</h2>";
print "Hello world!<br>";
print "I'm about to learn PHP!";
?>
```

Output/Results snippet:



PHP is Fun!

Hello world!
I'm about to learn PHP!
This string was made with multiple parameters.

PHP is Fun!

Hello world!
I'm about to learn PHP!

References:

- https://www.w3schools.com/
- https://www.w3resource.com/

Activity 10

Aim: Write a PHP script which will display the array.

Learning outcome: Able to design and develop dynamic websites with PHP

Duration: 2 hours

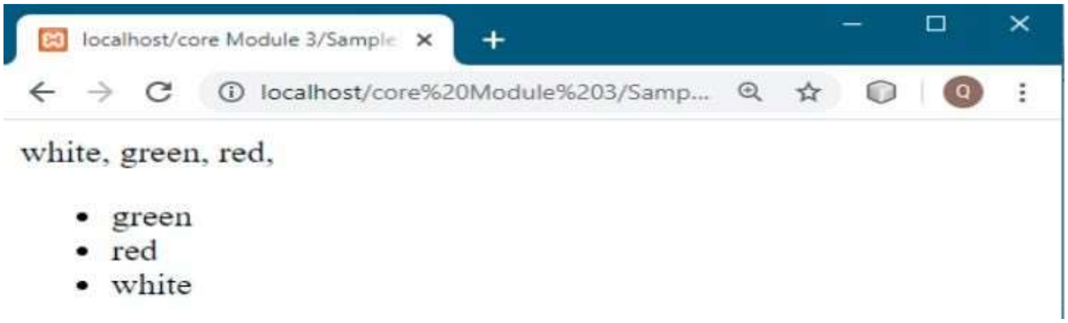
List of Hardware/Software requirements:

1. Notepad++
2. Xampp Server

Program:

```
<?php
// Declaring a array
$color = array('white', 'green', 'red');
// using the foreach loop to loop through in array
foreach ($color as $c)
{
echo "$c, "; }
sort($color);                              // Sorting the array
// Displaying the content of array in unordered list
echo "<ul>";
foreach ($color as $y)
{ echo "<li>$y</li>";
} echo "</ul>";
?>
```

Output/Results snippet:


localhost/core Module 3/Sample
localhost/core%20Module%203/Samp...

```
white, green, red,

  • green
  • red
  • white
```

Reference:

- https://www.w3resource.com

Activity 11

Aim: Write a PHP script to sorting

Learning outcome: Able to design and develop dynamic websites with PHP

Duration: 2 Hrs

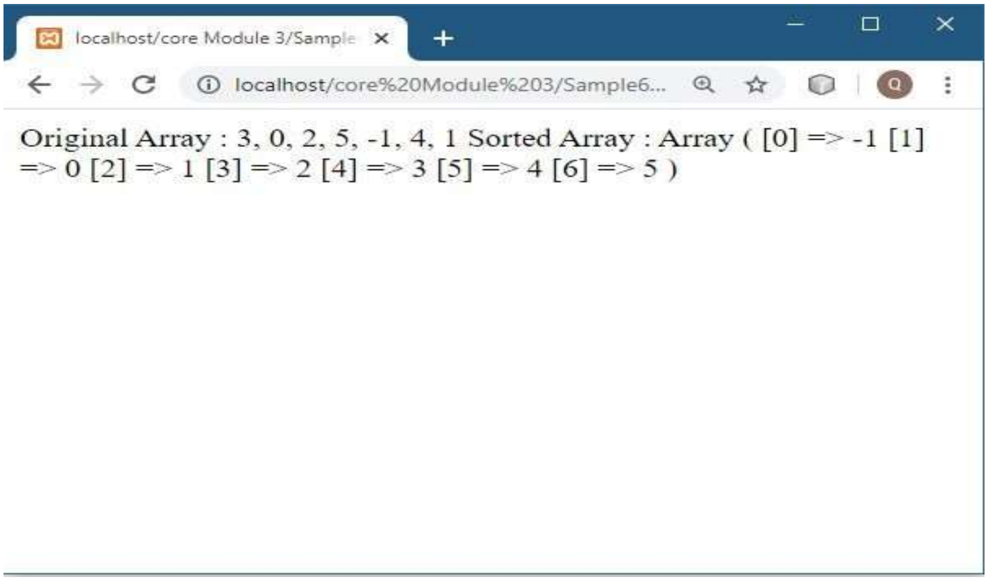
List of Hardware/Software requirements:

1. Notepad++
2. Xampp Server

Program:

```
<?php
// Writing the function to sort an array
// This function requires the parameter
function insertion_Sort($my_array)
{
for($i=0;$i<count($my_array);$i++){
$val = $my_array[$i];
$j = $i-1;
while($j>=0 && $my_array[$j] > $val){
$my_array[$j+1] = $my_array[$j];
$j--;
}
$my_array[$j+1] = $val;
}
return $my_array;
}
//Writing the array which need to be sort
$test_array = array(3, 0, 2, 5, -1, 4, 1);
echo "Original Array :\n";
echo implode(', ',$test_array );
echo "\nSorted Array :\n";
//Calling the function with parameter
//Displays the content of array in sorted manner
print_r(insertion_Sort($test_array));
?>
```

Output/Results snippet:



localhost/core Module 3/Sample
localhost/core%20Module%203/Sample6...

```
Original Array : 3, 0, 2, 5, -1, 4, 1 Sorted Array : Array ( [0] => -1 [1]
=> 0 [2] => 1 [3] => 2 [4] => 3 [5] => 4 [6] => 5 )
```

Reference:

- https://www.w3resource.com

Activity 12

Aim: Write a PHP script to calculate and display average temperature, five lowest and highest temperatures in given data

Learning outcome: Able to design and develop dynamic websites with PHP

Duration: 2 Hrs

List of Hardware/Software requirements:

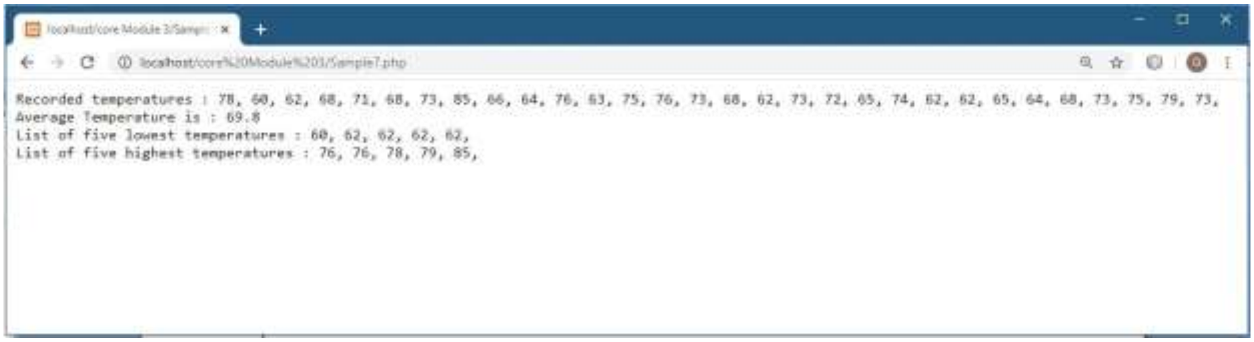
1. Notepad++
2. Xampp Server

Program:

```
<?php
echo "<pre>";
// Hard coded values for temperature
$temperatures = array(78, 60, 62, 68, 71, 68, 73, 85, 66, 64, 76, 63, 75, 76, 73, 68, 62, 73, 72, 65,
74,
62, 62, 65, 64, 68, 73, 75, 79, 73);
// Declaring a parameterized function
// Function name is listvalues
 function listvalues($value)
{
echo "$value, ";
}
// Function name is printAverage
function printAverage($array)
{
$total = 0;
foreach($array as $element)
{
$total += $element;
}
echo number_format($total / count($array), 1);
}
echo "Recorded temperatures : ";
array_walk($temperatures, "listvalues");
echo "<br>";
// Displaying the average of given temperatures echo
"Average Temperature is: ";
printAverage($temperatures);echo "<br>";
```

```
//sort the temperatures in ascending order for both of the following lists.
sort($temperatures);
//print the first 5 values
echo "List of five lowest temperatures :";
for($i = 0; $i < 5; $i++)
{
echo "$temperatures[$i], ";
}
echo "<br>";
//print the last 5 values
echo "List of five highest temperatures : ";
for($i = count($temperatures) - 5; $i <= count($temperatures) - 1; $i++)
{
echo "$temperatures[$i], ";
}
echo "<br>"; echo
"</pre>";
?>
```

Output/Results snippet:



localhost/core Module 3/Sampl ×
localhost/core%20Module%203/Sample7.php

```
Recorded temperatures : 78, 60, 62, 68, 71, 68, 73, 85, 66, 64, 76, 63, 75, 76, 73, 68, 62, 73, 72, 65, 74, 62, 62, 65, 64, 68, 73, 75, 79, 73,
Average Temperature is : 69.8
List of five lowest temperatures : 60, 62, 62, 62, 62,
List of five highest temperatures : 76, 76, 78, 79, 85,
```

Reference:

- https://www.w3resource.com

Activity 13

Aim: Write a program to calculate and print the factorial of a number using a for loop.

Learning outcome: Able to design and develop dynamic websites with PHP

Duration: 2 Hrs

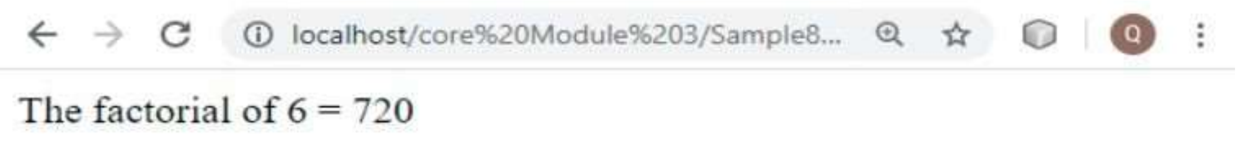
List of Hardware/Software requirements:

1. Notepad++
2. Xampp Server

Program:

```
<?php
// Declaring a variable whose factorial is to be find
$n = 6;
$x = 1;
//Calculating the factorial of a given number
for($i=1;$i<=$n-1;$i++)
{
$x*=($i+1);
}
// Displaying the factorial of a given number echo
"The factorial of $n = $x"."\n";
?>
```

Output/Results snippet:



```
The factorial of 6 = 720
```

Reference:

- https://www.w3resource.com

Activity 14

Aim: Write a PHP script using nested for loop

Learning outcome: Able to design and develop dynamic websites with PHP

Duration: 1 Hrs.

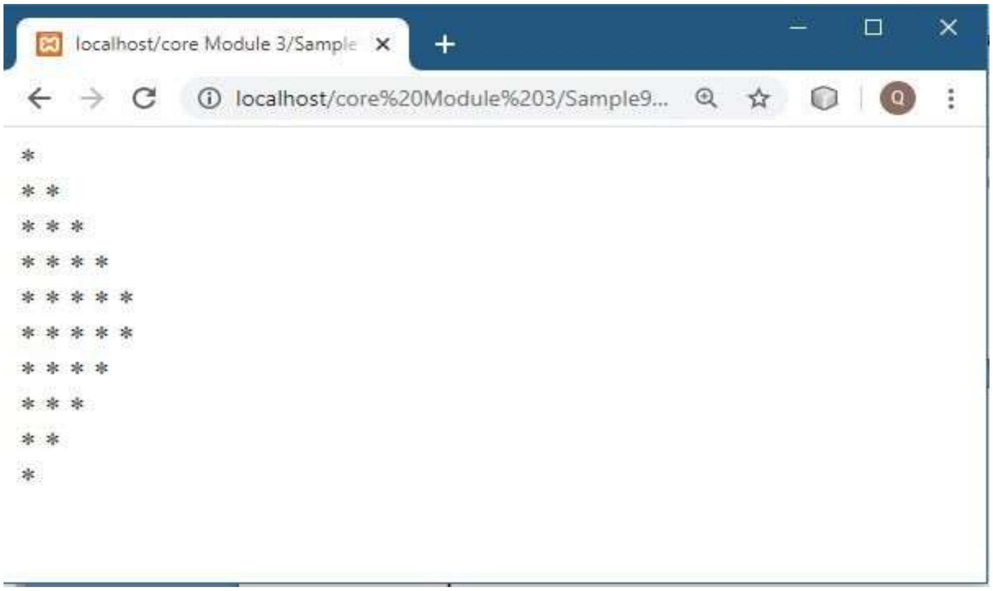
List of Hardware/Software requirements:

1. Notepad++
2. Xampp Server

Program:

```
<?php
$n=5;
// A for loop
for($i=1; $i<=$n; $i++)
{
//Declaring a for loop inside in the another for loop
// A nester for loop
for($j=1; $j<=$i; $j++)
{
echo ' * ';
}
echo nl2br("\n");
}
// A for loop for($i=$n;
$i>=1; $i--)
{
//Declaring a for loop inside in the another for loop
// A nester for loop
for($j=1; $j<=$i; $j++)
{
echo ' * ';
}
echo nl2br("\n ");
}
?>
```

Output/Results snippet:



localhost/core Module 3/Sample
localhost/core%20Module%203/Sample9...

```
*
* *
* * *
* * * *
* * * * *
* * * * *
* * * *
* * *
* *
*
```

Reference:

- https://www.w3resource.com

Activity 15

Aim: Write a PHP program to generate and display the first n lines of a Floyd

Learning outcome: Able to design and develop dynamic websites with PHP

Duration: 2 Hrs

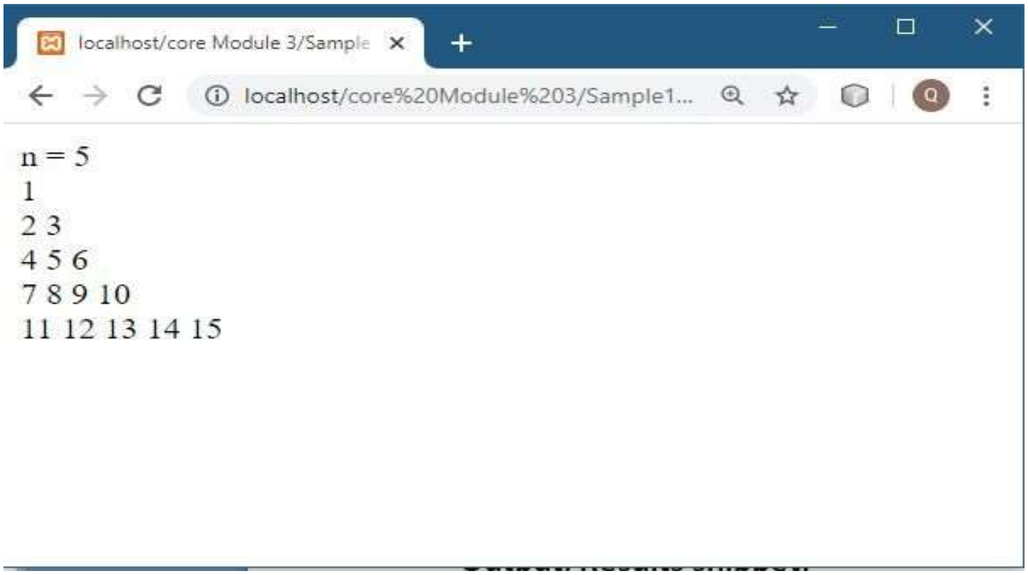
List of Hardware/Software requirements:

1. Notepad++
2. Xampp Server

Program:

```
<?php
// Declaring number of lines we need in a triangle
$n = 5;
echo "n = " . $n . "<br>";
//Specifying from where we need to print
$count = 1;
for ($i = $n; $i > 0; $i--)
{
for ($j = $i; $j < $n + 1; $j++)
{
printf("%4s", $count);
$count++;
}
echo "<br>";
}
?>
```

Output/Results snippet:


localhost/core Module 3/Sample
localhost/core%20Module%203/Sample1...

```
n = 5
1
2 3
4 5 6
7 8 9 10
11 12 13 14 15
```

References:

- https://www.w3resource.com/php-exercises/php-for-loop-exercise-12.php
- https://www.w3resource.com/php-exercises/
- https://www.w3schools.com/php/default.asp
- https://www.javatpoint.com/php-tutorial

Activity 16

Aim: Write a function to calculate the factorial of a number

Learning outcome: Able to design and develop dynamic websites with PHP.

Duration: 2 hours

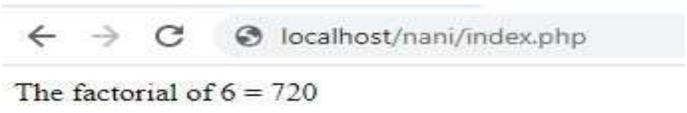
List of Hardware/Software requirements:

1. Laptop/Computer with Linux OS - Ubuntu 18.04 LTS/Windows
2. Lamp Server with PHP

Program:

```
<?php
// PHP code to get the factorial of a number
// function to get factorial in iterative way
function Factorial($number){
   if($number <= 1){
      return 1;
   }
   else{
      return $number * Factorial($number - 1);
   }
}
// Driver Code
$number = 10;
$fact = Factorial($number);
echo "Factorial = $fact";
?>
```

Output/Results snippet:



```
The factorial of 6 = 720
```

References:

- https://www.javatpoint.com/php-factorial-program

Activity 17

Aim: Write a function to check a number is prime or not

Learning outcome: Able to design and develop dynamic websites with PHP.

Duration: 1 hour

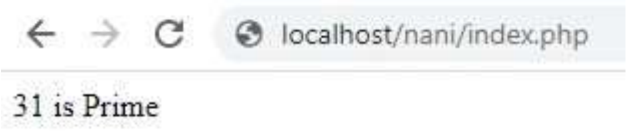
List of Hardware/Software requirements:

1. Laptop/Computer with Linux OS - Ubuntu 18.04 LTS/Windows
2. LAMP Server with PHP

Program:

```
<?php
function primeCheck($number){ //creating a function
 primeCheck
 if ($number == 1)
 return 0;
 for($i=2;$i<=$number/2;
 $i++){ if ($number % $i
 == 0)
 return 0; }
 return 1;
}
$number = 31;                              //input number 31
$flag = primeCheck($number);          //checking
prime or not
 if ($flag == 1)
echo "$number
is Prime"; else
echo "$number is Not Prime"
?>
```

Output/Results snippet:



31 is Prime

References:

- https://www.w3resource.com/php-exercises/php-function-exercise-2.php

Activity 18

Aim: Write a function to reverse a string

Learning outcome: Able to design and develop dynamic websites with PHP.

Duration: 1 hour

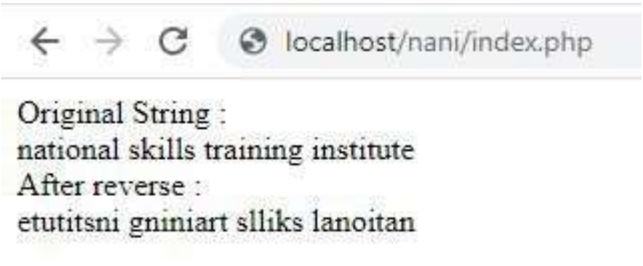
List of Hardware/Software requirements:

1. Laptop/Computer with Linux OS - Ubuntu 18.04 LTS
2. LAMP Server with PHP

Program:

```
          <?php
function Reverse($str){     //creating a function Reverse and passing $str as
 parameterreturn strrev($str);           //used strrev built-in function to get reverse
 of string
}
          $str = "national skills training institute";        //input for $str as a string
          echo "Original String :<br>";                   // Display original
          stringecho $str."<br>";

          echo "After reverse : <br>";         //Display reverse of
          stringecho Reverse($str);
?>
```

Output/Results snippet:



```
Original String :
national skills training institute
After reverse :
etutitsni gniniart slliks lanoitan
```

References:

- https://www.geeksforgeeks.org/php-reverse-string/

Activity 19

Aim: Write a PHP function that checks whether a passed string is a palindrome or not?

Learning outcome: Able to design and develop dynamic websites with PHP.

Duration: 1 hour

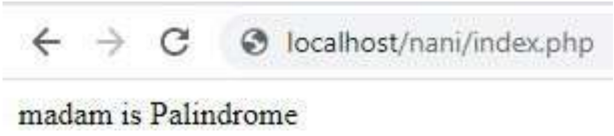
List of Hardware/Software requirements:

1. Laptop/Computer with Linux OS - Ubuntu 18.04 LTS
2. LAMP Server with PHP

Program:

```
        <?php
        function check_palindrome($string) //creating a function check_palindrome
{
 if ($string == strrev($string)) //comparing string with reverse of same
 stringreturn 1;              //return 1 if reverse and original string is
 same else

 return 0;                    //return 0 if reverse and original strings are not same
}
        $string="madam"; //taking example string as
        madam if(check_palindrome($string)==1)
                                  //checking palindrome or
        notecho "$string is Palindrome";     //display the
        output

        else
        echo "$string is Not a palindrome";
?>
```

Output/Results snippet:



localhost/nani/index.php

```
madam is Palindrome
```

References:

- https://www.geeksforgeeks.org/php-palindrome-check/

Activity 20

Aim: Write a simple PHP class which displays the given string

Learning outcome: Able to design and develop dynamic websites with PHP.

Duration: 1 hour

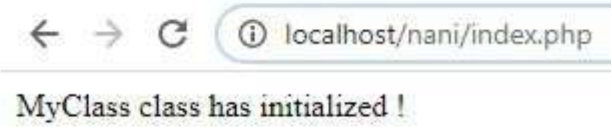
List of Hardware/Software requirements:

1. Laptop/Computer with Linux OS - Ubuntu 18.04 LTS
2. LAMP Server with PHP

Program:

```
           <?php
class MyClass { //creating a class  ‘MyClass’ class name
 public function __construct() //creating a constructor for
 class
 {
 echo 'MyClass class has initialized !'."\n";
 }
}
           $userclass = new MyClass; //creating object for class
?>
```

Output/Results snippet:



```
MyClass class has initialized !
```

References:

- https://www.w3resource.com/php-exercises/php-class-exercise-1.php

Activity 21

Aim: Write a PHP Calculator class which will accept two values as arguments, then add them, subtract them, multiply them together, or divide them on request

Learning outcome: Able to design and develop dynamic websites with PHP.

Duration: 3 hours

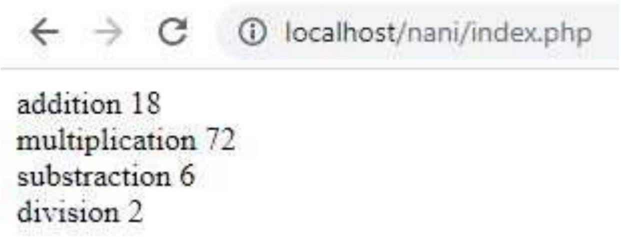
List of Hardware/Software requirements:

1. Laptop/Computer with Linux OS - Ubuntu 18.04 LTS
2. LAMP Server with PHP

Program:

```
        <?php
        class MyCalculator { //class name MyCalculator
        private $_fval, $_sval; //Data members declared as private
        public function___construct( $fval, $sval ) { //passing parameters to the function
        $this->_fval = $fval; //initializing the data members
        $this->_sval = $sval; //initializing the data members
}
        public function add() { // creating a add
        function return $this->_fval + $this-
        >_sval; //adding the values
}
        public function subtract() { //creating the subtract
        function return $this->_fval - $this->_sval;
        //subtraction is done here
}
        public function multiply() { //creating the multiply
        function return $this->_fval * $this->_sval;
        //multiplication is done here
}
        public function divide() { //creating the divide
        function return $this->_fval / $this->_sval;
        //division is done here
}}
        $mycalc = new MyCalculator(12, 6); //passing the parameter values to the
        function 12,6 echo "addition ",$mycalc-> add()."<br>"; // Displays addition
        12+6=18

        echo "multiplication ",$mycalc-> multiply()."<br>"; // Displays multiplication
        12*6=72echo "substraction ",$mycalc-> subtract()."<br>"; // Displays
        subtraction 12-6=6

        echo "division ",$mycalc-> divide(); // Displays division 12/6=2
?>
```

Output/Results snippet:



```
addition 18
multiplication 72
substraction 6
division 2
```

References:

1. https://www.w3resource.com/php-exercises/php-class-exercise-6.php

Activity 22

Aim: Write a PHP script to: - a) transform a string all uppercase letters. b) transform a string all lowercase letters. c) make a string's first character uppercase. d) make a string's first character of all the words uppercase.

Learning outcome: Able to design and develop dynamic websites with PHP.

Duration: 2 hours

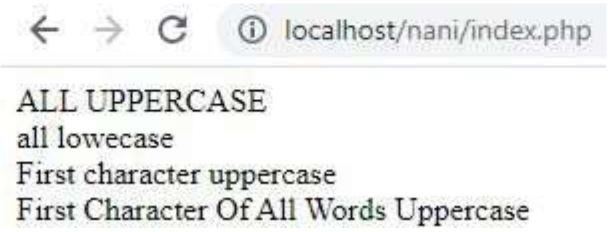
List of Hardware/Software requirements:

1. Laptop/Computer with Linux OS - Ubuntu 18.04 LTS
2. Lamp Server with PHP

Program:

```
        <?php
        //strtoupper is in-built function used for to make uppercase
        lettersprint(strtoupper("all uppercase<br>"))."\n";

        //strtolower is in-built function used for to make all into lower letters
        print(strtolower("all lowecase<br>"))."\n";

        // ucfirst is in-built function used for to make first character in a statement as Uppercase
        print(ucfirst("first character uppercase<br>"))."\n";

// ucwords is in-built function used for to make first character of every word in a sentence as
    Uppercase

        print(ucwords("first character of all words uppercase"))."\n";
?>
```

Output/Results snippet:



```
ALL UPPERCASE
all lowecase
First character uppercase
First Character Of All Words Uppercase
```

References:

1. https://www.w3resource.com/php-exercises/php-string-exercise-1.php

Activity 23

Aim: Create a form in PHP and apply validations.

Learning outcome: Able to design and develop dynamic websites with PHP.

Duration: 2 hours

List of Hardware/Software requirements:

1. Laptop/Computer with Linux OS - Ubuntu 18.04 LTS
2. Lamp Server with PHP

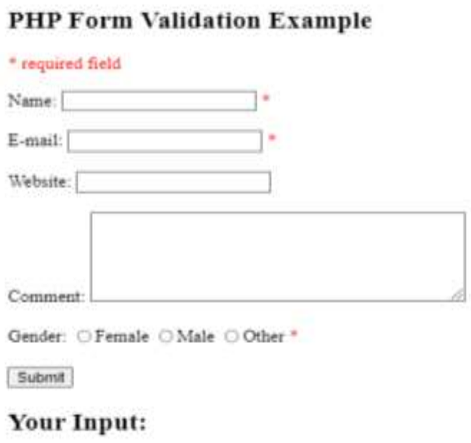
Program:

```
<!DOCTYPE HTML>
<html>
<head>
</head>
<body>

<?php
// define variables and set to empty values
$name = $email = $gender = $comment = $website = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
  $name = test_input($_POST["name"]);
  $email = test_input($_POST["email"]);
  $website = test_input($_POST["website"]);
  $comment = test_input($_POST["comment"]);
  $gender = test_input($_POST["gender"]);
}
```

```
function test_input($data) {
  $data = trim($data);
  $data = stripslashes($data);
  $data = htmlspecialchars($data);
  return $data;
}
?>

<h2>PHP Form Validation Example</h2>
<form method="post" action="<?php echo
htmlspecialchars($_SERVER["PHP_SELF"]);?>">
  Name: <input type="text" name="name">
  <br><br>
  E-mail: <input type="text" name="email">
  <br><br>

Website: <input type="text" name="website">
  <br><br>
  Comment: <textarea name="comment" rows="5" cols="40"></textarea>
  <br><br>
  Gender:
  <input type="radio" name="gender" value="female">Female
  <input type="radio" name="gender" value="male">Male
  <input type="radio" name="gender" value="other">Other
  <br><br>
  <input type="submit" name="submit" value="Submit">
</form>
```

```
<?php
echo "<h2>Your Input:</h2>";
echo $name;
echo "<br>";
echo $email;
echo "<br>";
echo $website;
echo "<br>";
echo $comment;
echo "<br>";
echo $gender;
?>

</body>
</html>
```

Output/Results snippet:



PHP Form Validation Example

* required field

Name: *

E-mail: *

Website:

Comment:

Gender: ○ Female ○ Male ○ Other *

Your Input:

References:

1. https://www.w3schools.com/php/php_form_validation.asp

Activity 24

Aim: Create a date and time from a number of parameters in mktime().

Learning outcome: Able to design and develop dynamic websites with PHP.

Duration: 1 hour

List of Hardware/Software requirements:

1. Laptop/Computer with Linux OS - Ubuntu 18.04 LTS
2. Lamp Server with PHP

Program:

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

<?php
// Prints: October 3, 1975 was on a Friday
echo "Oct 3, 1975 was on a ".date("l", mktime(0,0,0,10,3,1975)) . "<br><br>";

//The mktime() function is useful for doing date arithmetic and validation.
//It will automatically calculate the correct value for out-of-range input:
echo date("M-d-Y",mktime(0,0,0,12,36,2001)) . "<br>";
echo date("M-d-Y",mktime(0,0,0,14,1,2001)) . "<br>";
echo date("M-d-Y",mktime(0,0,0,1,1,2001)) . "<br>";
echo date("M-d-Y",mktime(0,0,0,1,1,99)) . "<br>";
?>

</body>
```

</html>

Output/Results snippet:

```
Oct 3, 1975 was on a Friday


Jan-05-2002

Feb-01-2002

Jan-01-2001

Jan-01-1999
```

References:

1. https://www.w3schools.com/php/func_date_mktime.asp

Activity 25

Aim: Create a date and time from the strtotime() function

Learning outcome: Able to design and develop dynamic websites with PHP.

Duration: 1 hour

List of Hardware/Software requirements:

1. Laptop/Computer with Linux OS - Ubuntu 18.04 LTS
2. Lamp Server with PHP

Program:

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

<?php
echo(strtotime("now") . "<br>");
echo(strtotime("3 October 2005") . "<br>");
echo(strtotime("+5 hours") . "<br>");
echo(strtotime("+1 week") . "<br>");
echo(strtotime("+1 week 3 days 7 hours 5 seconds") . "<br>");
echo(strtotime("next Monday") . "<br>");
echo(strtotime("last Sunday"));
?>

</body>
</html>
```

Output/Results snippet:

```
1589972726

1128297600

1589990726

1590577526

1590861931

1590364800

1589673600
```

References:

1. https://www.w3schools.com/php/func_date_strtotime.asp

Activity 26

Aim: Output the dates for the next six Saturdays

Learning outcome: Able to design and develop dynamic websites with PHP.

Duration: 1 hour

List of Hardware/Software requirements:

1. Laptop/Computer with Linux OS - Ubuntu 18.04 LTS
2. Lamp Server with PHP

Program:

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

<?php
$startdate=strtotime("Saturday");
$enddate=strtotime("+6 weeks", $startdate);

while ($startdate < $enddate) {
  echo date("M d", $startdate) . "<br>";
  $startdate = strtotime("+1 week", $startdate);
}
?>

</body>
</html>
```

Output/Results snippet:

```
Mar 12
Mar 19
Mar 26
Apr 02
Apr 09
Apr 16
```

References:

1. https://www.w3schools.com/php/php_date.asp

Activity 27

Aim: Create and retrieve a cookie

Learning outcome: Able to design and develop dynamic websites with PHP.

Duration: 1 hour

List of Hardware/Software requirements:

1. Laptop/Computer with Linux OS - Ubuntu 18.04 LTS
2. Lamp Server with PHP

Program:

```
<!DOCTYPE html>
<?php
$cookie_name = "user";
$cookie_value = "John Doe";
setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/"); // 86400 = 1 day
?>
<html>
<body>

<?php
if(!isset($_COOKIE[$cookie_name])) {
    echo "Cookie named '" . $cookie_name . "' is not set!";
} else {
    echo "Cookie '" . $cookie_name . "' is set!<br>";
    echo "Value is: " . $_COOKIE[$cookie_name];
}
?>
```

```
<p><strong>Note:</strong> You might have to reload the page to see the value of the cookie.</p>


</body>

</html>
```

Output/Results snippet:

```
Cookie named 'user' is not set!

Note: You might have to reload the page to see the value of the cookie.
```

References:

1. https://www.w3schools.com/php/php_cookies.asp

Activity 28

Aim: Modify a cookie value

Learning outcome: Able to design and develop dynamic websites with PHP.

Duration: 1 hour

List of Hardware/Software requirements:

1. Laptop/Computer with Linux OS - Ubuntu 18.04 LTS
2. Lamp Server with PHP

Program:

```
<!DOCTYPE html>
<?php
$cookie_name = "user";
$cookie_value = "Alex Porter";
setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/");
?>
<html>
<body>

<?php
if(!isset($_COOKIE[$cookie_name])) {
    echo "Cookie named '" . $cookie_name . "' is not set!";
} else {
    echo "Cookie '" . $cookie_name . "' is set!<br>";
    echo "Value is: " . $_COOKIE[$cookie_name];
}
?>
```

```
<p><strong>Note:</strong> You might have to reload the page to see the new value of the cookie.</p>


</body>

</html>
```

Output/Results snippet:

```
Cookie named 'user' is not set!
Note: You might have to reload the page to see the new value of the cookie.
```

References:

1. https://www.w3schools.com/php/php_cookies.asp

Activity 29

Aim: Delete a cookie

Learning outcome: Able to design and develop dynamic websites with PHP.

Duration: 1 hour

List of Hardware/Software requirements:

1. Laptop/Computer with Linux OS - Ubuntu 18.04 LTS
2. Lamp Server with PHP

Program:

```
<!DOCTYPE html>
<?php
// set the expiration date to one hour ago
setcookie("user", "", time() - 3600);
?>
<html>
<body>

<?php
echo "Cookie 'user' is deleted.";
?>

</body>
</html>
```

Output/Results snippet:

```
Cookie 'user' is deleted.
```

References:

1. https://www.w3schools.com/php/php_cookies.asp

Activity 30

Aim: Check if cookies are enabled

Learning outcome: Able to design and develop dynamic websites with PHP.

Duration: 1 hour

List of Hardware/Software requirements:

1. Laptop/Computer with Linux OS - Ubuntu 18.04 LTS
2. Lamp Server with PHP

Program:

```
<!DOCTYPE html>
<?php
setcookie("test_cookie", "test", time() + 3600, '/');
?>
<html>
<body>

<?php
if(count($_COOKIE) > 0) {
    echo "Cookies are enabled.";
} else {
    echo "Cookies are disabled.";
}
?>

</body>
</html>
```

Output/Results snippet:

```
Cookies are enabled.
```

References:

1. https://www.w3schools.com/php/php_cookies.asp

Activity 31

Aim: Select data with MySQLi (Object-oriented)

Learning outcome: Able to design and develop dynamic websites with PHP.

Duration: 2 hours

List of Hardware/Software requirements:

1. Laptop/Computer with Linux OS - Ubuntu 18.04 LTS
2. Lamp Server with PHP

Program:

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

<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

$sql = "SELECT id, firstname, lastname FROM MyGuests";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
    // output data of each row
    while($row = $result->fetch_assoc()) {
        echo "<br> id: ". $row["id"]. " - Name: ". $row["firstname"]. " " . $row["lastname"] .
"<br>";
    }
} else {
    echo "0 results";
}

$conn->close();
?>

</body>
</html>
```

Output/Results snippet:

```
id: 1 - Name: John Doe

id: 2 - Name: Mary Moe

id: 3 - Name: Julie Dooley
```

References:

1. https://www.w3schools.com/php/php_mysql_select.asp

Activity 32

Aim: Select data with MySQLi (Object-oriented) and put result in an HTML table

Learning outcome: Able to design and develop dynamic websites with PHP.

Duration: 2 hours

List of Hardware/Software requirements:

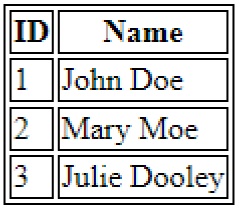
1. Laptop/Computer with Linux OS - Ubuntu 18.04 LTS
2. Lamp Server with PHP

Program:

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

<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

$sql = "SELECT id, firstname, lastname FROM MyGuests";
$result = $conn->query($sql);
```

```
if ($result->num_rows > 0) {
    echo "<table><tr><th>ID</th><th>Name</th></tr>";
    // output data of each row
    while($row = $result->fetch_assoc()) {
        echo "<tr><td>" . $row["id"]. "</td><td>" . $row["firstname"]. " " . $row["lastname"]. "</td></tr>";
    }
    echo "</table>";
} else {
    echo "0 results";
}

$conn->close();
?>

</body>
</html>
```

Output/Results snippet:



ID	Name
1	John Doe
2	Mary Moe
3	Julie Dooley

References:

1. https://www.w3schools.com/php/php_mysql_select.asp

Activity 33

Aim: Select data with MySQLi (Procedural)

Learning outcome: Able to design and develop dynamic websites with PHP.

Duration: 1 hours

List of Hardware/Software requirements:

1. Laptop/Computer with Linux OS - Ubuntu 18.04 LTS
2. Lamp Server with PHP

Program:

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

<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = mysqli_connect($servername, $username, $password, $dbname);
// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}

$sql = "SELECT id, firstname, lastname FROM MyGuests";
```

```
    $result = mysqli_query($conn, $sql);

    if (mysqli_num_rows($result) > 0) {
        // output data of each row
        while($row = mysqli_fetch_assoc($result)) {
            echo "id: " . $row["id"]. " - Name: " . $row["firstname"]. " " . $row["lastname"]. "<br>";
        }
    } else {
        echo "0 results";
    }

    mysqli_close($conn);
    ?>

    </body>
</html>
```

Output/Results snippet:

```
id: 1 - Name: John Doe
id: 2 - Name: Mary Moe
id: 3 - Name: Julie Dooley
```

References:

1. https://www.w3schools.com/php/php_mysql_select.asp

Activity 34

Aim: Select data with PDO (+Prepared statements)

Learning outcome: Able to design and develop dynamic websites with PHP.

Duration: 3 hours

List of Hardware/Software requirements:

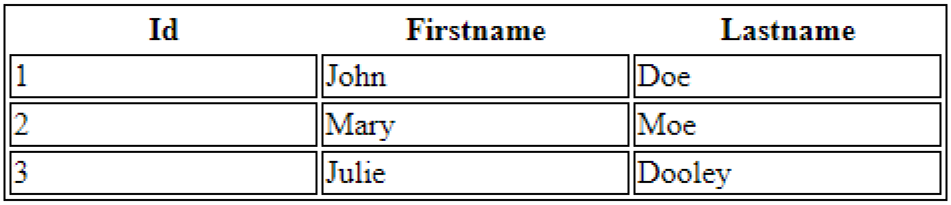
1. Laptop/Computer with Linux OS - Ubuntu 18.04 LTS
2. Lamp Server with PHP

Program:

```
<!DOCTYPE html>

<html>

<body>


<?php

echo "<table style='border: solid 1px black;'>";

 echo "<tr><th>Id</th><th>Firstname</th><th>Lastname</th></tr>";


class TableRows extends RecursiveIteratorIterator {

    function __construct($it) {

        parent::__construct($it, self::LEAVES_ONLY);

    }


    function current() {

        return "<td style='width: 150px; border: 1px solid black;'>" .
parent::current(). "</td>";

    }


    function beginChildren() {

        echo "<tr>";

    }


    function endChildren() {

        echo "</tr>" . "\n";
    }
}

$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDBPDO";

try {
    $conn = new PDO("mysql:host=$servername;dbname=$dbname",
$username, $password);
    $conn->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);
    $stmt = $conn->prepare("SELECT id, firstname, lastname FROM
MyGuests");
    $stmt->execute();

    // set the resulting array to associative
    $result = $stmt->setFetchMode(PDO::FETCH_ASSOC);

    foreach(new TableRows(new RecursiveArrayIterator($stmt->fetchAll())) as
$k=>$v) {
        echo $v;
    }
}
catch(PDOException $e) {
    echo "Error: " . $e->getMessage();
}
```

```
        $conn = null;
        echo "</table>";
        ?>

        </body>
</html>
```

Output/Results snippet:



Id	Firstname	Lastname
1	John	Doe
2	Mary	Moe
3	Julie	Dooley

References:

1. https://www.w3schools.com/php/php_mysql_select.asp

Learning Outcome

After completing this module, the student should be **able to understand Laravel Framework.**

To meet the learning outcome, a student has to complete the following activities

1. Installation of the Laravel in the System
2. First Program by edit the View as Hello World
3. Create a Web application of minimum 4 blades demonstrating the blade template in laravel
4. Create a web Application which demonstrate routing through controller.
5. Create a Form which inserts the data in the database (7 hrs)
6. Demonstrate all the CRUD operation using (DB) from controllers (7 hrs)
7. Create a Model and Demonstrate all CRUD operations (8 hrs)
8. Creating a complete end-to–end solution which demonstrate session and database connectivity (8 hrs)
9. Demonstrate csv file upload in Laravel, extract its data and insert into database (8 Hrs.)
10. Installation and usage of jet brains Live wire and its component (8 Hrs.)
11. Create a login and register Forms using Live wire (9 Hrs.)

Activity 1

Aim: Installation of the Laravel in the System.

Learning outcome: Able to understand Laravel Framework.

Duration: 5 hours

List of Hardware/Software requirements:

1. Laptop/Computer with Windows 10 / Linux OS - Ubuntu 18.04 LTS
2. Php , XAMMP or LAMP Server, composer, Laravel 8

Code/Program/Procedure (with comments):

Install Laravel framework 8 for windows 10

Install composer on Windows, Follow below given steps to install composer on windows 10:

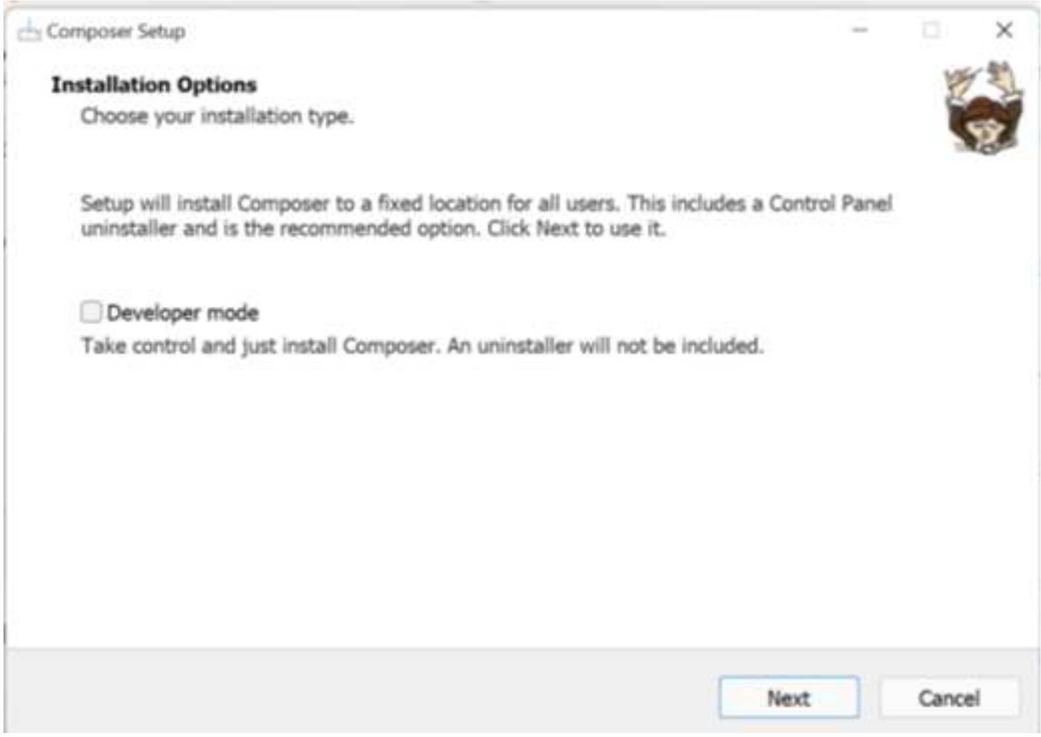
Step 1: Download Composer .exe File

In step 1, you need to open your browser and type getcomposer. Then visit on www.getcomposer.org. And download composer. If you already composer download / installed Composer. So, you need to move on next step.

Click Download Composer from www.getcomposer.org.

Step 2 – Run the setup and Install Composer

In step 2, click on download composer.exe file. Then open one prompt box :



Composer Setup

Installation Options

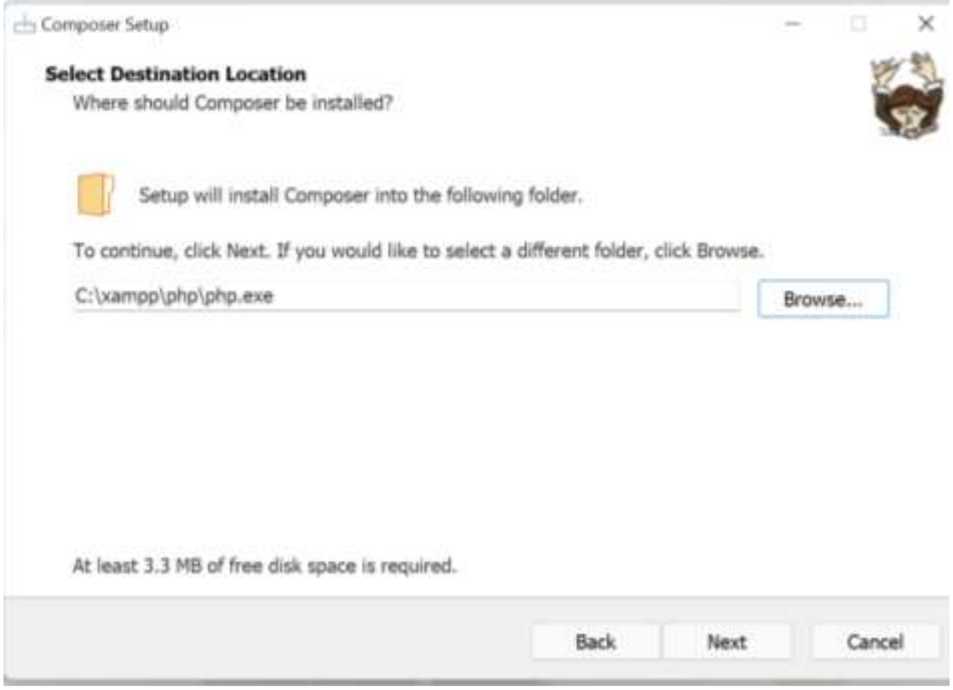
Choose your installation type.

Setup will install Composer to a fixed location for all users. This includes a Control Panel uninstaller and is the recommended option. Click Next to use it.

□ Developer mode

Take control and just install Composer. An uninstaller will not be included.

Now Set PHP path and click on next:



Composer Setup

Select Destination Location

Where should Composer be installed?

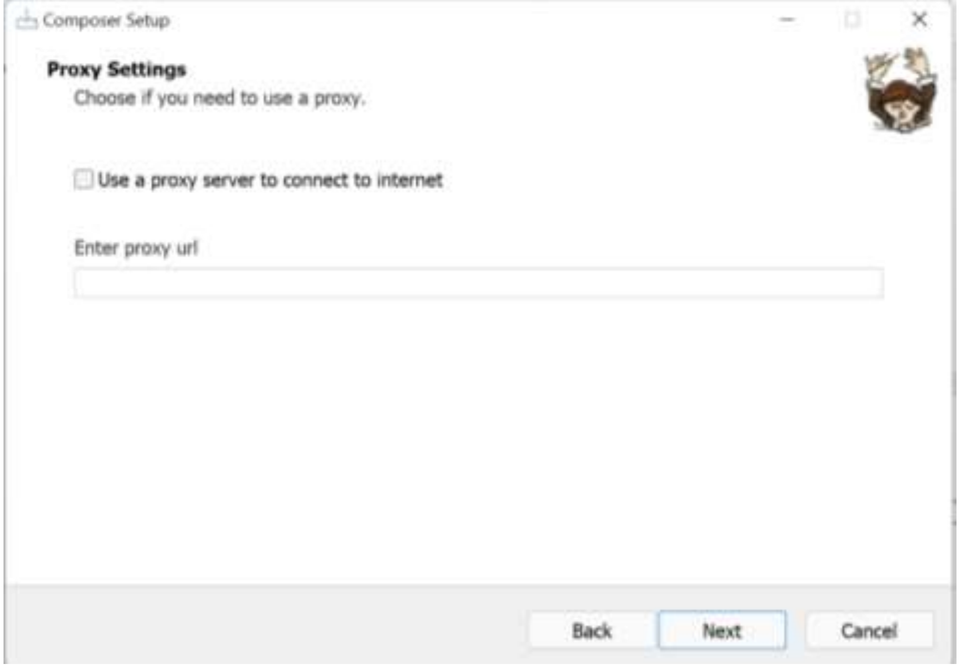
Setup will install Composer into the following folder.

To continue, click Next. If you would like to select a different folder, click Browse.

C:\xampp\php\php.exe

At least 3.3 MB of free disk space is required.

If you have any proxy URL enter here, don't know to leave it to click next:



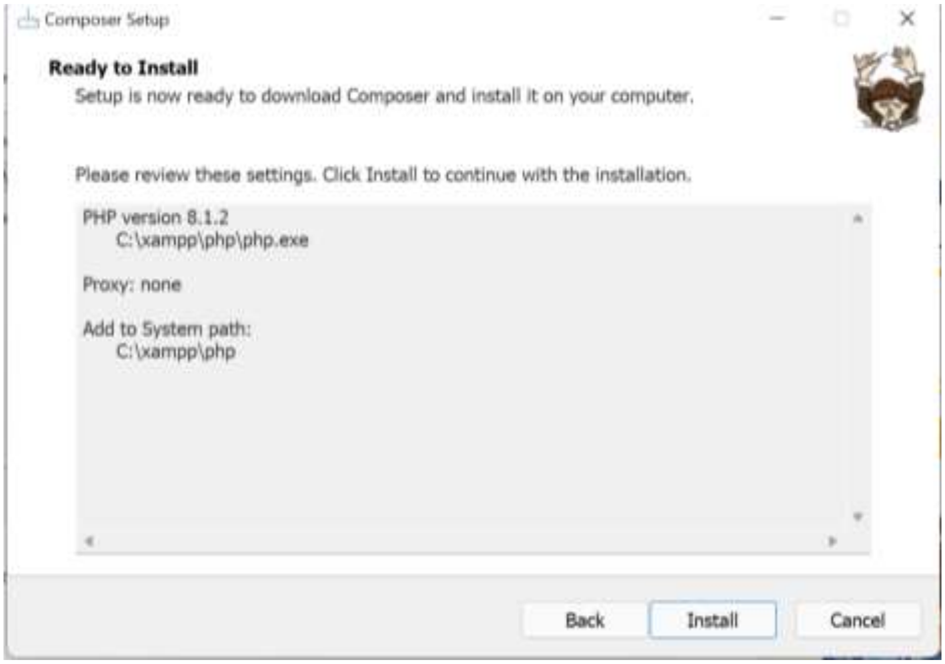
Proxy Settings

Choose if you need to use a proxy.

□ Use a proxy server to connect to internet

Enter proxy url

Now, review the composer setup wizard. And click on install.



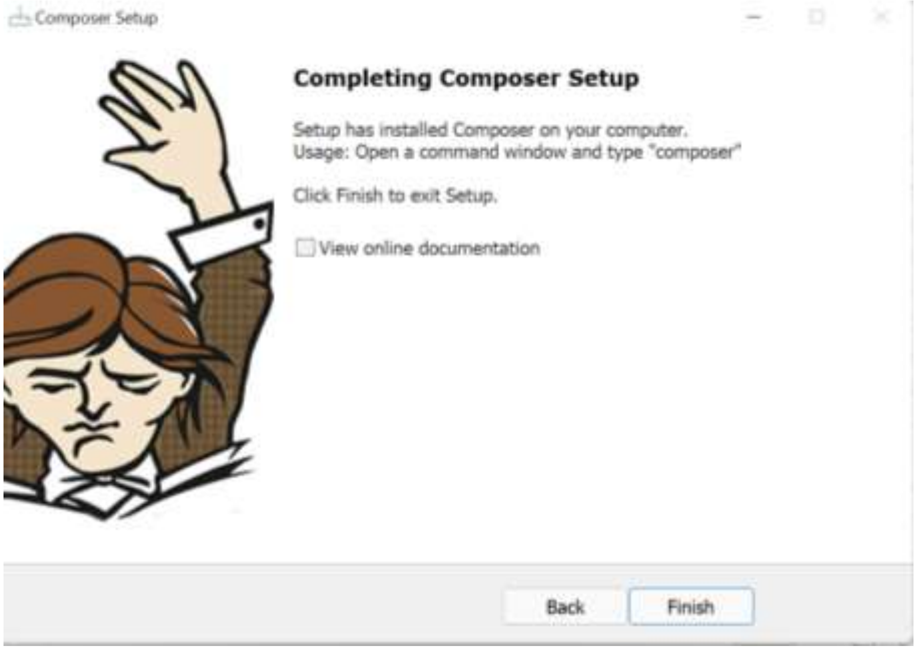
Ready to Install

Setup is now ready to download Composer and install it on your computer.

Please review these settings. Click Install to continue with the installation.

```
PHP version 8.1.2
    C:\xampp\php\php.exe

Proxy: none

Add to System path:
    C:\xampp\php
```

Composer has been installed successfully. Click to finish button.



Completing Composer Setup

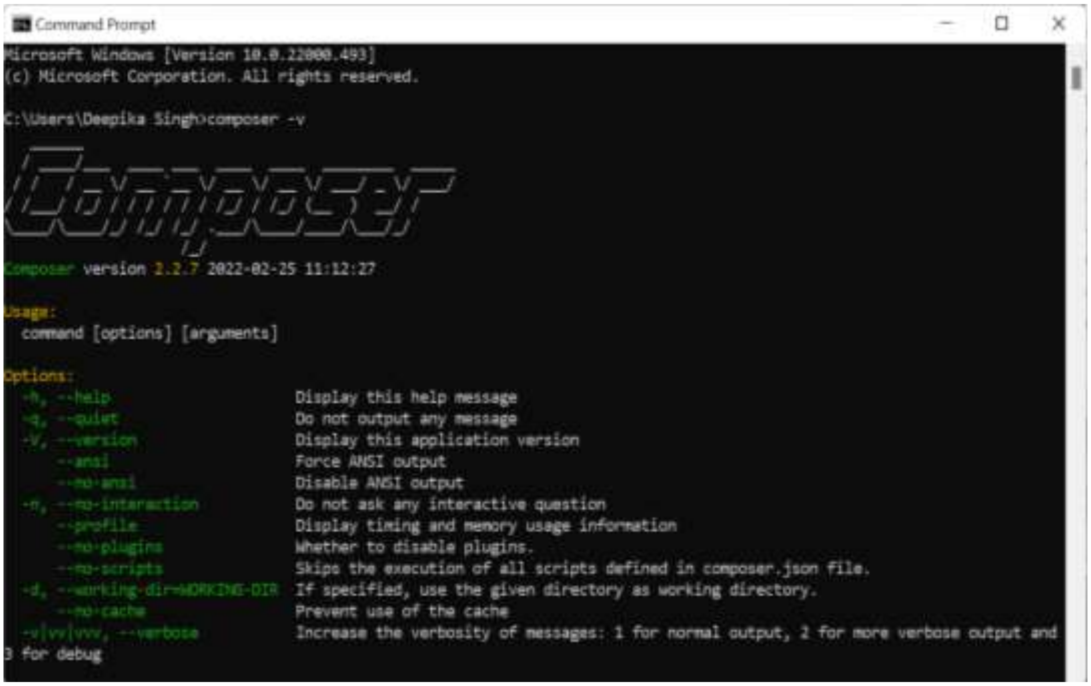
Setup has installed Composer on your computer.
Usage: Open a command window and type "composer"

Click Finish to exit Setup.

□ View online documentation

Step 3 – How to check composer is installed or not in windows

Now, open your terminal and type the following command on command prompt:



```
icrosoft Windows [Version 10.0.22000.493]
(c) Microsoft Corporation. All rights reserved.

:\Users\Deepika Singh>composer -v
   ______
  / ____/___  ____ ___  ____  ____  ________  _____
 / /   / __ \/ __ `__ \/ __ \/ __ \/ ___/ _ \/ ___/
/ /___/ /_/ / / / / / / /_/ / /_/ (__  )  __/ /
\____/\____/_/ /_/ /_/ .___/\____/____/\___/_/
                    /_/
omposer version 2.2.7 2022-02-25 11:12:27

sage:
  command [options] [arguments]

ptions:
  -h, --help                     Display this help message
  -q, --quiet                    Do not output any message
  -V, --version                  Display this application version
      --ansi                     Force ANSI output
      --no-ansi                  Disable ANSI output
  -n, --no-interaction           Do not ask any interactive question
      --profile                  Display timing and memory usage information
      --no-plugins               Whether to disable plugins.
      --no-scripts               Skips the execution of all scripts defined in composer.json file.
  -d, --working-dir=WORKING-DIR  If specified, use the given directory as working directory.
      --no-cache                 Prevent use of the cache
  -v|vv|vvv, --verbose           Increase the verbosity of messages: 1 for normal output, 2 for more verbose output and
3 for debug
```

After run the above command on cmd. If You will look like the image given below. So composer successfully installed on your windows 10 xampp.

Check the server requirement for the setup:

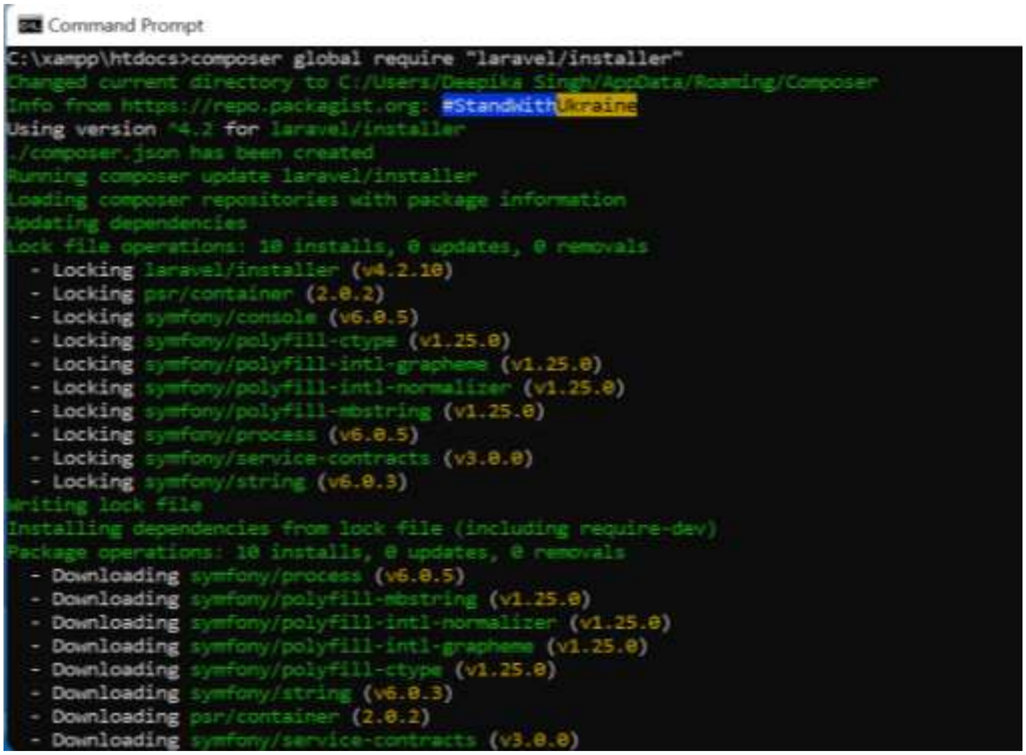
- PHP >= 7.3
- BCMath PHP Extension
- Ctype PHP Extension
- Fileinfo PHP extension
- JSON PHP Extension
- Mbstring PHP Extension
- OpenSSL PHP Extension
- PDO PHP Extension
- Tokenizer PHP Extension
- XML PHP Extension.

Installing Laravel:

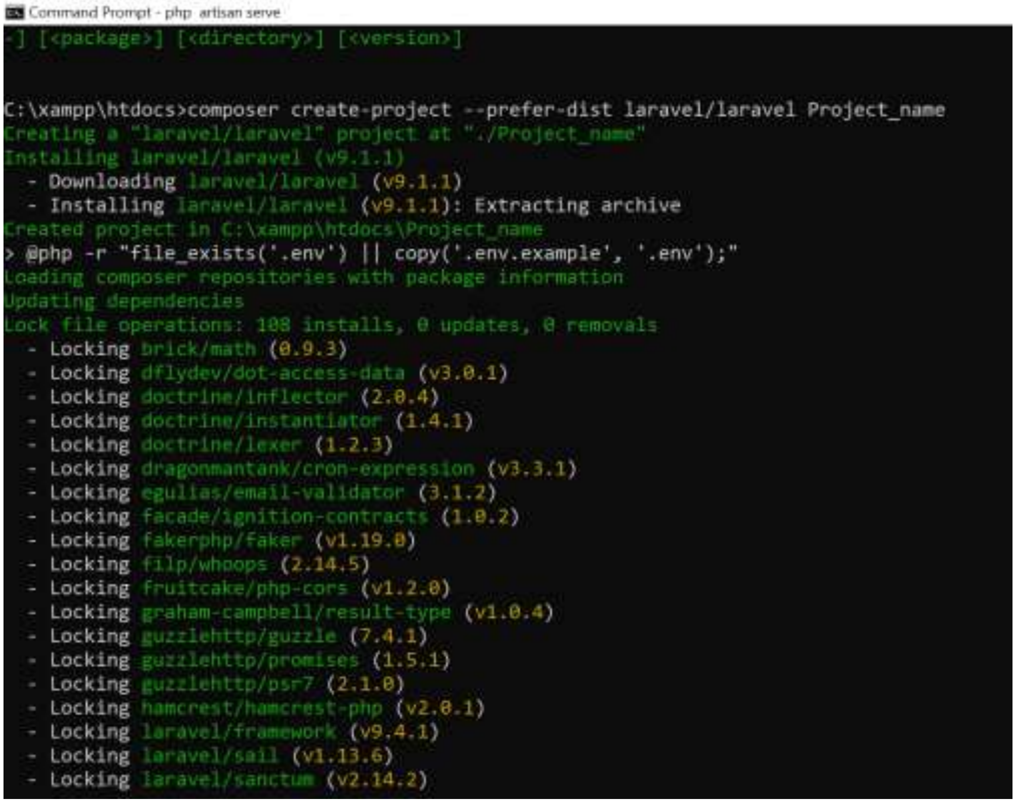
Type the following command in your command prompt window:

- composer global require “laravel/installer”.

- composer create-project –prefer-dist laravel/laravel Project_name: this command will install Laravel and other dependencies with it also generate the ANSI key.



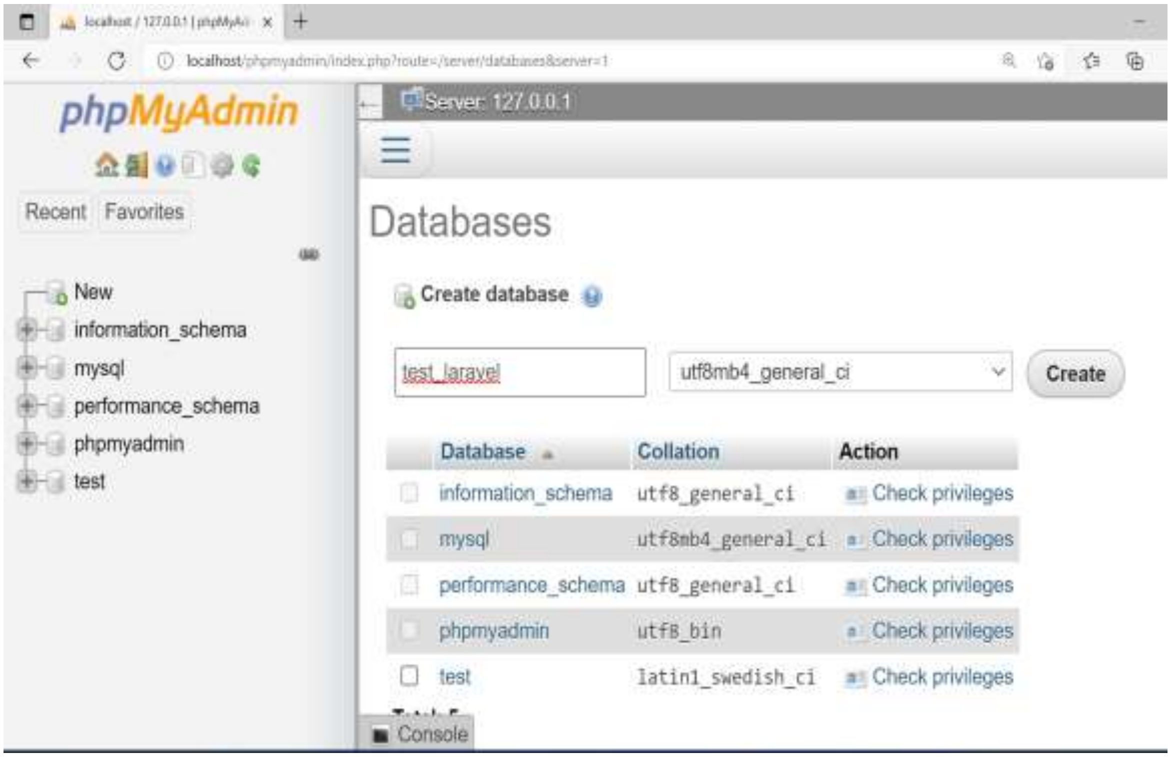
```
C:\xampp\htdocs>composer global require "laravel/installer"
Changed current directory to C:/Users/Deepika Singh/AppData/Roaming/Composer
Info from https://repo.packagist.org: #StandWithUkraine
Using version ^4.2 for laravel/installer
./composer.json has been created
Running composer update laravel/installer
Loading composer repositories with package information
Updating dependencies
Lock file operations: 10 installs, 0 updates, 0 removals
  - Locking laravel/installer (v4.2.10)
  - Locking psr/container (2.0.2)
  - Locking symfony/console (v6.0.5)
  - Locking symfony/polyfill-ctype (v1.25.0)
  - Locking symfony/polyfill-intl-grapheme (v1.25.0)
  - Locking symfony/polyfill-intl-normalizer (v1.25.0)
  - Locking symfony/polyfill-mbstring (v1.25.0)
  - Locking symfony/process (v6.0.5)
  - Locking symfony/service-contracts (v3.0.0)
  - Locking symfony/string (v6.0.3)
Writing lock file
Installing dependencies from lock file (including require-dev)
Package operations: 10 installs, 0 updates, 0 removals
  - Downloading symfony/process (v6.0.5)
  - Downloading symfony/polyfill-mbstring (v1.25.0)
  - Downloading symfony/polyfill-intl-normalizer (v1.25.0)
  - Downloading symfony/polyfill-intl-grapheme (v1.25.0)
  - Downloading symfony/polyfill-ctype (v1.25.0)
  - Downloading symfony/string (v6.0.3)
  - Downloading psr/container (2.0.2)
  - Downloading symfony/service-contracts (v3.0.0)
```



```
-] [<package>] [<directory>] [<version>]

C:\xampp\htdocs>composer create-project --prefer-dist laravel/laravel Project_name
Creating a "laravel/laravel" project at "./Project_name"
Installing laravel/laravel (v9.1.1)
  - Downloading laravel/laravel (v9.1.1)
  - Installing laravel/laravel (v9.1.1): Extracting archive
Created project in C:\xampp\htdocs\Project_name
> @php -r "file_exists('.env') || copy('.env.example', '.env');"
Loading composer repositories with package information
Updating dependencies
Lock file operations: 108 installs, 0 updates, 0 removals
  - Locking brick/math (0.9.3)
  - Locking dflydev/dot-access-data (v3.0.1)
  - Locking doctrine/inflector (2.0.4)
  - Locking doctrine/instantiator (1.4.1)
  - Locking doctrine/lexer (1.2.3)
  - Locking dragonmantank/cron-expression (v3.3.1)
  - Locking egulias/email-validator (3.1.2)
  - Locking facade/ignition-contracts (1.0.2)
  - Locking fakerphp/faker (v1.19.0)
  - Locking filp/whoops (2.14.5)
  - Locking fruitcake/php-cors (v1.2.0)
  - Locking graham-campbell/result-type (v1.0.4)
  - Locking guzzlehttp/guzzle (7.4.1)
  - Locking guzzlehttp/promises (1.5.1)
  - Locking guzzlehttp/psr7 (2.1.0)
  - Locking hamcrest/hamcrest-php (v2.0.1)
  - Locking laravel/framework (v9.4.1)
  - Locking laravel/sail (v1.13.6)
  - Locking laravel/sanctum (v2.14.2)
```

Create Database for Project

- Go to phpMyAdmin click on create a new tab.
- Name the database.
- Press create button.



Databases

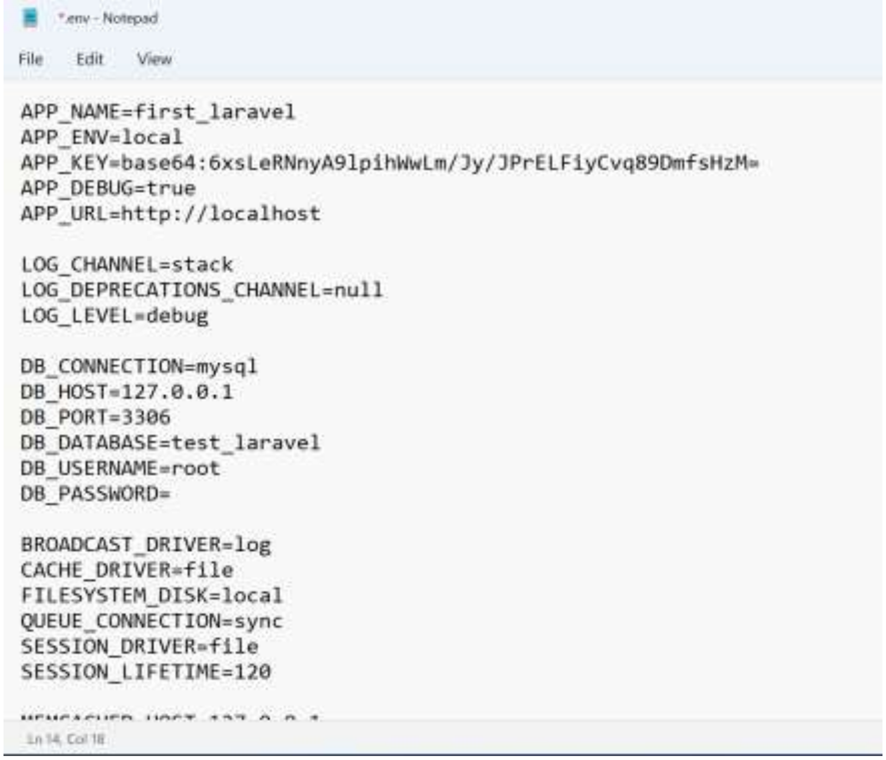
Create database

test_laravel | utf8mb4_general_ci | Create

	Database	Collation	Action
□	information_schema	utf8_general_ci	Check privileges
□	mysql	utf8mb4_general_ci	Check privileges
□	performance_schema	utf8_general_ci	Check privileges
□	phpmyadmin	utf8_bin	Check privileges
□	test	latin1_swedish_ci	Check privileges

Update .Env file:

- APP_NAME=first_laravel
- APP_ENV=local
- APP_KEY=base64:TJ9Sob7KFPhL5XkqT+TyQux3x7UbW08QLb0xtirLWSs=
- APP_DEBUG=true
- APP_URL=http://127.0.0.1:8000
- LOG_CHANNEL=stack
- LOG_LEVEL=debug
- DB_CONNECTION=mysql
- DB_HOST=127.0.0.1
- DB_PORT=3306
- DB_DATABASE=test_laravel
- DB_USERNAME=root
- DB_PASSWORD=



```
APP_NAME=first_laravel
APP_ENV=local
APP_KEY=base64:6xsLeRNnyA9lpihWwLm/Jy/JPrELFiyCvq89DmfsHzM=
APP_DEBUG=true
APP_URL=http://localhost

LOG_CHANNEL=stack
LOG_DEPRECATIONS_CHANNEL=null
LOG_LEVEL=debug

DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=test_laravel
DB_USERNAME=root
DB_PASSWORD=

BROADCAST_DRIVER=log
CACHE_DRIVER=file
FILESYSTEM_DISK=local
QUEUE_CONNECTION=sync
SESSION_DRIVER=file
SESSION_LIFETIME=120

MEMCACHED_HOST=127.0.0.1
```

Migrate database:

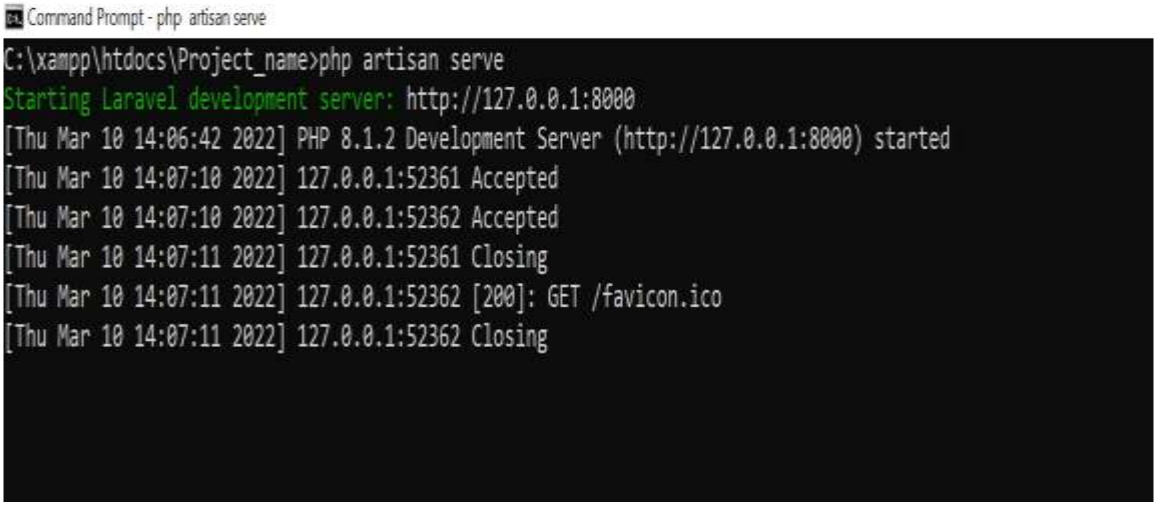
Create tables in the database for Laravel access, also helps in database version control following are the commands for migration.

PHP artisan make migration create_databse_table: this command is used to create the DB migration file in your ‘database/migration’ folder.

PHP artisan migrate: used to run the pending migration changes to the database

Start development server:

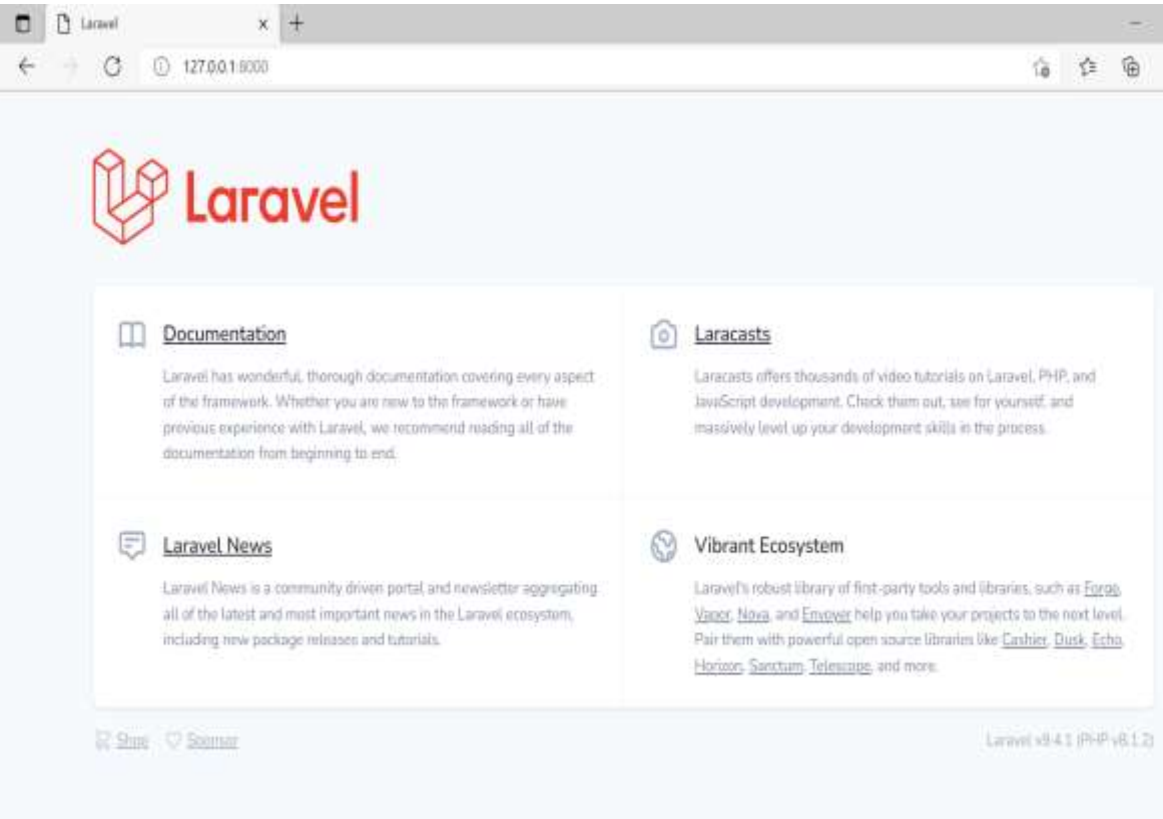
PHP artisan serve: this command starts your development server.



```
C:\xampp\htdocs\Project_name>php artisan serve
Starting Laravel development server: http://127.0.0.1:8000
[Thu Mar 10 14:06:42 2022] PHP 8.1.2 Development Server (http://127.0.0.1:8000) started
[Thu Mar 10 14:07:10 2022] 127.0.0.1:52361 Accepted
[Thu Mar 10 14:07:10 2022] 127.0.0.1:52362 Accepted
[Thu Mar 10 14:07:11 2022] 127.0.0.1:52361 Closing
[Thu Mar 10 14:07:11 2022] 127.0.0.1:52362 [200]: GET /favicon.ico
[Thu Mar 10 14:07:11 2022] 127.0.0.1:52362 Closing
```

Output/Results snippet:

Go to the IP URL that you see on your CMD screen.



Laravel

Documentation

Laravel has wonderful, thorough documentation covering every aspect of the framework. Whether you are new to the framework or have previous experience with Laravel, we recommend reading all of the documentation from beginning to end.

Laracasts

Laracasts offers thousands of video tutorials on Laravel, PHP, and JavaScript development. Check them out, see for yourself, and massively level up your development skills in the process.

Laravel News

Laravel News is a community driven portal and newsletter aggregating all of the latest and most important news in the Laravel ecosystem, including new package releases and tutorials.

Vibrant Ecosystem

Laravel's robust library of first-party tools and libraries, such as Forge, Vapor, Nova, and Envoyer help you take your projects to the next level. Pair them with powerful open source libraries like Cashier, Dusk, Echo, Horizon, Sanctum, Telescope, and more.

Laravel v9.4.1 (PHP v8.1.2)

Activity 2

Aim: First program by edit the view as Hello World.

Learning outcome: Able to understand Laravel Framework.

Duration: 5 hour

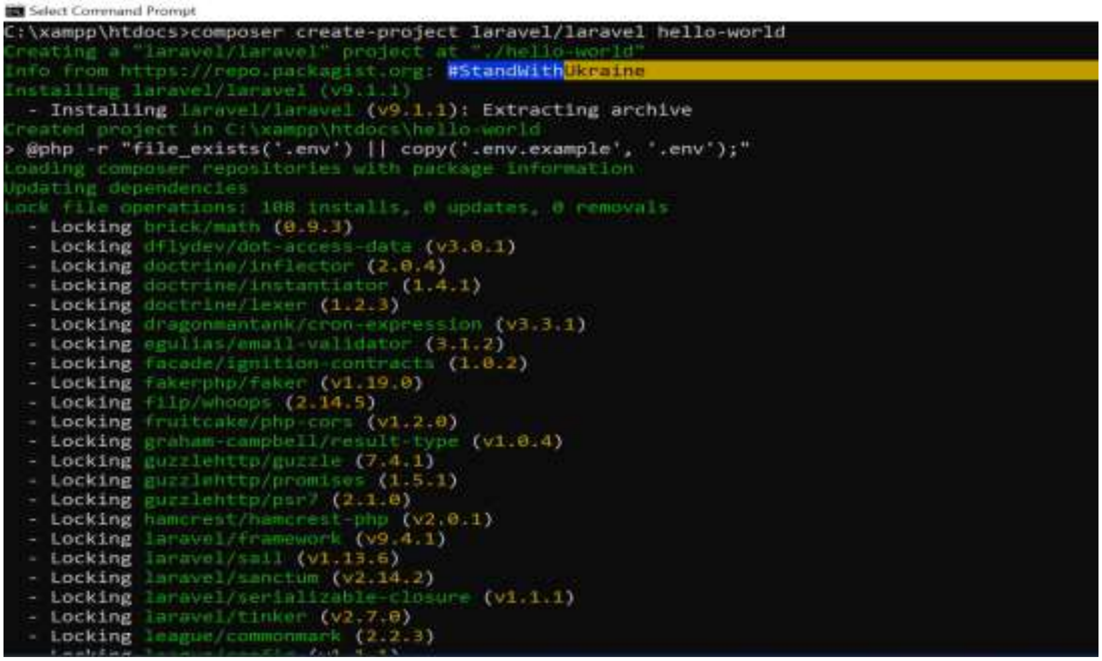
List of Hardware/Software requirements:

1. Laptop/Computer with Windows 10 / Linux OS - Ubuntu 18.04 LTS
2. Php , XAMMP or LAMP Server, composer, Laravel 8

Code/Program/Procedure (with comments):

Create a Laravel application:

```
composer create-project laravel/laravel hello-world
```



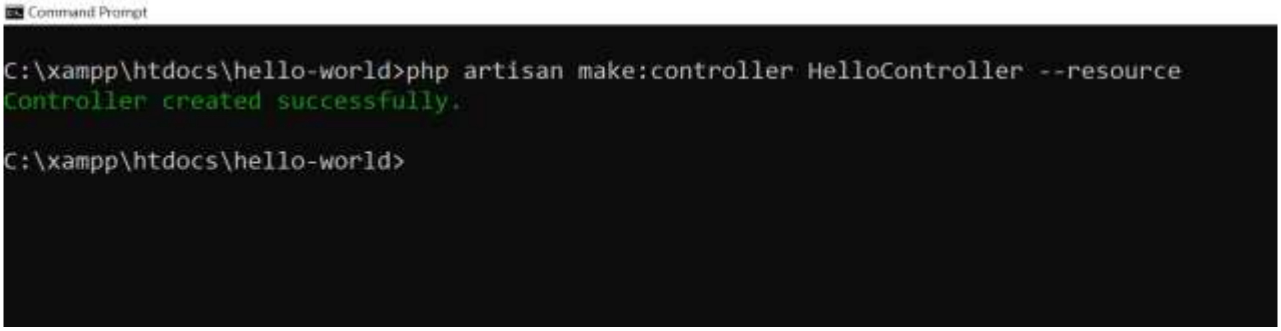
```
C:\xampp\htdocs>composer create-project laravel/laravel hello-world
Creating a "laravel/laravel" project at "./hello-world"
Info from https://repo.packagist.org: #StandWithUkraine
Installing laravel/laravel (v9.1.1)
  - Installing laravel/laravel (v9.1.1): Extracting archive
Created project in C:\xampp\htdocs\hello-world
> @php -r "file_exists('.env') || copy('.env.example', '.env');"
Loading composer repositories with package information
Updating dependencies
Lock file operations: 108 installs, 0 updates, 0 removals
  - Locking brick/math (0.9.3)
  - Locking dflydev/dot-access-data (v3.0.1)
  - Locking doctrine/inflector (2.0.4)
  - Locking doctrine/instantiator (1.4.1)
  - Locking doctrine/lexer (1.2.3)
  - Locking dragonmantank/cron-expression (v3.3.1)
  - Locking egulias/email-validator (3.1.2)
  - Locking facade/ignition-contracts (1.0.2)
  - Locking fakerphp/faker (v1.19.0)
  - Locking filp/whoops (2.14.5)
  - Locking fruitcake/php-cors (v1.2.0)
  - Locking graham-campbell/result-type (v1.0.4)
  - Locking guzzlehttp/guzzle (7.4.1)
  - Locking guzzlehttp/promises (1.5.1)
  - Locking guzzlehttp/psr7 (2.1.0)
  - Locking hamcrest/hamcrest-php (v2.0.1)
  - Locking laravel/framework (v9.4.1)
  - Locking laravel/sail (v1.13.6)
  - Locking laravel/sanctum (v2.14.2)
  - Locking laravel/serializable-closure (v1.1.1)
  - Locking laravel/tinker (v2.7.0)
  - Locking league/commonmark (2.2.3)
```

Navigate to the project folder, e.g.

```
$ cd C:\xampp\htdocs\hello-world
```

Create a controller:

```
$ php artisan make:controller HelloController –resource
```



```
C:\xampp\htdocs\hello-world>php artisan make:controller HelloController --resource
Controller created successfully.

C:\xampp\htdocs\hello-world>
```

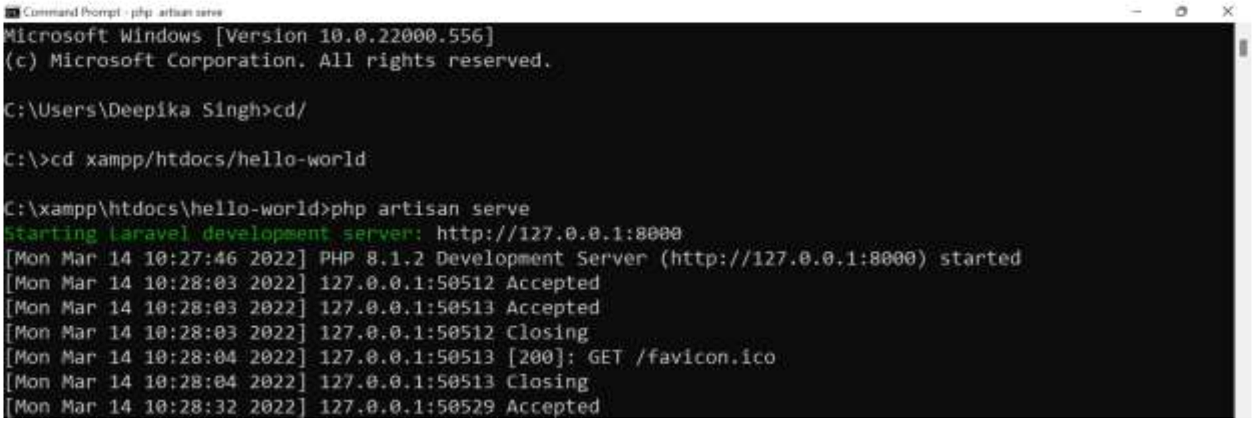
Step 1: Create a view named index.php in the resources/views directory and save the following code.

```
<!DOCTYPE html>

<html lang="en">

    <body>

        <h1>Hello World!</h1>

    </body>

</html>
```

Step 2: Add the following code segment in the routes/web.php file to register a new route.

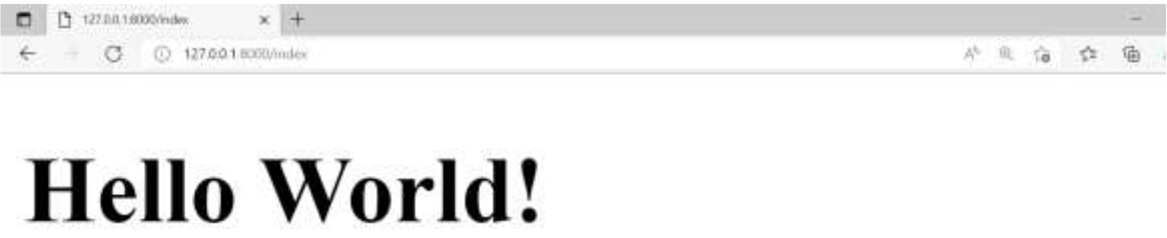
```
Route::get('/index', function () {

    return view('index');

});
```

Step 3: Visit the URL: http://127.0.0.1:8000/index to see the output.



```
Microsoft Windows [Version 10.0.22000.556]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Deepika Singh>cd/

C:\>cd xampp/htdocs/hello-world

C:\xampp\htdocs\hello-world>php artisan serve
Starting Laravel development server: http://127.0.0.1:8000
[Mon Mar 14 10:27:46 2022] PHP 8.1.2 Development Server (http://127.0.0.1:8000) started
[Mon Mar 14 10:28:03 2022] 127.0.0.1:50512 Accepted
[Mon Mar 14 10:28:03 2022] 127.0.0.1:50513 Accepted
[Mon Mar 14 10:28:03 2022] 127.0.0.1:50512 Closing
[Mon Mar 14 10:28:04 2022] 127.0.0.1:50513 [200]: GET /favicon.ico
[Mon Mar 14 10:28:04 2022] 127.0.0.1:50513 Closing
[Mon Mar 14 10:28:32 2022] 127.0.0.1:50529 Accepted
```

Step 4: The following screenshot shows the output.



Hello World!

Activity 3

Aim: Create a web application of minimum 4 blades demonstrating the blade template in Laravel.

Learning outcome: Able to understand Laravel Framework.

Duration: 5 hours

List of Hardware/Software requirements:

1. Laptop/Computer with Windows 10 / Linux OS - Ubuntu 18.04 LTS
2. Php , XAMMP or LAMP Server, composer, Laravel 8

Code/Program/Procedure (with comments):

First, I will create the routes to gain access to my page. Open routes/web.php and add the following lines in it.

```
Route::get('/', function()
{
  return View::make('pages.home');
});
Route::get('/about', function()
{
  return View::make('pages.contact');
});
```

We are only working on views only, and thus don’t need to access the controllers. To learn more about controllers, take a look at: Working with Controllers and Middleware

Create Views Structure

Now that the routes are ready, let’s create the Views structure by creating the following folders and files.

```
- resources

-- views

--- layouts

------- default.blade.php

--- pages

------- home.blade.php

------- contact.blade.php

--- includes

------- head.blade.php

------- header.blade.php

------- footer.blade.php
```

Create Includes

Create the following includes, with the following code:

head.blade.php

```
<meta charset="utf-8">

<meta name="description" content="">

<meta name="Saquib" content="Blade">
```

```
<title>Checkout our layout</title>
<!-- load bootstrap from a cdn -->
<link rel="stylesheet" href="//netdna.bootstrapcdn.com/twitter-bootstrap/3.0.3/css/bootstrap-combined.min.css">
```

Header.blade.php

```
<div class="navbar">
   <div class="navbar-inner">
       <a id="logo" href="/">Single Malt</a>
       <ul class="nav">
           <li><a href="/">Home</a></li>
           <li><a href="/contact">Contact</a></li>
       </ul>
   </div>
</div>
```

footer.blade.php

```
<div id="copyright text-right">© Copyright 2017 Saquib Rizwan </div>
```

Create the Layout

I will use @include to bring in tiny parts of the code that I have created in includes folders, and @yield to bring in content from the individual pages I will be using.

```
<!doctype html>
```

```
<html>
<head>
  @include('includes.head')
</head>
<body>
<div class="container">
  <header class="row">
     @include('includes.header')
  </header>
  <div id="main" class="row">
        @yield('content')
  </div>
  <footer class="row">
     @include('includes.footer')
  </footer>
</div>
</body>
</html>
```

Blade allows the use of the layout that I just created by using @extends. By creating @section, I will create a section that will be used in the layout. Here I will use @section(‘content’) and in the layout, everything that I will type here will be injected in @yield in the layout.

Go to resources/views/pages and put the following code in these files.

pages/home.blade.php

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

@section('content')

   i am the home page

@stop
```

pages/contact.blade.php

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

@section('content')

   i am the contact page

@stop
```

Activity 4

Aim: Create a Web Application which demonstrate routing through controller.

Learning outcome: Able to understand Laravel Framework.

Duration: 5 hours

List of Hardware/Software requirements:

1. Laptop/Computer with Windows 10 / Linux OS - Ubuntu 18.04 LTS
2. Php , XAMMP or LAMP Server, composer, Laravel 8

Code/Program/Procedure (with comments):

Here are some of the steps for creating laravel routing controllers which are explained below:

Step 1: The very first step would be to create a controller. If you are not familiar with creating a controller, then go through the below points of creating a controller otherwise move directly to step 2 for Routing Controllers.

Use the below artisan command to create the controller.

Code:

```
Php artisan make: Controller MyController
```

MyController.php file will be created whose default code is as below.

Code:

```
<?php

namespace App\Http\Controllers;
```

```
use App\Http\Controllers\Controller;

class MyController extends Controller

{

public function show($id)

{

//

}

}
```

Step 2: Now you have to write this below route in web.php file.

Code:

```
Route::get('/post','MyController@show');
```

Here the first parameter is URL which you want access to, and MyController is pretty obviously our controller name. The ‘show’ as you can see in MyController.php file, is the method. So, @show here indicates that the show() method would be called when we hit the URL ‘/post’.

Step 3: You can now add coding lines as shown below.

Code:

```
<?php

namespace App\Http\Controllers;

use App\Http\Controllers\Controller;

class MyController extends Controller
```

```
{
/**
*Display resource
*
*/
public function show($id)
{
//
}
/**
*Display resource listing
*
*/
public function index()
{
//
}
/**
*Editing resource
*
*/
public function edit($id)
{
```

```
//

}

}
```

Step 4: Now it’s time to hit the URL. You will get a specified output after entering the URL. Hopefully, we have covered enough insights of controllers that you will be able to access your Controller now.

Activity 5

Aim: Create a Form which inserts the data in the database

Learning outcome: Create a Form which inserts the data in the database

Duration: 7 hrs

List of Hardware/Software requirements:

1. Laptop/Computer with Windows 10 / Linux OS - Ubuntu 18.04 LTS
2. Php , XAMMP or LAMP Server, composer, Laravel 8

Code/Program/Procedure (with comments):

// First, creating table with SQL query:

```
CREATE TABLE student_details

(

id int NOT NULL AUTO_INCREMENT,

first_name varchar(50),

last_name varchar(50),

city_name varchar(50),

email varchar(50),

PRIMARY KEY (id)

);
```

// Now, create three files for insert data in Laravel.

Step 1. Create a controller name as StudInsertController.php.

The file location is : (app/Http/Controllers/StudInsertController.php)

Step 2. Create view page name as stud_create.php

The file location is: (resources/views/stud_create.php)

// Then put this code in your StudInsertController.php.

```
<?php
namespace App\Http\Controllers;
use Illuminate\Http\Request;
use DB;
use App\Http\Requests;
use App\Http\Controllers\Controller;
class StudInsertController extends Controller
{
        //
public function insertform(){
return view('stud_create');
        }
        public function insert(Request $request){
        $first_name = $request->input('first_name');
        $last_name = $request->input('last_name');
        $city_name = $request->input('city_name');
        $email = $request->input('email');
$data=array('first_name'=>$first_name,"last_name"=>$last_name,"city_name"=>$city_name,"email"=>$email);
        DB::table('student_details')->insert($data);
        echo "Record inserted successfully.<br/>";
        echo '<a href = "/insert">Click Here</a> to go back.';
        }
        }
// Then put this code in your stud_create.blade.php.
<!DOCTYPE html>
<html lang="en">
<head>
          <title>Student Management</title>
          <meta charset="utf-8">
          <meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css">
          <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.js">
</script>
          <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.min.js">
</script>
```

```
</head>
<body>
<div class="container">
        <h2 class="text-center">Student Management | Add</h2><br>
<form action = "/create" method = "post" class="form-group" style="width:70%; margin-left:15%;" action="/action_page.php">
<input type = "hidden" name = "_token" value = "<?php echo csrf_token(); ?>">
<input type = "hidden" name = "_token" value = "<?php echo csrf_token(); ?>">
<label class="form-group">First Name:</label>
<input type="text" class="form-control" placeholder="First Name" name="first_name">
                <label>Last Name:</label>
<input type="text" class="form-control" placeholder="Last Name" name="last_name">
                <label>City Name:</label>
                <select class="form-control" name="city_name">
                        <option value="bhubaneswar">Bhubaneswar</option>
                        <option value="cuttack">Cuttack</option>
                </select>
<label>Email:</label>
<input type="text" class="form-control" placeholder="Enter Email" name="email"><br>
<button type="submit"  value = "Add student" class="btn btn-primary">Submit</button>
  </form>
</div>
</body>
```

```
</html>
```

Step 3. Then go to routes as web.php and put this code.

The file location is : (routes/web.php)

```
Route::get('insert','StudInsertController@insertform');

Route::post('create','StudInsertController@insert');
```

// Then, you can see the data is inserted.

// So, the data is inserted in the database, then we need to retrieve a record or data from the MySQL database.

Step 4. Create a controller for view name as StudViewController.php.

The file location is : (app/Http/Controllers/StudViewController.php)

Step 5. Create a view page name as stud_view.blade.php.

The file location is : (resources/views/stud_view.blade.php).

// Then put this code StudViewController.php.

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

use DB;

use App\Http\Controllers\Controller;

class StudViewController extends Controller

{

        //

        public function index(){

        $users = DB::select('select * from student_details');
        return view('stud_view',['users'=>$users]);
      }
}
// Then, retrieve the students data then put this code stud_view.blade.php.
<!DOCTYPE html>
<html lang="en">
<head>
        <title>View Student Records</title>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1">
 <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css">
        <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
 <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.js">
</script>
        <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.min.js">
</script>
</head>
<body>
<div class="container">
        <h2 class="text-center">View Student Records</h2>
                <table class="table table-bordered table-striped">
```

```
     <thead>
          <tr>
               <th>ID</th>
               <th>First Name</th>
               <th>Last Name</th>
               <th>City Name</th>
               <th>Email</th>
          </tr>
  </thead>
  <tbody>
  @foreach ($users as $user)
   <tr>
     <td>{{ $user->id }}</td>
     <td>{{ $user->first_name }}</td>
     <td>{{ $user->last_name }}</td>
     <td>{{ $user->city_name }}</td>
     <td>{{ $user->email }}</td>
   </tr>
   @endforeach
  </tbody>
 </table>
</div>
</body>
```

```
</html>
```

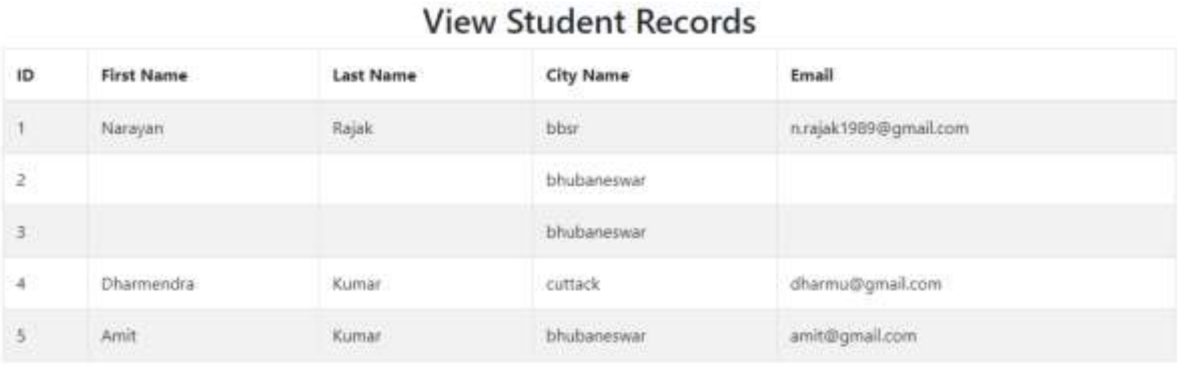
Step 6. Then go to routes as web.php and put this code.

The file location is (routes/web.php)

```
Route::get('view-records','StudViewController@index');
```

// Then, you can see the view page.

Output/Results snippet:



View Student Records

ID	First Name	Last Name	City Name	Email
1	Narayan	Rajak	bbsr	n.rajak1989@gmail.com
2			bhubaneswar	
3			bhubaneswar	
4	Dharmendra	Kumar	cuttack	dharmu@gmail.com
5	Amit	Kumar	bhubaneswar	amit@gmail.com

References:

- https://www.devopsschool.com/blog/how-to-insert-and-retrieve-data-in-database-laravel-framework/

Activity 6

Aim: Demonstrate all the CRUD operation using (DB) from controllers

Learning outcome: Demonstrate all the CRUD operation using (DB) from controllers

Duration: 7 hrs.

List of Hardware/Software requirements:

1. Laptop/Computer with Windows 10 / Linux OS - Ubuntu 18.04 LTS
2. Php , XAMMP or LAMP Server, composer, Laravel 8

Code/Program/Procedure (with comments):

Step 1 - Setup Laravel project

// Using the Composer create-project command.

```
composer create-project --prefer-dist laravel/laravel laravel_crud
```

Step 2 - Create a database named laravel_crud and configure it.

// Open the .env file in your root path and setup the username and password.

```
LOG_CHANNEL=stack

DB_CONNECTION=mysql

DB_HOST=127.0.0.1

DB_PORT=3306

DB_DATABASE=laravel_crud

DB_USERNAME=root

DB_PASSWORD=
```

// Create a table named articles. You can so it creating a migration:

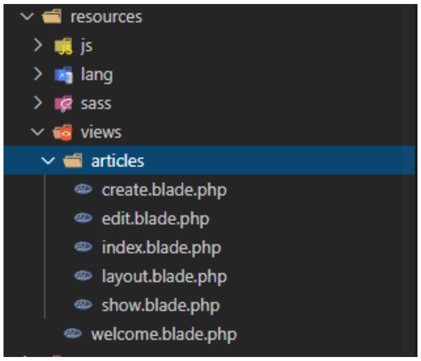
```
php artisan make:migration create_articles_table --create=articles
```

Check the file created in database/migrations/ and make this edition:

```
    public function up()
    {
        Schema::create('articles', function (Blueprint $table) {
            $table->increments('id');
            $table->string('topic');
            $table->text('description');
            $table->string('categorie');
            $table->timestamps();
        });
    }
// Migrate it using
php artisan migrate
/*Sometimes it can give an error about maxLength, collection, or the charset. Therefore,
if it is your case open AppServiceProvider.php in app/Providers/ and make this file looks
like this:*/
<?php
namespace App\Providers;
use Illuminate\Support\ServiceProvider;
use Illuminate\Support\Facades\Schema;
class AppServiceProvider extends ServiceProvider
{
    /**
```

```
     * Register any application services.
     *
     * @return void
     */
    public function register()
    {
        //
    }
    /**
     * Bootstrap any application services.
     *
     * @return void
     */
    public function boot()
    {
        Schema::defaultStringLength(191);
    }
}
```

Step 3 - Create the views.



```
resources
  js
  lang
  sass
  views
    articles
      create.blade.php
      edit.blade.php
      index.blade.php
      layout.blade.php
      show.blade.php
    welcome.blade.php
```

// Setup a Resource Router. Open routes/web.php and add this line:

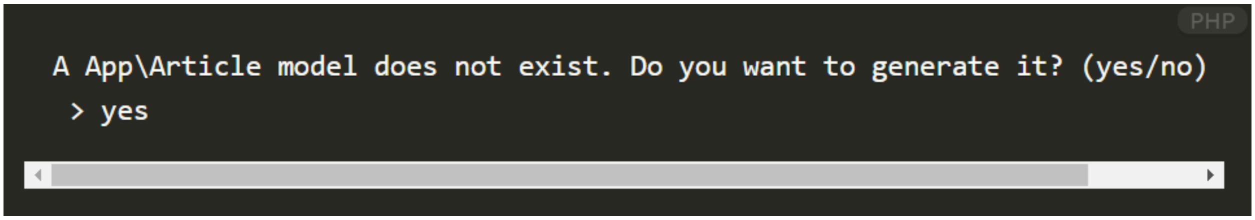
Route::resource('articles', 'ArticleController');

Step 4 - Create Controller and Model

// We need to create a new controller as ArcticleController as well as its Model.

php artisan make:controller ArticleController--resource --model=Article

// Enter yes to the question that will appear.



```
A App\Article model does not exist. Do you want to generate it? (yes/no)
 > yes
```

// Open your model Article.php and create these variables:

<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Article extends Model

{

protected $fillable = [

```
        'topic','description', 'categorie',
    ];
}
// Open ArticleController.php:
// index() lists table data and make a pagination with three elements on each page.
    public function index()
    {
        $articles = DB::table('articles')->paginate(3);
        return view('articles.index', ['articles' => $articles]);
    }
// create() – Returns the View to create a table item.
    public function create()
    {
        return view('articles.create');
    }
// store() – Saves the new item to the table
    public function store(Request $request)
    {
        $request->validate([
            'topic' => 'required',
            'description' => 'required',
            'categorie' => 'required',
        ]);
        Article::create($request->all());
        return redirect()->route('articles.index')
        ->with('success','Article created successfully.');
    }
// show() – Shows a specific item.
    public function show(Article $article)
    {
        return view('articles.show',compact('article'));
    }
// edit() – Returns the View for editing the data
    public function edit(Article $article)
    {
        return view('articles.edit',compact('article'));
    }
// update() – Save data update.
    public function update(Request $request, Article $article)
    {
        $request->validate([
            'topic' => 'required',
            'description' => 'required',
            'categorie' => 'required',
        ]);
        $article->update($request->all());
        return redirect()->route('articles.index')
                        ->with('success','Article updated successfully');
    }
// delete() – Removes data.
    public function destroy(Article $article)
    {
        $article->delete();
          return redirect()->route('articles.index')
                        ->with('success','Article deleted successfully');
    }
```

Execute:

```
php artisan serve
```

And go to: http://127.0.0.1:8000/articles

```
// Setup the Layout
// Open layout.blade.php
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Layout</title>
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
```

```
integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
crossorigin="anonymous">

</head>

<body>

<div class="container">

  <!-- Content here -->

  @yield('content')

</div>

</body>

</html>

// READ Operation

// Open index.blade.php

@extends('articles.layout')

@section('content')

...

...

@endsection
```

Inside the section content add:

Button to add a new article

```
<div class="row">

        <div class="col-lg-12 margin-tb">

            <div class="pull-right">
```

```
                <a class="btn btn-primary" href="{{ route('articles.create') }}"> Create new article</a>
            </div>
        </div>
</div>
```

Section to display a success message

```
@if ($message = Session::get('success'))
        <div class="alert alert-success">
            <p>{{ $message }}</p>
        </div>
@endif
```

And a table to display the articles:

```
<table class="table table-striped table-bordered">
  <thead>
    <tr>
      <th scope="col">#</th>
      <th scope="col">Topic</th>
      <th scope="col">Description</th>
      <th scope="col">Categorie</th>
      <th scope="col"></th>
    </tr>
  </thead>
  <tbody>
```

```
@foreach ($articles as $article)
    <tr>
      <th scope="row">{{ $article->id }}</th>
      <td>{{ $article->topic }}</td>
      <td>{{ $article->description }}</td>
      <td>{{ $article->categorie }}</td>
      <td>
      <form action="{{ route('articles.destroy',$article->id) }}" method="POST">
       <a class="btn btn-info" href="{{ route('articles.show',$article->id) }}">Show</a>
        <a class="btn btn-primary" href="{{ route('articles.edit',$article->id) }}">Edit</a>
        @csrf
        @method('DELETE')
        <button type="submit" class="btn btn-danger">Delete</button>
        </form>
      </td>
    </tr>
@endforeach
  </tbody>
</table>
```

Finally, the links for the pagination:

```
{{ $articles->links() }}
```

// CREATE Operation

// Bellow, we have a button to return to the home/initial page, a section to display an error message and the form.

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

@section('content')

<div class="row">

    <div class="col-lg-12 margin-tb">

        <div class="pull-left">

            <h2>Create Ne   w Article</h2>

        </div>

        <div class="pull-right">

            <a class="btn btn-info" href="{{ route('articles.index') }}"> Back</a>

        </div>

    </div>

</div>

@if ($errors->any())

    <div class="alert alert-danger">

        <strong>Warning!</strong> Please check your fields<br><br>

        <ul>

            @foreach ($errors->all() as $error)

                <li>{{ $error }}</li>

            @endforeach

        </ul>

    </div>
```

```
@endif
<form action="{{ route('articles.store') }}" method="POST">
@csrf
  <div class="form-group">
    <label for="topic">Topic title</label>
    <input type="text" class="form-control"  placeholder="Enter Topic" name ="topic">
  </div>
  <div class="form-group">
    <label for="description">Description</label>
    <textarea class="form-control" rows="3"placeholer ="Enter description" name="description"></textarea>
  </div>
  <div class="form-group">
    <label for="categorie">Categorie</label>
    <input type="text" class="form-control” placeholder="Enter Categorie" name ="categorie">
  </div>
  <button type="submit" class="btn btn-primary">Submit</button>
</form>
@endsection
// UPDATE Operation
// A form to update the article:
```

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

@section('content')

<div class="row">

    <div class="col-lg-12 margin-tb">

        <div class="pull-left">

            <h2>Edit Article</h2>

        </div>

        <div class="pull-right">

            <a class="btn btn-info" href="{{ route('articles.index') }}"> Back</a>

        </div>

    </div>

</div>

@if ($errors->any())

        <div class="alert alert-danger">

            <strong>Warning!</strong> Please check your fields.<br><br>

            <ul>

                @foreach ($errors->all() as $error)

                    <li>{{ $error }}</li>

                @endforeach

            </ul>

        </div>

@endif
```

```
<form action="{{ route('articles.update',$article->id) }}" method="POST">
@csrf
@method('PUT')
  <div class="form-group">
    <label for="topic">Topic title</label>
    <input type="text" class="form-control" value="{{ $article->topic }}" placeholder="Enter Topic" name ="topic">
  </div>
  <div class="form-group">
    <label for="description">Description</label>
    <textarea class="form-control" rows="3" name="description">{{ $article->description }}</textarea>
  </div>
<div class="form-group">
    <label for="categorie">Categorie</label>
    <input type="text" class="form-control" value="{{ $article->categorie }}" placeholder="Enter Categorie" name ="categorie">
  </div>
  <button type="submit" class="btn btn-primary">Submit</button>
</form>
@endsection

// DELETE Operaton
// On index.blade.php you saw that we have these lines:
```

```
@method('DELETE')

<button type="submit" class="btn btn-danger">Delete</button>
```

References:

- https://codesource.io/build-a-complete-laravel-crud-application-with-mysql/#Create-Controller-and-Model

Activity 7

Aim: Create a Model and Demonstrate all CRUD operations

Learning outcome: Create a Model and Demonstrate all CRUD operations

Duration: 8 hrs.

List of Hardware/Software requirements:

1. Laptop/Computer with Windows 10 / Linux OS - Ubuntu 18.04 LTS
2. Php , MySQL , XAMMP or LAMP Server, composer, Laravel 8

Code/Program/Procedure (with comments):

Step 1 — Installing Laravel 8 using composer

// Open a new command-line interface and run the following command:

```
$ composer create-project laravel/laravel=8.0 laravel8app --prefer-dist
```

// Wait for composer to install the dependencies and set you your project

Step 2 — Setting up a MySQL Database

// In your terminal, run the following command to run the mysql client:

```
$ mysql -u root -p
```

// When prompted, enter the password for your MySQL server when you've installed it.

// Next, run the following SQL statement to create a db database:

```
mysql> create database db;
```

// Open the .env file and update the credentials to access your MySQL database:

```
DB_CONNECTION=mysql

DB_HOST=127.0.0.1

DB_PORT=3306
```

```
DB_DATABASE=db

DB_USERNAME=root

DB_PASSWORD=******
```

// You need to provide the database name, the username and password.

// run the migrate command to create your database and a bunch of SQL tables needed by Laravel:

```
$ php artisan migrate
```

Note: You can run the migrate command at any other points of your development to add other SQL tables in your database or to later your database if you need to add any changes later.

Step 3 — Creating a Database Migration

// Open your terminal and run the following commands:

```
$ cd laravel8app

$ php artisan make:migration create_products_table --create=products
```

A migration file will be created inside the database/migrations folder of your product, next we need to add the fields to our database table. A product will have a name, description, price, date created, and date updated.

Open the create_products_table.php file that contains the migration class and update it as follows:

```
<?php

use Illuminate\Database\Migrations\Migration;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Support\Facades\Schema;

class CreateProductsTable extends Migration

{
```

```
/**
 * Run the migrations.
 *
 * @return void
 */
public function up()
{
    Schema::create('products', function (Blueprint $table) {
        $table->id();
        $table->string('name', 255)->nullable();
        $table->string('description', 500)->nullable();
        $table->decimal('price', 22)->nullable()->default(0.00);
        $table->timestamp('created_at')->useCurrent();
        $table->timestamp('updated_at')->nullable();
    });
}
/**
 * Reverse the migrations.
 *
 * @return void
 */
public function down()
{
        Schema::dropIfExists('products');
    }
}
```

To avoid running into errors, you need to specify the default string length before running your migration.

Open the app/Providers/AppServiceProvider.php file and add Schema::defaultstringLength(191) as follows:

```
<?php
namespace App\Providers;
use Illuminate\Support\ServiceProvider;
use Illuminate\Support\Facades\Schema;
class AppServiceProvider extends ServiceProvider
{
    /**
     * Register any application services.
     *
     * @return void
     */
    public function register()
    {
        //
    }
    /**
     * Bootstrap any application services.
     *
     * @return void
     */
    public function boot()
    {
        Schema::defaultStringLength(191);
    }
}
```

Open your terminal and run the following command:

```
$ php artisan migrate
```

This will add the fields to our database table.

Step 4 — Adding a Resource Route

// Add routes for our CRUD operations.

Open the routes\web.php file and add our resource route as follows:

```
<?php
use Illuminate\Support\Facades\Route;
use App\Http\Controllers\ProductController;
Route::get('/', function () {
    return view('welcome');
});
Route::resource('products', ProductController::class);
```

Step 5 — Adding a Laravel 8 Controller and Model

Next, we need to create a Laravel controller and model by running the following command:

```
$ php artisan make:controller ProductController --resource --model=Product
```

You'll be prompted if you want to create the Product model because it does not exist. Type yes and it will create the model and controller.

Open the app/Http/Controllers/ProductController.php file and update it as follows:

```
<?php

namespace App\Http\Controllers;


use App\Models\Product;

use Illuminate\Http\Request;


class ProductController extends Controller

{

    /**

     * Display a listing of the resource.

     *

     * @return \Illuminate\Http\Response
```

```
 */
public function index()
{
    $products = Product::latest()->paginate(5);

    return view('products.index', compact('products'))
        ->with('i', (request()->input('page', 1) - 1) * 5);
}

/**
 * Show the form for creating a new resource.
 *
 * @return \Illuminate\Http\Response
 */
public function create()
{
    return view('products.create');
}

/**
 * Store a newly created resource in storage.
 *
 * @param  \Illuminate\Http\Request  $request
 * @return \Illuminate\Http\Response
 */
public function store(Request $request)
{
    $request->validate([
        'name' => 'required',
        'description' => 'required',
        'price' => 'required'
    ]);
    Product::create($request->all());
    return redirect()->route('products.index')
        ->with('success', 'Product created successfully.');
}
/**
 * Display the specified resource.
 *
 * @param  \App\Models\Product  $product
 * @return \Illuminate\Http\Response
 */
public function show(Product $product)
{
    return view('products.show', compact('product'));
}
```

```
/**
 * Show the form for editing the specified resource.
 *
 * @param  \App\Models\Product  $product
 * @return \Illuminate\Http\Response
 */
public function edit(Product $product)
{
    return view('products.edit', compact('product'));
}
/**
 * Update the specified resource in storage.
 *
 * @param  \Illuminate\Http\Request  $request
 * @param  \App\Models\product  $product
 * @return \Illuminate\Http\Response
 */
public function update(Request $request, Product $product)
{
    $request->validate([
        'name' => 'required',
        'description' => 'required',
        'price' => 'required'
        ]);
        $product->update($request->all());
        return redirect()->route('products.index')
            ->with('success', 'Product updated successfully');
    }
    /**
     * Remove the specified resource from storage.
     *
     * @param  \App\Models\Product  $product
     * @return \Illuminate\Http\Response
     */
    public function destroy(Product $product)
    {
        $product->delete();
        return redirect()->route('products.index')
            ->with('success', 'Product deleted successfully');
    }
}
```

Laravel 8 makes use of a Models folder for storing model files.

Open the app/Models/Product.php, add the following functions and the fillable, the fillable are the fields in the database that a user can fill:

```
<?php
```

```
namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;

use Illuminate\Database\Eloquent\Model;

class Product extends Model

{

    use HasFactory;

    protected $table = 'products';

    public $timestamps = true;

    protected $casts = [

        'price' => 'float'

    ];

    protected $fillable = [

        'name',

        'description',

        'price',

        'created_at'

    ];

}
```

Step 6 — Adding your Laravel 8 Blade Views

Laravel makes use of the blade templating system for views.

Inside the resources/views folder, create two Layouts and Products folders.

Update the Layouts/App.blade.php file with the following content:

```
<html>
<head>
    <title>App Name - @yield('title')</title>
    <!-- Bootstrap -->
    <link href="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/4.0.0
alpha/css/bootstrap.css" rel="stylesheet">
    <!-- Font Awesome JS -->
    <script defer src="https://use.fontawesome.com/releases/v5.0.13/js/solid.js"
        integrity="sha384-
tzzSw1/Vo+0N5UhStP3bvwWPq+uvzCMfrN1fEFe+xBmv1C/AtVX5K0uZtmcHitFZ"
crossorigin="anonymous">
    </script>
    <script defer
src="https://use.fontawesome.com/releases/v5.0.13/js/fontawesome.js"integrity="sha38
4-6OIrr52G08NpOFSZdxxz1xdNSndlD4vdcf/q2myIUVO0VsqaGHJsB0RaBE01VTOY"
crossorigin="anonymous">
    </script>
    <style>
        .footer {
            position: fixed;
            left: 0;
            bottom: 0;
            width: 100%;
            background-color: #9C27B0;
            color: white;
            text-align: center;
        }
    </style>
</head>
<body>
    @section('sidebar')
    @show
    <div class="container">
        @yield('content')
    </div>
</body>
</html>
// Open the Index.blade.php and update it as follows:
@extends('layouts.app')
@section('content')
    <div class="row">
        <div class="col-lg-12 margin-tb">
            <div class="pull-left">
                <h2>Laravel 8 CRUD Example </h2>
            </div>
            <div class="pull-right">
                <a class="btn btn-success" href="" title="Create a product"> <i class="fas fa-plus-circle"></i>
                    </a>
            </div>
        </div>
    </div>
    @if ($message = Session::get('success'))
        <div class="alert alert-success">
            <p></p>
        </div>
    @endif
    <table class="table table-bordered table-responsive-lg">
        <tr>
            <th>No</th>
            <th>Name</th>
            <th>description</th>
            <th>Price</th>
            <th>Date Created</th>
            <th>Actions</th>
        </tr>
        @foreach ($products as $product)
            <tr>
                <td></td>
                <td></td>
                <td></td>
                <td></td>

                <td></td>

                <td>

                   <form action="" method="POST">

                      <a href="" title="show">

                         <i class="fas fa-eye text-success  fa-lg"></i>

                      </a>

                      <a href="">

                         <i class="fas fa-edit  fa-lg"></i>

                      </a>

                      @csrf

                      @method('DELETE')

    <button type="submit" title="delete" style="border: none; background-color:transparent;">

                         <i class="fas fa-trash fa-lg text-danger"></i>

                      </button>

                   </form>

                </td>

             </tr>

          @endforeach

       </table>

       {!! $products->links() !!}

    @endsection
```

// Open the create.blade.php file and update it as follows:

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

@section('content')

    <div class="row">

        <div class="col-lg-12 margin-tb">

            <div class="pull-left">

                <h2>Add New Product</h2>

            </div>

            <div class="pull-right">

<a class="btn btn-primary" href="" title="Go back"> <i class="fas fa-backward "></i>
</a>

            </div>

        </div>

    </div>

    @if ($errors->any())

        <div class="alert alert-danger">

            <strong>Error!</strong>

            <ul>

                @foreach ($errors->all() as $error)

                    <li></li>

                @endforeach

            </ul>

        </div>
```

```
@endif
<form action="" method="POST" >
    @csrf
    <div class="row">
        <div class="col-xs-12 col-sm-12 col-md-12">
            <div class="form-group">
                <strong>Name:</strong>
                <input type="text" name="name" class="form-control" placeholder="Name">
            </div>
        </div>
        <div class="col-xs-12 col-sm-12 col-md-12">
            <div class="form-group">
                <strong>Description:</strong>
                <textarea class="form-control" style="height:50px" name="introduction"
                    placeholder="description"></textarea>
            </div>
        </div>
        <div class="col-xs-12 col-sm-12 col-md-12">
        <div class="form-group">
                <strong>Price:</strong>
                <input type="number" name="price" class="form-control" placeholder="Put the price">
                </div>
            </div>
            <div class="col-xs-12 col-sm-12 col-md-12 text-center">
                <button type="submit" class="btn btn-primary">Submit</button>
            </div>
        </div>
    </form>
@endsection
// Open the edit.blade.php file and update it as follows:
@extends('layouts.app')
@section('content')
    <div class="row">
        <div class="col-lg-12 margin-tb">
            <div class="pull-left">
                <h2>Edit Product</h2>
            </div>
            <div class="pull-right">
                <a class="btn btn-primary" href="" title="Go back"> <i class="fas fa-backward
"></i> </a>
            </div>
        </div>
    </div>
    @if ($errors->any())
```

```
        <div class="alert alert-danger">
            <strong>Error!</strong>
            <ul>
                @foreach ($errors->all() as $error)
                    <li></li>
                @endforeach
            </ul>
        </div>
    @endif
    <form action="" method="POST">
        @csrf
        @method('PUT')
        <div class="row">
            <div class="col-xs-12 col-sm-12 col-md-12">
                <div class="form-group">
                    <strong>Name:</strong>
                    <input type="text" name="name" value="" class="form-control"
placeholder="Name">
                </div>
            </div>
            <div class="col-xs-12 col-sm-12 col-md-12">
                <div class="form-group">
                    <strong>Description</strong>
                    <textarea class="form-control" style="height:50px" name="description"
                        placeholder="description"></textarea>
                </div>
            </div>
            <div class="col-xs-12 col-sm-12 col-md-12">
                <div class="form-group">
                    <strong>Price</strong>
                    <input type="number" name="price" class="form-control" placeholder=""
                        value="">
                </div>
            </div>
            <div class="col-xs-12 col-sm-12 col-md-12 text-center">
                <button type="submit" class="btn btn-primary">Submit</button>
            </div>
        </div>
    </form>
@endsection
// Open the show.blade.php file and update it as follows:

@extends('layouts.app')
@section('content')
    <div class="row">
        <div class="col-lg-12 margin-tb">
```

```
            <div class="pull-left">
                <h2>  </h2>
            </div>
            <div class="pull-right">
                <a class="btn btn-primary" href="" title="Go back"> <i class="fas fa-backward
"></i> </a>
            </div>
        </div>
    </div>
    <div class="row">
        <div class="col-xs-12 col-sm-12 col-md-12">
            <div class="form-group">
                <strong>Name:</strong>
            </div>
        </div>
        <div class="col-xs-12 col-sm-12 col-md-12">
            <div class="form-group">
                <strong>Description</strong>

            </div>
        </div>
        <div class="col-xs-12 col-sm-12 col-md-12">
            <div class="form-group">
```

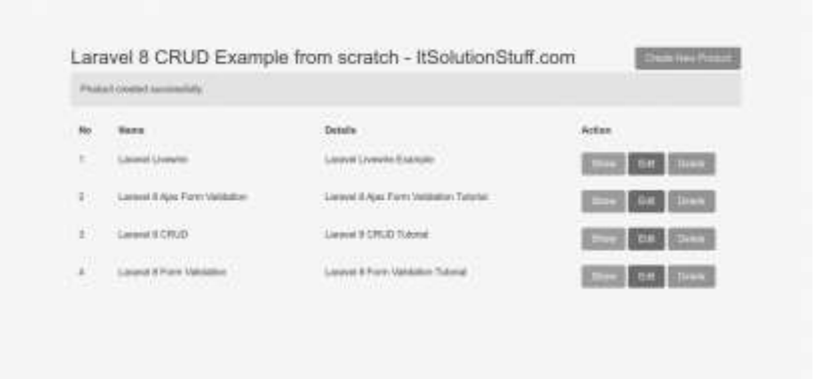
```
                <strong>Price</strong>
            </div>
        </div>
        <div class="col-xs-12 col-sm-12 col-md-12">
            <div class="form-group">
                <strong>Date Created</strong>
            </div>
        </div>
    </div>
@endsection
```

You can serve your Laravel 8 application using the following command:

```
$ php artisan serve
```

You can access your app from http://127.0.0.1:8000.

Output:



Laravel 8 CRUD Example from scratch - ItSolutionStuff.com

Product created successfully.

No	Name	Details	Action
1.	Laravel Livewire	Laravel Livewire Example	Show Edit Delete
2.	Laravel 8 Ajax Form Validation	Laravel 8 Ajax Form Validation Tutorial	Show Edit Delete
3.	Laravel 8 CRUD	Laravel 8 CRUD Tutorial	Show Edit Delete
4.	Laravel 8 Form Validation	Laravel 8 Form Validation Tutorial	Show Edit Delete

References:

- https://www.itsolutionstuff.com/post/laravel-8-crud-application-tutorial-for-beginnersexample.html

Activity 8

Aim: Creating a complete end-to–end solution which demonstrate session and database connectivity

Learning outcome: Creating a complete end-to–end solution which demonstrate session and database connectivity

Duration: 8 hrs.

List of Hardware/Software requirements:

1. Laptop/Computer with Windows 10 / Linux OS - Ubuntu 18.04 LTS
2. Php , XAMMP or LAMP Server, composer, Laravel 8

Create Laravel App

```
composer create-project --prefer-dist laravel/laravel laravel_demo_app
```

Next, head over to app folder:

```
cd laravel_demo_app
```

Connect to Database

// To add database name, username, and password into the .env configuration file to connect the laravel app to the database:

```
DB_CONNECTION=mysql

DB_HOST=127.0.0.1

DB_PORT=3306

DB_DATABASE=database_name
```

```
DB_USERNAME=database_user_name

DB_PASSWORD=database_password
```

The laravel app comes with a default User model and migration file, and we only have to run the following command to create the new table inside the database. So, get to the terminal and execute the following command to run the migration.

```
        php artisan migrate
```

Set Up Auth Controller

```
        php artisan make:controller CustomAuthController
```

Thereafter, open app\Http\Controllers\CustomAuthController.php file and carefully place the following code within the file.

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

use Hash;

use Session;

use App\Models\User;

use Illuminate\Support\Facades\Auth;

class CustomAuthController extends Controller

{

   public function index()

   {

      return view('auth.login');

   }

   public function customLogin(Request $request)
```

```
{
    $request->validate([
        'email' => 'required',
        'password' => 'required',
    ]);
    $credentials = $request->only('email', 'password');
    if (Auth::attempt($credentials)) {
        return redirect()->intended('dashboard')
                    ->withSuccess('Signed in');
    }
     return redirect("login")->withSuccess('Login details are not valid');
}
public function registration()
{
    return view('auth.registration');
}
 public function customRegistration(Request $request)
{
    $request->validate([
        'name' => 'required',
        'email' => 'required|email|unique:users',
        'password' => 'required|min:6',
    ]);
    $data = $request->all();
    $check = $this->create($data);
     return redirect("dashboard")->withSuccess('You have signed-in');
}
public function create(array $data)
{
  return User::create([
    'name' => $data['name'],
    'email' => $data['email'],
    'password' => Hash::make($data['password'])
  ]);
}
public function dashboard()
{
    if(Auth::check()){
        return view('dashboard');
    }
    return redirect("login")->withSuccess('You are not allowed to access');
}
public function signOut() {
    Session::flush();
    Auth::logout();
    return Redirect('login');
```

```
   }
}
```

Create Auth Routes

This step explains how to create routes with POST and GET methods for handling custom authentication in laravel application. Consequently, open and add the recommended code in the routes/web.php file:

```
<?php

use Illuminate\Support\Facades\Route;

use App\Http\Controllers\CustomAuthController;

Route::get('dashboard', [CustomAuthController::class, 'dashboard']);

Route::get('login', [CustomAuthController::class, 'index'])->name('login');

Route::post('custom-login', [CustomAuthController::class, 'customLogin'])->name('login.custom');

Route::get('registration', [CustomAuthController::class, 'registration'])->name('register-user');

Route::post('custom-registration', [CustomAuthController::class, 'customRegistration'])->name('register.custom');

Route::get('signout', [CustomAuthController::class, 'signOut'])->name('signout');
```

Create Auth Blade View Files

You need to create auth folder in resources/views/ folder and likewise create a new login.blade.php file within, there after place the following code in resources/views/auth/login.blade.php file:

```
@extends('app')

@section('content')

<main class="login-form">
```

```
    <div class="cotainer">
        <div class="row justify-content-center">
            <div class="col-md-4">
                <div class="card">
                    <h3 class="card-header text-center">Login</h3>
                    <div class="card-body">
                        <form method="POST" action="{{ route('login.custom') }}">
                            @csrf
                            <div class="form-group mb-3">
                                <input type="text" placeholder="Email" id="email" class="form-control" name="email" required
                                    autofocus>
                                @if ($errors->has('email'))
                                <span class="text-danger">{{ $errors->first('email') }}</span>
                                @endif
                            </div>
                            <div class="form-group mb-3">
                                <input type="password" placeholder="Password" id="password" class="form-control" name="password" required>
                                @if ($errors->has('password'))
                                <span class="text-danger">{{ $errors->first('password') }}</span>
                                @endif
                            </div>
                            <div class="form-group mb-3">
                                <div class="checkbox">
                                    <label>
                                        <input type="checkbox" name="remember"> Remember Me
                                    </label>
                                </div>
                            </div>
                            <div class="d-grid mx-auto">
                                <button type="submit" class="btn btn-dark btn-block">Signin</button>
                            </div>
                        </form>
                    </div>
                </div>
            </div>
        </div>
    </div>
</main>
@endsection
```

You have to move to resources/views/auth folder, similarly create a new registration.blade.php file within, after that update the suggested code in the resources/views/auth/registration.blade.php file:

```
@extends('app')

@section('content')
```

```
<main class="signup-form">
    <div class="cotainer">
        <div class="row justify-content-center">
            <div class="col-md-4">
                <div class="card">
                    <h3 class="card-header text-center">Register User</h3>
                    <div class="card-body">
                        <form action="{{ route('register.custom') }}" method="POST">
                            @csrf
                            <div class="form-group mb-3">
                                <input type="text" placeholder="Name" id="name" class="form-
control" name="name"
                                    required autofocus>
                                @if ($errors->has('name'))
                                <span class="text-danger">{{ $errors->first('name') }}</span>
                                @endif
                            </div>
                            <div class="form-group mb-3">
                                <input type="text" placeholder="Email" id="email_address"
class="form-control"
                                    name="email" required autofocus>
                                @if ($errors->has('email'))
                                <span class="text-danger">{{ $errors->first('email') }}</span>
                                @endif
                            </div>
                            <div class="form-group mb-3">
                                <input type="password" placeholder="Password" id="password" class="form-control"
                                    name="password" required>
                                @if ($errors->has('password'))
                                <span class="text-danger">{{ $errors->first('password') }}</span>
                                @endif
                            </div>
                            <div class="form-group mb-3">
                                <div class="checkbox">
                                    <label><input type="checkbox" name="remember"> Remember Me</label>
                                </div>
                            </div>
                            <div class="d-grid mx-auto">
                                <button type="submit" class="btn btn-dark btn-block">Sign up</button>
                            </div>
                        </form>
                    </div>
                </div>
            </div>
```

```
        </div>
    </div>
</main>
@endsection
```

Laravel 8 custom login and registration tutorial; In this tutorial, we will explain how to create custom authentication login and registration in the Laravel application.

Laravel is a top-notch PHP framework that is a boon for web developers, and it offers numerous packages and plugins to build any type of functionality.

When we talk about the authentication feature, you can install and use the Laravel JetStream package.

Having said that, we are going to share with you the traditional method through which you can create custom authentication in laravel.

This quick guide, bit by bit, describes the simple method to build custom login, registration and dashboard pages or templates.

Laravel 8 Custom Auth Login and Registration Example

Step 1: Create Laravel App

Step 2: Connect to Database

Step 3: Set Up Auth Controller

Step 4: Create Auth Routes

Step 5: Create Auth Blade View Files

Step 6: Run Laravel Development Server

Create Laravel App

We assume you have already configured Composer on your system, run the following command to install the new laravel app. However, you can skip this step if the app is already installed.

```
composer create-project --prefer-dist laravel/laravel laravel_demo_app
```

Bash

Next, head over to app folder:

```
cd laravel_demo_app
```

Bash

Connect to Database

Now, you have to add database name, username, and password into the .env configuration file to connect the laravel app to the database:

```
DB_CONNECTION=mysql

DB_HOST=127.0.0.1

DB_PORT=3306

DB_DATABASE=database_name

DB_USERNAME=database_user_name

DB_PASSWORD=database_password
```

Bash

The laravel app comes with a default User model and migration file, and we only have to run the following command to create the new table inside the database. So, get to the terminal and execute the following command to run the migration.

```
php artisan migrate
```

Bash

Set Up Auth Controller

Next, type the suggested command on the command prompt and execute the command to generate a new controller file by the name of CustomAuthController.

```
php artisan make:controller CustomAuthController
```

Bash

Thereafter, open app\Http\Controllers\CustomAuthController.php file and carefully place the following code within the file.

```
<?php
namespace App\Http\Controllers;
use Illuminate\Http\Request;
use Hash;
use Session;
use App\Models\User;
use Illuminate\Support\Facades\Auth;
class CustomAuthController extends Controller
{
    public function index()
    {
        return view('auth.login');
    }

    public function customLogin(Request $request)
    {
        $request->validate([
            'email' => 'required',
            'password' => 'required',
        ]);

        $credentials = $request->only('email', 'password');
        if (Auth::attempt($credentials)) {
            return redirect()->intended('dashboard')
                        ->withSuccess('Signed in');
        }

        return redirect("login")->withSuccess('Login details are not valid');
    }

    public function registration()
    {
        return view('auth.registration');
    }
```

```
public function customRegistration(Request $request)
{
    $request->validate([
        'name' => 'required',
        'email' => 'required|email|unique:users',
        'password' => 'required|min:6',
    ]);

    $data = $request->all();
    $check = $this->create($data);

    return redirect("dashboard")->withSuccess('You have signed-in');
}

public function create(array $data)
{
  return User::create([
    'name' => $data['name'],
    'email' => $data['email'],
    'password' => Hash::make($data['password'])
  ]);
}
```

```
    public function dashboard()
    {
        if(Auth::check()){
            return view('dashboard');
        }

        return redirect("login")->withSuccess('You are not allowed to access');
    }

    public function signOut() {
        Session::flush();
        Auth::logout();

        return Redirect('login');
    }
}
```

PHP

Create Auth Routes

This step explains how to create routes with POST and GET methods for handling custom authentication in laravel application. Consequently, open and add the recommended code in the routes/web.php file:

```
<?php
```

```
use Illuminate\Support\Facades\Route;
use App\Http\Controllers\CustomAuthController;


/*
|--------------------------------------------------------------------------
| Web Routes
|--------------------------------------------------------------------------
*/
Route::get('dashboard', [CustomAuthController::class, 'dashboard']);
Route::get('login', [CustomAuthController::class, 'index'])->name('login');
Route::post('custom-login', [CustomAuthController::class, 'customLogin'])->name('login.custom');
Route::get('registration', [CustomAuthController::class, 'registration'])->name('register-user');
Route::post('custom-registration', [CustomAuthController::class, 'customRegistration'])->name('register.custom');
Route::get('signout', [CustomAuthController::class, 'signOut'])->name('signout');
```

PHP

Create Auth Blade View Files

You need to create auth folder in resources/views/ folder and likewise create a new login.blade.php file within, there after place the following code in resources/views/auth/login.blade.php file:

```
@extends('app')
```

```
@section('content')
<main class="login-form">
    <div class="cotainer">
        <div class="row justify-content-center">
            <div class="col-md-4">
                <div class="card">
                    <h3 class="card-header text-center">Login</h3>
                    <div class="card-body">
                        <form method="POST" action="{{ route('login.custom') }}">
                            @csrf
                            <div class="form-group mb-3">
                                <input type="text" placeholder="Email" id="email" class="form-control" name="email" required
                                    autofocus>
                                @if ($errors->has('email'))
                                <span class="text-danger">{{ $errors->first('email') }}</span>
                                @endif
                            </div>
                            <div class="form-group mb-3">
                                <input type="password" placeholder="Password" id="password" class="form-control" name="password" required>
                                @if ($errors->has('password'))
                                <span class="text-danger">{{ $errors->first('password') }}</span>
                                @endif
                            </div>
                            <div class="form-group mb-3">
                                <div class="checkbox">
                                    <label>
                                        <input type="checkbox" name="remember"> Remember Me
                                    </label>
                                </div>
                            </div>
                            <div class="d-grid mx-auto">
                                <button type="submit" class="btn btn-dark btn-block">Signin</button>
                            </div>
                        </form>
                    </div>
                </div>
            </div>
        </div>
    </div>
</main>
@endsection
```

PHP

You have to move to resources/views/auth folder, similarly create a new registration.blade.php file within, after that update the suggested code in the resources/views/auth/registration.blade.php file:

```
@extends('app')

@section('content')

<main class="signup-form">

    <div class="cotainer">

        <div class="row justify-content-center">

            <div class="col-md-4">

                <div class="card">

                    <h3 class="card-header text-center">Register User</h3>

                    <div class="card-body">

                        <form action="{{ route('register.custom') }}" method="POST">

                            @csrf

                            <div class="form-group mb-3">

                                <input type="text" placeholder="Name" id="name" class="form-control" name="name"

                                    required autofocus>

                                @if ($errors->has('name'))

                                <span class="text-danger">{{ $errors->first('name') }}</span>

                                @endif

                            </div>

                            <div class="form-group mb-3">
                                <input type="text" placeholder="Email" id="email_address"
class="form-control"
                                    name="email" required autofocus>
                                @if ($errors->has('email'))
                                <span class="text-danger">{{ $errors->first('email') }}</span>
                                @endif
                            </div>
                            <div class="form-group mb-3">
                                <input type="password" placeholder="Password" id="password"
class="form-control"
                                    name="password" required>
                                @if ($errors->has('password'))
                                <span class="text-danger">{{ $errors->first('password') }}</span>
                                @endif
                            </div>
                            <div class="form-group mb-3">
                                <div class="checkbox">
                                    <label><input type="checkbox" name="remember"> Remember
Me</label>
                                </div>
                            </div>
                            <div class="d-grid mx-auto">
                                <button type="submit" class="btn btn-dark btn-block">Sign
up</button>
```

```
                            </div>
                        </form>
                    </div>
                </div>
            </div>
        </div>
    </div>
</main>
@endsection
```

Head over to resources/views/ folder, then create the new dashboard.blade.php file, then add the given below code in the resources/views/dashboard.blade.php file:

```
<!DOCTYPE html>
<html>
<head>
    <title>Custom Auth in Laravel</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/css/bootstrap.min.css" rel="stylesheet">
</head>
<body>
    <nav class="navbar navbar-light navbar-expand-lg mb-5" style="background-color: #e3f2fd;">
        <div class="container">
            <a class="navbar-brand mr-auto" href="#">PositronX</a>
    <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNav"
        aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">
        <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarNav">
        <ul class="navbar-nav">
            @guest
            <li class="nav-item">
                <a class="nav-link" href="{{ route('login') }}">Login</a>
            </li>
            <li class="nav-item">
                <a class="nav-link" href="{{ route('register-user') }}">Register</a>
            </li>
            @else
            <li class="nav-item">
                <a class="nav-link" href="{{ route('signout') }}">Logout</a>
            </li>
            @endguest
        </ul>
    </div>
</div>
```

```
    </nav>
    @yield('content')
</body>
</html>
```

Laravel 8 Custom Auth Login and Registration Example

Step 1: Create Laravel App

Step 2: Connect to Database

Step 3: Set Up Auth Controller

Step 4: Create Auth Routes

Step 5: Create Auth Blade View Files

Step 6: Run Laravel Development Server

Create Laravel App

// We assume you have already configured Composer on your system, run the following command to install the new laravel app. However, you can skip this step if the app is already installed.

```
composer create-project --prefer-dist laravel/laravel laravel_demo_app
```

Head over to app folder:

```
cd laravel_demo_app
```

Connect to Database

// To add database name, username, and password into the .env configuration file to connect the laravel app to the database:

```
DB_CONNECTION=mysql

DB_HOST=127.0.0.1
```

```
DB_PORT=3306

DB_DATABASE=database_name

DB_USERNAME=database_user_name

DB_PASSWORD=database_password
```

// The laravel app comes with a default User model and migration file, and we only have to run the following command to create the new table inside the database. So, get to the terminal and execute the following command to run the migration.

```
php artisan migrate
```

Set Up Auth Controller

// Type the suggested command on the command prompt and execute the command to generate a new controller file by the name of CustomAuthController.

```
php artisan make:controller CustomAuthController
```

// open app\Http\Controllers\CustomAuthController.php file and carefully place the following code within the file.

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

use Hash;

use Session;

use App\Models\User;

use Illuminate\Support\Facades\Auth;

class CustomAuthController extends Controller

{

    public function index()
```

```
{
    return view('auth.login');
}

public function customLogin(Request $request)
{
    $request->validate([
        'email' => 'required',
        'password' => 'required',
    ]);
    $credentials = $request->only('email', 'password');
    if (Auth::attempt($credentials)) {
        return redirect()->intended('dashboard')
                    ->withSuccess('Signed in');
    }
      return redirect("login")->withSuccess('Login details are not valid');
}
public function registration()
{
    return view('auth.registration');
}
public function customRegistration(Request $request)
{
    $request->validate([
        'name' => 'required',
        'email' => 'required|email|unique:users',
        'password' => 'required|min:6',
    ]);
    $data = $request->all();
    $check = $this->create($data);

    return redirect("dashboard")->withSuccess('You have signed-in');
}
public function create(array $data)
{
  return User::create([
    'name' => $data['name'],
    'email' => $data['email'],
    'password' => Hash::make($data['password'])
  ]);
}

public function dashboard()
{
    if(Auth::check()){
        return view('dashboard');
    }
     return redirect("login")->withSuccess('You are not allowed to access');
  }
    public function signOut() {
    Session::flush();
    Auth::logout();

    return Redirect('login');
  }
}
```

Create Auth Routes

// This step explains how to create routes with POST and GET methods for handling custom authentication in laravel application. Consequently, open and add the recommended code in the routes/web.php file:

```
<?php
use Illuminate\Support\Facades\Route;
use App\Http\Controllers\CustomAuthController;
/*
|--------------------------------------------------------------------------
| Web Routes
|--------------------------------------------------------------------------
*/
Route::get('dashboard', [CustomAuthController::class, 'dashboard']);
```

```
Route::get('login', [CustomAuthController::class, 'index'])->name('login');

Route::post('custom-login', [CustomAuthController::class, 'customLogin'])->name('login.custom');

Route::get('registration', [CustomAuthController::class, 'registration'])->name('register-user');

Route::post('custom-registration', [CustomAuthController::class, 'customRegistration'])->name('register.custom');

Route::get('signout', [CustomAuthController::class, 'signOut'])->name('signout');
```

Create Auth Blade View Files

// You need to create auth folder in resources/views/ folder and likewise create a new login.blade.php file within, there after place the following code in resources/views/auth/login.blade.php file:

```
@extends('app')

@section('content')

<main class="login-form">

    <div class="cotainer">

        <div class="row justify-content-center">

            <div class="col-md-4">

                <div class="card">

                    <h3 class="card-header text-center">Login</h3>

                    <div class="card-body">

                        <form method="POST" action="{{ route('login.custom') }}">

                            @csrf

                            <div class="form-group mb-3">
        <input type="text" placeholder="Email" id="email" class="form-control" name="email" required
            autofocus>
        @if ($errors->has('email'))
        <span class="text-danger">{{ $errors->first('email') }}</span>
        @endif
    </div>
    <div class="form-group mb-3">
        <input type="password" placeholder="Password" id="password" class="form-control" name="password" required>
        @if ($errors->has('password'))
        <span class="text-danger">{{ $errors->first('password') }}</span>
        @endif
    </div>
    <div class="form-group mb-3">
        <div class="checkbox">
            <label>
                <input type="checkbox" name="remember"> Remember Me
            </label>
        </div>
    </div>
    <div class="d-grid mx-auto">
        <button type="submit" class="btn btn-dark btn-block">Signin</button>
```

```
                            </div>
                        </form>
                    </div>
                </div>
            </div>
        </div>
    </div>
</main>
@endsection
```

// You have to move to resources/views/auth folder, similarly create a new registration.blade.php file within, after that update the suggested code in the resources/views/auth/registration.blade.php file:

```
@extends('app')
@section('content')
<main class="signup-form">
    <div class="cotainer">
        <div class="row justify-content-center">
            <div class="col-md-4">
                <div class="card">
                    <h3 class="card-header text-center">Register User</h3>
                    <div class="card-body">
                        <form action="{{ route('register.custom') }}" method="POST">
```

```
@csrf
<div class="form-group mb-3">
    <input type="text" placeholder="Name" id="name" class="form-control" name="name"
        required autofocus>
    @if ($errors->has('name'))
    <span class="text-danger">{{ $errors->first('name') }}</span>
    @endif
</div>
<div class="form-group mb-3">
    <input type="text" placeholder="Email" id="email_address" class="form-control"
        name="email" required autofocus>
    @if ($errors->has('email'))
    <span class="text-danger">{{ $errors->first('email') }}</span>
    @endif
</div>
<div class="form-group mb-3">
    <input type="password" placeholder="Password" id="password" class="form-control"
        name="password" required>
    @if ($errors->has('password'))
    <span class="text-danger">{{ $errors->first('password') }}</span>
    @endif
```

```
                            </div>
                            <div class="form-group mb-3">
                                <div class="checkbox">
                                    <label><input type="checkbox" name="remember"> Remember Me</label>
                                </div>
                            </div>
                            <div class="d-grid mx-auto">
                                <button type="submit" class="btn btn-dark btn-block">Sign up</button>
                            </div>
                        </form>
                    </div>
                </div>
            </div>
        </div>
    </div>
</main>
@endsection
```

// Head over to resources/views/ folder, then create the new dashboard.blade.php file, then add the given below code in the resources/views/dashboard.blade.php file:

```
<!DOCTYPE html>
<html>
<head>
```

```
    <title>Custom Auth in Laravel</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/css/bootstrap.min.css" rel="stylesheet">
</head>
<body>
    <nav class="navbar navbar-light navbar-expand-lg mb-5" style="background-color: #e3f2fd;">
        <div class="container">
            <a class="navbar-brand mr-auto" href="#">PositronX</a>
            <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNav"
                aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">
                <span class="navbar-toggler-icon"></span>
            </button>
            <div class="collapse navbar-collapse" id="navbarNav">
                <ul class="navbar-nav">
                    @guest
                    <li class="nav-item">
                        <a class="nav-link" href="{{ route('login') }}">Login</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="{{ route('register-user') }}">Register</a>
                    </li>
                    @else
                    <li class="nav-item">
                        <a class="nav-link" href="{{ route('signout') }}">Logout</a>
                    </li>
                    @endguest
                </ul>
            </div>
        </div>
    </nav>
    @yield('content')
</body>
</html>
```

Run Laravel Development Server

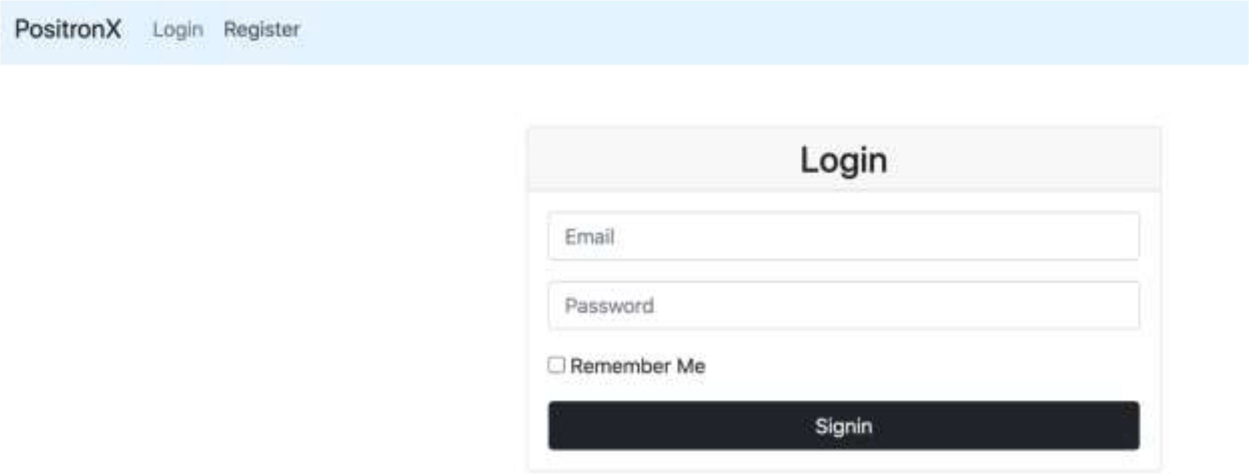
// Finally, we need to run the laravel development server, which will help us start the app on the browser. Ensure that you execute the given below command through the command prompt.

```
php artisan serve
```

Add the following url on the browser’s address bar and test the app recklessly.

http://127.0.0.1:8000/login

Output



References:

- https://www.positronx.io/laravel-custom-authentication-login-and-registration-tutorial/

Activity 9

Aim: Demonstrate csv file upload in Laravel, extract its data and insert into database

Learning outcome: Able to understand Laravel Framework

Duration: 8 hours

List of Hardware/Software requirements:

1. Laptop/Computer with Windows 10/Linux OS
2. XAMPP or LAMP Server / PHP & MySQL
3. Laravel 9 or Latest Version
4. Composer 2.2.7 or Higher Version

Code/Program/Procedure (with comments):

Install Laravel then Create the new Laravel Project

```
Composer create-project --prefer-dist laravel/laravel csv2sql
```

Step1: **Table structure**

```
CREATE TABLE `users` (`id` int(11) NOT NULL PRIMARY KEY AUTO_INCREMENT,
`first_name` varchar(80) NOT NULL, `last_name` varchar(80) NOT NULL, `gender`
varchar(10) NOT NULL, `email` varchar(80) NOT NULL) ENGINE=InnoDB DEFAULT
CHARSET=latin1;
```

Step 2: Database Configuration
Open `.env` file.

Specify the host, database name, username, and password.

```
DB_CONNECTION=mysql

DB_HOST=127.0.0.1

DB_PORT=3306

DB_DATABASE=csv2sql

DB_USERNAME=root

DB_PASSWORD=
```

Step 3: Model
Create a `Page` Model.

```
php artisan make: model csvData
```

Inserted code to csvData.php

Complete Code

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;

use Illuminate\Database\Eloquent\Model;

class CsvData extends Model

{

    use HasFactory;

    protected $fillable = [

        'first_name',

        'last_name',

        'email',

        'gender',

    ];

}
```

Step 4: Controller

Create a `ImportController` Controller.

```
php artisan make:controller ImportController
```

Add to the code

Complete Code

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

use App\Models\CsvData;

use Illuminate\Support\Facades\Session;



class ImportController extends Controller

{

 public function index(){

        return view('index');

 }

 public function uploadCsv(Request $request){

        if ($request->input('submit') != null ){

$file = $request->file('file');
```

```
// File Details
        $filename = $file->getClientOriginalName();
        $extension = $file->getClientOriginalExtension();
        $tempPath = $file->getRealPath();
        $fileSize = $file->getSize();
        $mimeType = $file->getMimeType();
// Valid File Extensions
        $valid_extension = array("csv");

// 2MB in Bytes
        $maxFileSize = 2097152;
// Check file extension
        if(in_array(strtolower($extension),$valid_extension)){
// Check file size
          if($fileSize <= $maxFileSize){
// File upload location
            $location = 'uploads';
```

```
// Upload file

            $file->move($location,$filename);

// Import CSV to Database

            $filepath = public_path($location."/".$filename);

// Reading file

            $file = fopen($filepath,"r");

        $dataRow = true;

            while (($data = fgetcsv($file, 4000, ",")) !== FALSE) {

              if (!$dataRow) {

                CsvData::create([

                  "first_name" => $data['1'],

                  "last_name" => $data['2'],

                  "email" => $data['3'],

                  "gender" => $data['4']

                ]);

              }

              $dataRow = false;
            Session::flash('message','Import Successful.');

          }

fclose($file);

 return redirect(url('/'));

 }

  else{

 Session::flash('message','File too large. File must be less than 2MB.');}

}else{

 Session::flash('message','Invalid File Extension.');

      }

    }

  }

}
```

Step 5: Routes

Open routes/web.php file.

Here, define two routes:

- **/ -** Root
- **/uploadFile –** This is post type route which uses on <form > action.

Completed Code

```
<?php

use Illuminate\Support\Facades\Route;

use App\Http\Controllers\ImportController;

Route::get('/', [ImportController::class, 'index']);

Route::post('/uploadFile', [ImportController::class, 'uploadCsv']);
```

Step 6: View

Create a new `index.blade.php` file in `resources/views/` directory.

If Session `message` variable if it is set then display value in `<p>`.

Create a `<form >` and pass `'/uploadFile'` in `action` attribute.

In `<form >` add `{{ csrf_field() }}`, a file element and a submit button.

Completed Code

```
<!doctype html>

<html>

<head>

   <title>Import CSV Data to MySQL database with Laravel</title>
```

```
</head>

<body>

    <!-- Message -->

    @if (Session::has('message'))

        <p>{{ Session::get('message') }}</p>

    @endif

    <!-- Form -->

    <form method='post' action='/uploadFile' enctype='multipart/form-data'>

        @csrf

        <input type='file' name='file'>

        <input type='submit' name='submit' value='Import'>

    </form>

</body>

</html>
```

Step 7: Run Development Server

Now we are set to run our example so run below command for quick run:

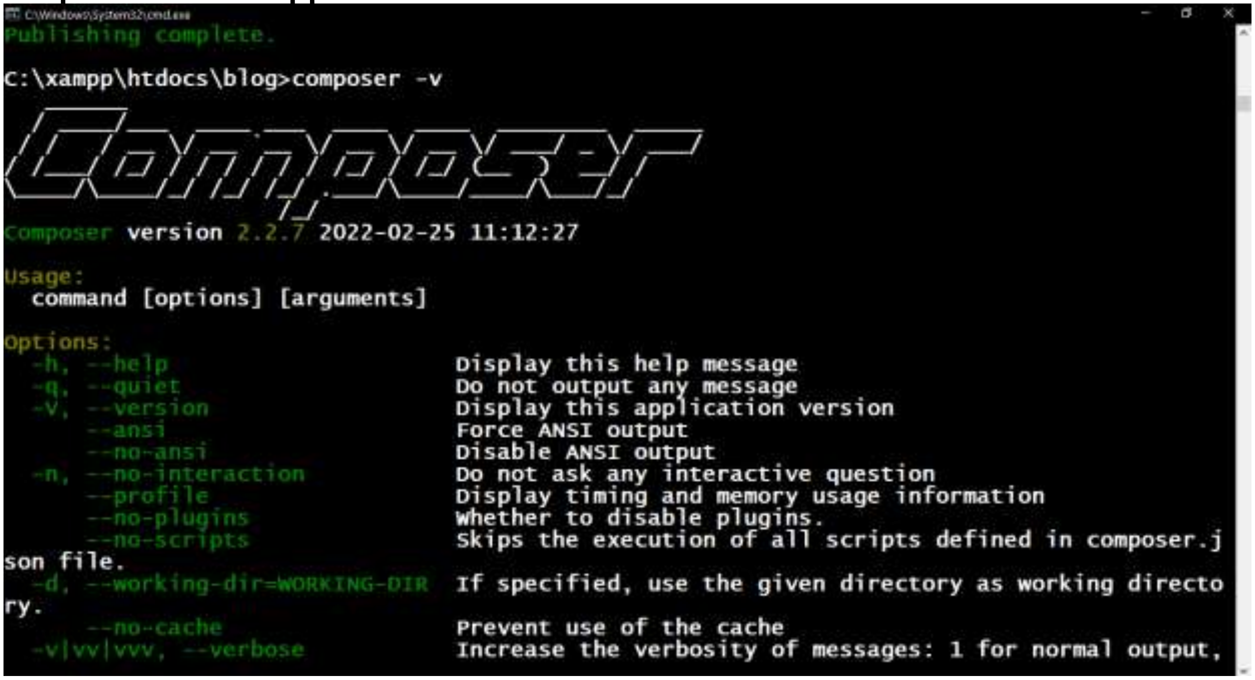
```
php artisan serve
```

Step 8:

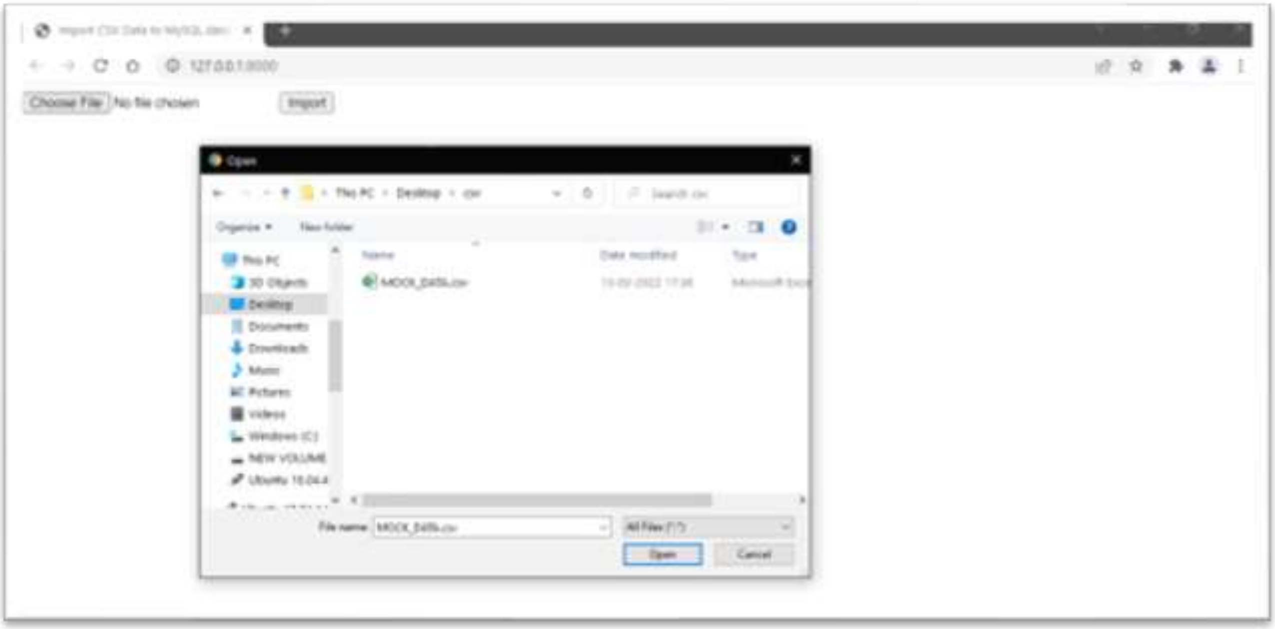
Now you can open bellow URL on your browser:

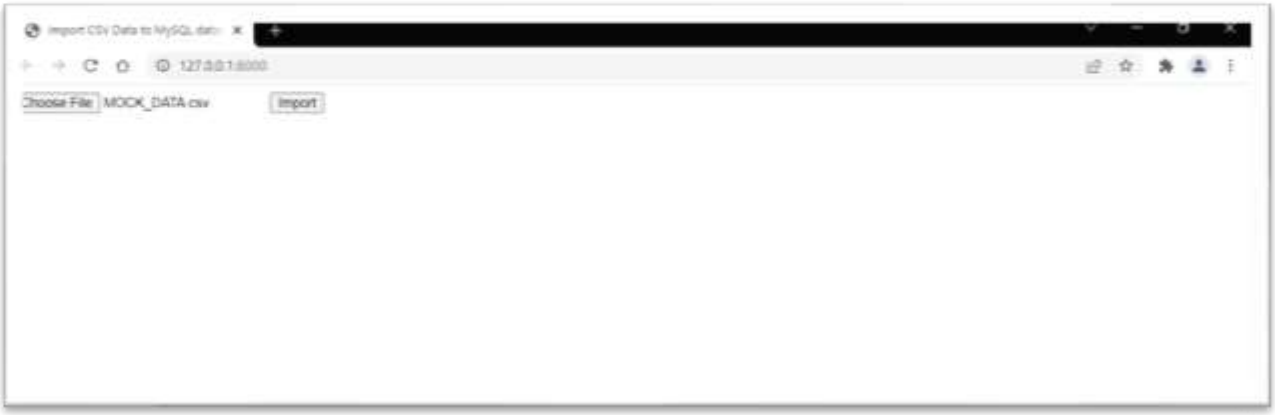
http://localhost:8000

Output/Results snippet:

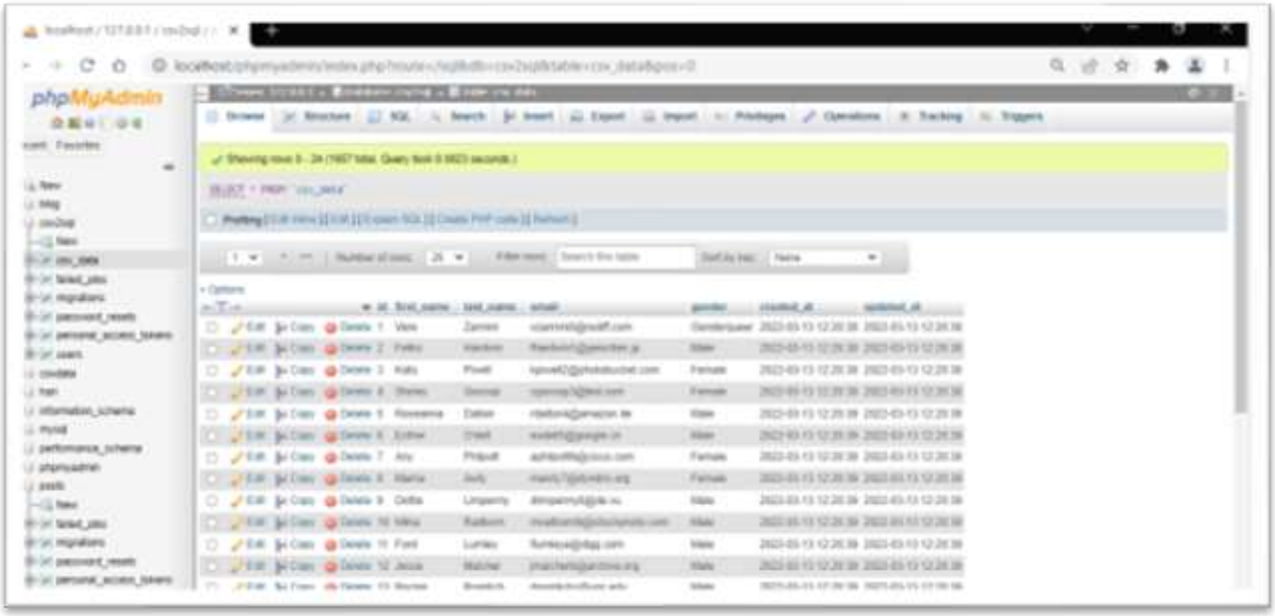


```
C:\Windows\System32\cmd.exe
Publishing complete.

C:\xampp\htdocs\blog>composer -v

Composer version 2.2.7 2022-02-25 11:12:27

Usage:
  command [options] [arguments]

Options:
  -h, --help                     Display this help message
  -q, --quiet                    Do not output any message
  -V, --version                  Display this application version
      --ansi                     Force ANSI output
      --no-ansi                  Disable ANSI output
  -n, --no-interaction           Do not ask any interactive question
      --profile                  Display timing and memory usage information
      --no-plugins               Whether to disable plugins.
      --no-scripts               Skips the execution of all scripts defined in composer.j
son file.
  -d, --working-dir=WORKING-DIR  If specified, use the given directory as working directo
ry.
      --no-cache                 Prevent use of the cache
  -v|vv|vvv, --verbose           Increase the verbosity of messages: 1 for normal output,
```





MOCK_DATA.csv


localhost / 127.0.0.1 / csv2sql / [illegible]
localhost/phpmyadmin/index.php?route=/sql&db=csv2sql&table=csv_data&pos=0
phpMyAdmin
Recent Favorites
Browse
Structure
SQL
Search
Insert
Export
Import
Privileges
Operations
Tracking
Triggers
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Number of rows:
Filter rows:
Sort by key:
Options
Edit
Copy
Delete

New
blog
csv2sql
New
csv_data
failed_jobs
migrations
password_resets
personal_access_tokens
users
csvdata
hari
information_schema
mysql
performance_schema
phpmyadmin
posts
New
failed_jobs
migrations
password_resets
personal_access_tokens

✓ Showing rows 0 - 24 (1007 total, Query took 0.0023 seconds.)

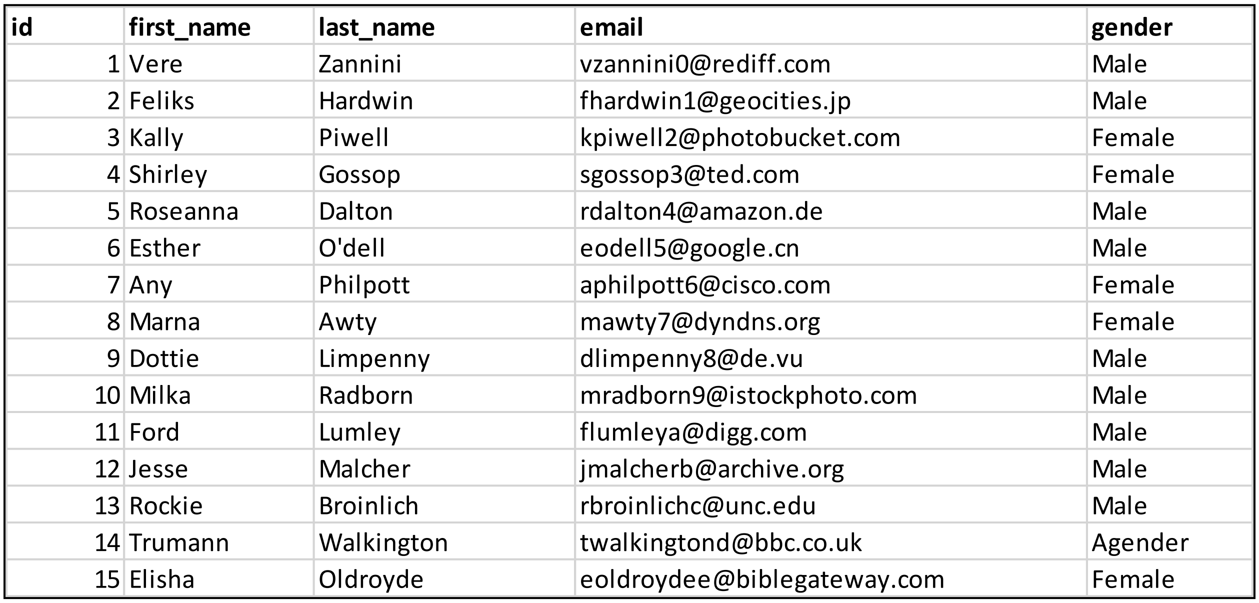
SELECT * FROM `csv_data`

Number of rows: 25

Filter rows: Search this table

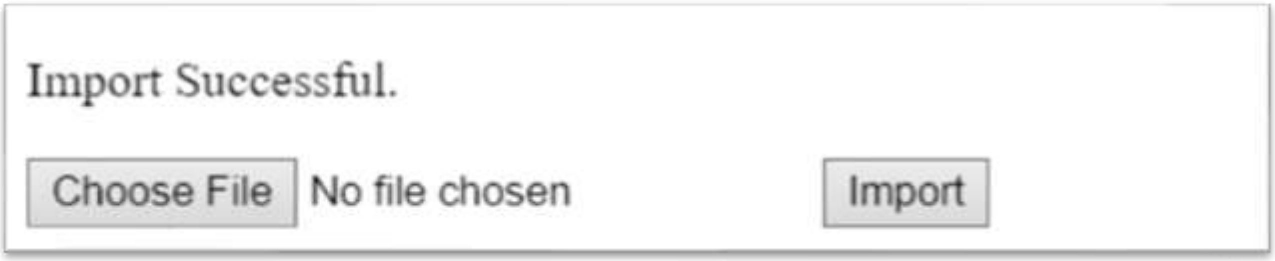
Sort by key: None

id	first_name	last_name	email	gender	created_at	updated_at
1	Vere	Zamieri	vzamirini0@rediff.com	Genderqueer	2023-03-13 12:28:38	2023-03-13 12:28:38
2	Fellio	Hardwin	fhardwin1@[illegible].jp	Male	2023-03-13 12:28:38	2023-03-13 12:28:38
3	Kaly	Powell	kpowell2@photobucket.com	Female	2023-03-13 12:28:38	2023-03-13 12:28:38
4	Sheree	Gonnop	sgonnop3@[illegible].com	Female	2023-03-13 12:28:38	2023-03-13 12:28:38
5	Roweena	Dalton	rdalton4@amazon.de	Male	2023-03-13 12:28:39	2023-03-13 12:28:39
6	Esther	Orrell	eorell5@google.cn	Male	2023-03-13 12:28:39	2023-03-13 12:28:39
7	Arly	Philpott	aphilpott6@cisco.com	Female	2023-03-13 12:28:39	2023-03-13 12:28:39
8	Marta	Awfy	mawfy7@[illegible].org	Female	2023-03-13 12:28:39	2023-03-13 12:28:39
9	Delta	Limpenny	dlimpenny8@jle.vu	Male	2023-03-13 12:28:39	2023-03-13 12:28:39
10	Milka	Radborn	mradborn9@[illegible].com	Male	2023-03-13 12:28:39	2023-03-13 12:28:39
11	Ford	Lumley	flumleya@digg.com	Male	2023-03-13 12:28:39	2023-03-13 12:28:39
12	Jecca	Matcher	jmatcherb@[illegible].org	Male	2023-03-13 12:28:39	2023-03-13 12:28:39
13	Boyce	Brenrick	[illegible]	Male	[illegible]	[illegible]



id	first_name	last_name	email	gender
1	Vere	Zannini	vzannini0@rediff.com	Male
2	Feliks	Hardwin	fhardwin1@geocities.jp	Male
3	Kally	Piwell	kpiwell2@photobucket.com	Female
4	Shirley	Gossop	sgossop3@ted.com	Female
5	Roseanna	Dalton	rdalton4@amazon.de	Male
6	Esther	O'dell	eodell5@google.cn	Male
7	Any	Philpott	aphilpott6@cisco.com	Female
8	Marna	Awty	mawty7@dyndns.org	Female
9	Dottie	Limpenny	dlimpenny8@de.vu	Male
10	Milka	Radborn	mradborn9@istockphoto.com	Male
11	Ford	Lumley	flumleya@digg.com	Male
12	Jesse	Malcher	jmalcherb@archive.org	Male
13	Rockie	Broinlich	rbroinlichc@unc.edu	Male
14	Trumann	Walkington	twalkingtond@bbc.co.uk	Agender
15	Elisha	Oldroyde	eoldroydee@biblegateway.com	Female

Mock_data.csv



Import Successful.

References:

- https://makitweb.com/import-csv-data-to-mysql-database-with-laravel/

Activity 10

Aim: Installation and usage of jet brains Live wire and its component

Learning outcome: Able to understand Laravel Framework

Duration: 8 hours

List of Hardware/Software requirements:

1. Laptop/Computer with Windows 10/Linux OS
2. XAMPP or LAMP Server / PHP & MySQL
3. Laravel 9 or Latest Version
4. Composer 2.2.7 or Higher Version
5. livewire

Code/Program/Procedure (with comments):

Install Laravel then Create the new Laravel Project

```
Composer create-project --prefer-dist laravel/laravel livewirepro
```

Step 1: Create Routes

Now we need to generate our routes. So, paste this below code in a web.php file.

routes/web.php

```
Route::view('contacts','users.contacts');
```

Here we used the view route and users.contacts shows blade view directories.

Step 2:

Now paste this following code to contacts.blade.php.

resources/views/users/contacts.blade.php
Complete Code

```
@extends('layouts.app')
@section('content')
<div class="flex justify-center">
        @livewire('contact')
</div>
```

@endsection

Step 3: Create Migration and Model

Here, we need to generate database migration for files table and also we will make a model for files table.

```
        php artisan make :model Contact -m
```

Now open Contact model and paste below code. **app/Models/Contact.php**

Complete Code

```
<?php
namespace App\Models;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
class Contact extends Model
{
   //use HasFactory;
   protected $guarded=[];
}
```

Step 4:

Now, add these fields to your contacts table and run the migrate command to save it database.

database/migrations/create_contacts_table.php

Complete code

```
<?php
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;
return new class extends Migration
{
   /**
    * Run the migrations.
    *
    * @return void
    */
   public function up()
   {
      Schema::create('posts', function (Blueprint $table) {
         $table->id();
         $table->string('title');
         $table->text('body');
```

```
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('posts');
    }
};
```

Step 5:

Then Run

php artisan migrate

Step 6: Install Livewire

You have to set up livewire in your project for that you need to run below command

resources/views/layouts/app.blade.php

composer require livewire/livewire

Step 7: Create Component

You can create livewire component using bellow command, so let’s run below command to build user form component:

php artisan make:livewire users

Now they created files on both path:

app/Http/Livewire/Users.php

resources/views/livewire/users.blade.php

Step 8:

Add Route

We need to add a route for image form in Laravel livewire application. So open your “routes/web.php” file and add the following route in it.

routes/web.php

Route::view('users','livewire.contact');

Step 9: Create Blade File

Now paste the code to your blade file. Here we will create contact, create and update files to build our Laravel livewire demo example.

resources/views/livewire/contact.blade.php

Complete Code

```
<div>

@if (session()->has('message'))

<div class="alert alert-success">

        {{session('message')}}

</div>

@endif

@if($updateMode)

        @include('livewire.update')

@else

        @include('livewire.create')

@endif

<table class="table table-bordered mt-5">

        <thead>
        <tr>
            <th>No.</th>
            <th>Title</th>
            <th>Body</th>
            <th width="150px">Action</th>
</tr>
</thead>
<tbody>
    @foreach($posts as $post)
    <tr>
        <td>{{$post->id}}</td>
        <td>{{$post->title}}</td>
        <td>{{$post->body}}</td>
        <td>
            <button wire:click="edit({{$post->id}})" class="btn btn-primary btn-sm">Edit</button>
```

```
                        <button wire:click="delete({{$post->id}})" class="btn btn-danger btn-sm">Delete</button>

</td>

</tr>

@endforeach

</tbody>

</table>

</div>
```

Step 10: Run Development Server

Now we are set to run our example so run below command for quick run:

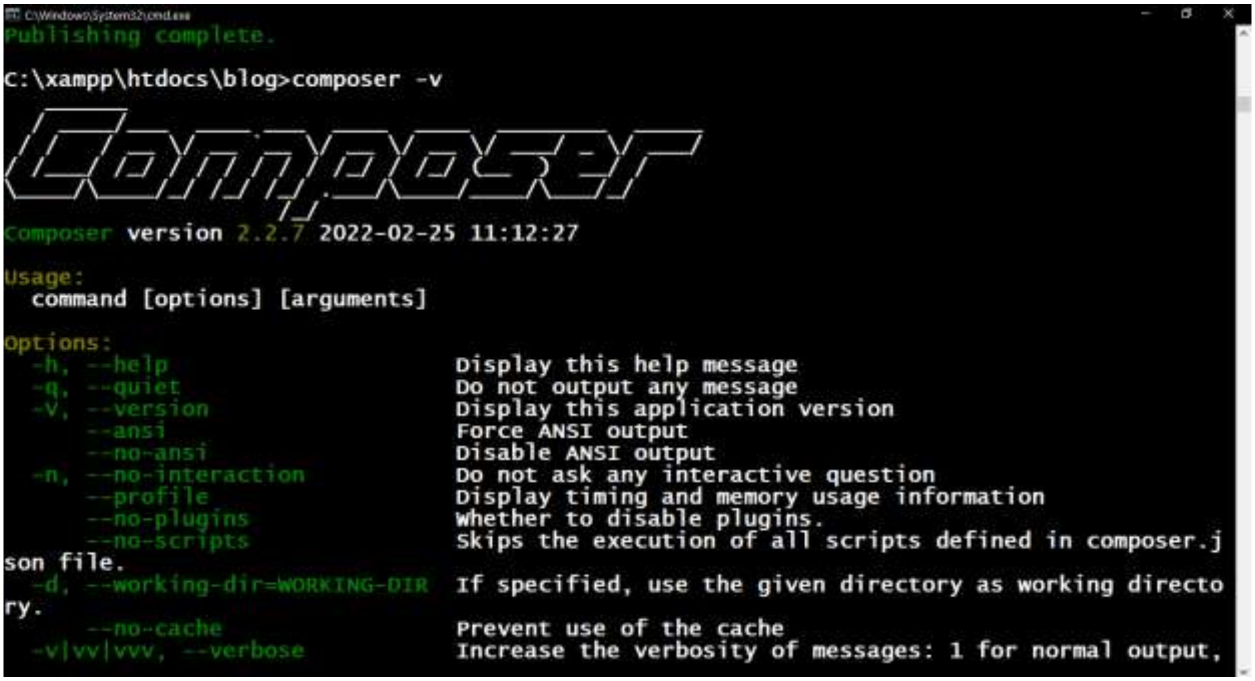
```
php artisan serve
```

Step 11:

Now you can open bellow URL on your browser:

http://127.0.0.1:8000/users

Output/Results snippet:



```
Publishing complete.

C:\xampp\htdocs\blog>composer -v
   ______
  / ____/___  ____ ___  ____  ____  ________  _____
 / /   / __ \/ __ `__ \/ __ \/ __ \/ ___/ _ \/ ___/
/ /___/ /_/ / / / / / / /_/ / /_/ (__  )  __/ /
\____/\____/_/ /_/ /_/ .___/\____/____/\___/_/
                    /_/
Composer version 2.2.7 2022-02-25 11:12:27

Usage:
  command [options] [arguments]

Options:
  -h, --help                     Display this help message
  -q, --quiet                    Do not output any message
  -V, --version                  Display this application version
      --ansi                     Force ANSI output
      --no-ansi                  Disable ANSI output
  -n, --no-interaction           Do not ask any interactive question
      --profile                  Display timing and memory usage information
      --no-plugins               Whether to disable plugins.
      --no-scripts               Skips the execution of all scripts defined in composer.j
son file.
  -d, --working-dir=WORKING-DIR  If specified, use the given directory as working directo
ry.
      --no-cache                 Prevent use of the cache
  -v|vv|vvv, --verbose           Increase the verbosity of messages: 1 for normal output,
```

References:

- https://medium.com/@avnshrathod/laravel-livewire-installation-and-demo-dfaf930f5f64

Activity 11

Aim: Create a login and register Forms using Live wire

Learning outcome: Able to understand Laravel Framework

Duration: 9 hours

List of Hardware/Software requirements:

1. Laptop/Computer with Windows 10/Linux OS
2. XAMPP or LAMP Server / PHP & MySQL
3. Laravel 9 or Latest Version
4. Composer 2.2.7 or Higher Version
5. livewire

Code/Program/Procedure (with comments):

Install Laravel then Create the new Laravel Project

```
Composer create-project --prefer-dist laravel/laravel livewirelogin
```

Step 1: Add Database Detail
In this step, Add database credentials in the .env file. So, open your project root directory and find .env file. Then add database detail in .env file:

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=here your database name here
DB_USERNAME=here database username here
DB_PASSWORD=here database password here
```

Step 2:
Run Migration Command

Now, Open command prompt and run the following command to create the table into your added database:

```
php artisan migrate
```

Step 3:
Install Livewire to Create Components

In this step, you need to install livewire to your laravel project using the following command:

```
composer require livewire/livewire
```

Step 4:
Generate Livewire Components for Login & Registration

In this step, create the livewire components for **creating a laravel livewire registration and login** components using the following command. So Open your cmd and run the following command:

```
php artisan make:livewire registration-login
```

This command will create the following components on the following path:
app/Http/Livewire/RegistrationLogin.php
resources/views/livewire/registration-login.blade.php
Now, Navigate to **app/Http/Livewire** folder and open **RegistrationLogin.php** file. Then add the following code into your **RegistrationLogin.php** file:

Complete Code

```
<?php
namespace App\Http\Livewire;
use Livewire\Component;
use Hash;
use App\Models\User;
class RegistrationLogin extends Component
{
    public $users, $email, $password, $name;
    public $registerForm = false;
     public function render()
    {
        return view('livewire.registration-login');
    }
    private function resetInputFields(){
        $this->name = '';
        $this->email = '';
        $this->password = '';
    }
    public function login()
    {
        $validatedDate = $this->validate([
            'email' => 'required|email',
            'password' => 'required',
        ]);

        if(\Auth::attempt(array('email' => $this->email, 'password' => $this->password))){
                session()->flash('message', "You have been successfully login.");
        }else{
            session()->flash('error', 'email and password are wrong.');
        }
    }
    public function register()
    {
        $this->registerForm = !$this->registerForm;
    }
    public function registerStore()
    {
        $v = $this->validate([
            'name' => 'required',
            'email' => 'required|email',
            'password' => 'required',
        ]);
        $this->password = Hash::make($this->password);
        $data = [ 'name' => $this->name,
                  'email' => $this->email,
                  'password' => $this->password
                ];

        User::create($data);
        session()->flash('message', 'You have been successfully registered.');
        $this->resetInputFields();

    }
}
```

After that, Navigate to **resources/views/livewire** folder and open **registration-login.blade.php** file. Then add the following code into your **registration-login.blade.php** file:

Compelete code

```
<div>
<div class="row">
<div class="col-md-12">
```

```
@if (session()->has('message'))
<div class="alert alert-success">
{{ session('message') }}
</div>
@endif
@if (session()->has('error'))
<div class="alert alert-danger">
{{ session('error') }}
</div>
@endif
</div>
</div>
@if($registerForm)
<form>
<div class="row">
<div class="col-md-12">
<div class="form-group">
<label>Name :</label>
<input type="text" wire:model="name" class="form-control">
@error('name') <span class="text-danger error">{{ $message }}</span>@enderror
</div>
</div>
<div class="col-md-12">
<div class="form-group">
<label>Email :</label>
<input type="text" wire:model="email" class="form-control">
@error('email') <span class="text-danger error">{{ $message }}</span>@enderror
</div>
</div>
<div class="col-md-12">
<div class="form-group">
<label>Password :</label>
<input type="password" wire:model="password" class="form-control">
@error('password') <span class="text-danger error">{{ $message }}</span>@enderror
</div>
</div>
<div class="col-md-12 text-center">
<button class="btn text-white btn-success"
wire:click.prevent="registerStore">Register</button>
</div>
<div class="col-md-12">
<a class="text-primary" wire:click.prevent="register"><strong>Login</strong></a>
```

```
</div>
</div>
</form>
@else
<form>
<div class="row">
<div class="col-md-12">
<div class="form-group">
<label>Email :</label>
<input type="text" wire:model="email" class="form-control">
@error('email') <span class="text-danger error">{{ $message }}</span>@enderror
</div>
</div>
<div class="col-md-12">
<div class="form-group">
<label>Password :</label>
<input type="password" wire:model="password" class="form-control">
@error('password') <span class="text-danger error">{{ $message }}</span>@enderror
</div>
</div>
<div class="col-md-12 text-center">
<button class="btn text-white btn-success" wire:click.prevent="login">Login</button>
</div>
<div class="col-md-12">
Don't have account? <a class="btn btn-primary text-white"
wire:click.prevent="register"><strong>Register Here</strong></a>
</div>
</div>
</form>
@endif
</div>
```

Step 5: Create Route

In this step, Navigate to routes folder and open web.php. Then add the following routes into your web.php file:

Route::view('registration-login','livewire.home');

Step 6:

Create View File

In this step, navigate to **resources/views/livewire** folder and create one blade view files that name **home.blade.php** file. Then add the following code into your **home.blade.php** file:

Complete code

```
<!DOCTYPE html>
<html lang="{{ str_replace('_', '-', app()->getLocale()) }}">
<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<title>Laravel Livewire Registration and Login</title>
<!-- Fonts -->
<link href="https://fonts.googleapis.com/css?family=Nunito:200,600" rel="stylesheet">
<link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.1/css/bootstrap.min.css">
<!-- Styles -->
<style>
html, body {
background-color: #fff;
color: #636b6f;
font-family: 'Nunito', sans-serif;
font-weight: 200;
height: 100vh;
margin: 0;
}
.full-height {
height: 100vh;
}
.flex-center {
align-items: center;
display: flex;
justify-content: center;
}
.position-ref {
position: relative;
}
.top-right {
position: absolute;
right: 10px;
top: 18px;
}
.content {
text-align: center;
}
.title {
font-size: 84px;
}
.links > a {
color: #636b6f;
padding: 0 25px;
font-size: 13px;
font-weight: 600;
letter-spacing: .1rem;
text-decoration: none;
text-transform: uppercase;
}
.m-b-md {
margin-bottom: 30px;
}
</style>
</head>
<body>
<div class="container mt-5">
<div class="row mt-5 justify-content-center">
<div class="mt-5 col-md-8">
<div class="card">
<div class="card-header bg-primary text-white"><h3>Laravel Livewire Registration and login</h3></div>
<div class="card-body">
@livewire('registration-login')
</div>
</div>
</div>
</div>
</div>
@livewireScripts
</body>
</html>
```

Step 7: Run Development Server

Finally, you need to run the following **PHP artisan serve** command to start your laravel livewire registration and login app:

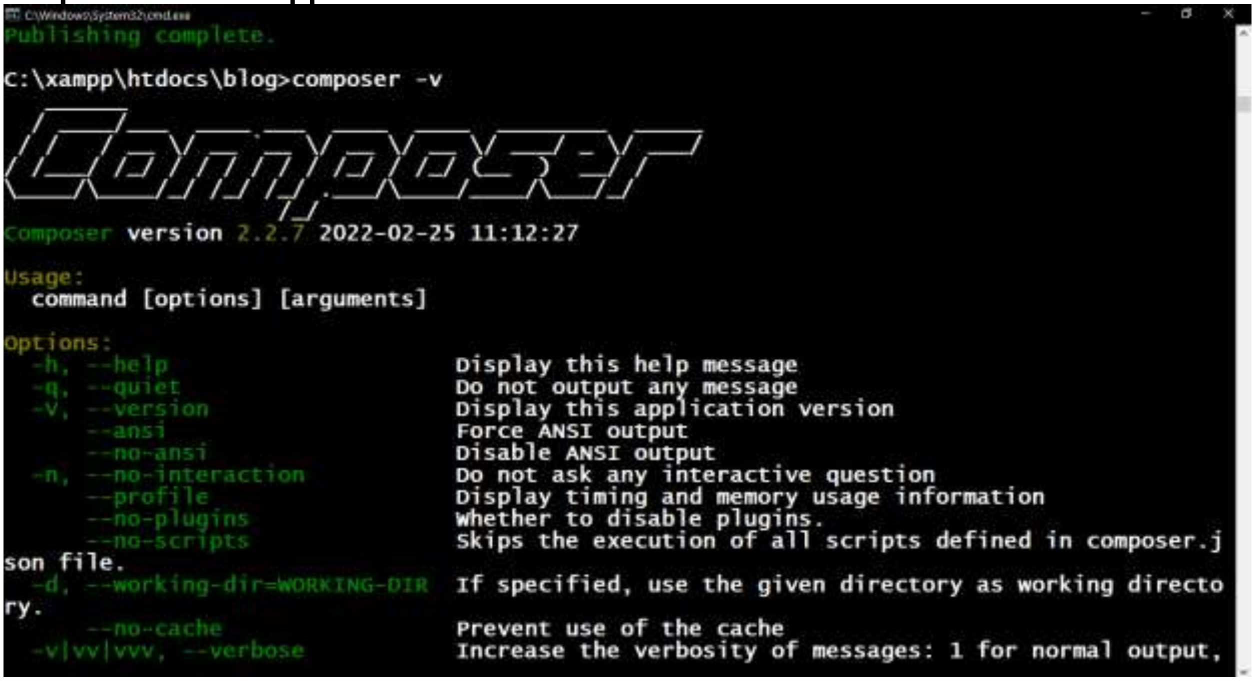
```
php artisan serve
```

Step 8:

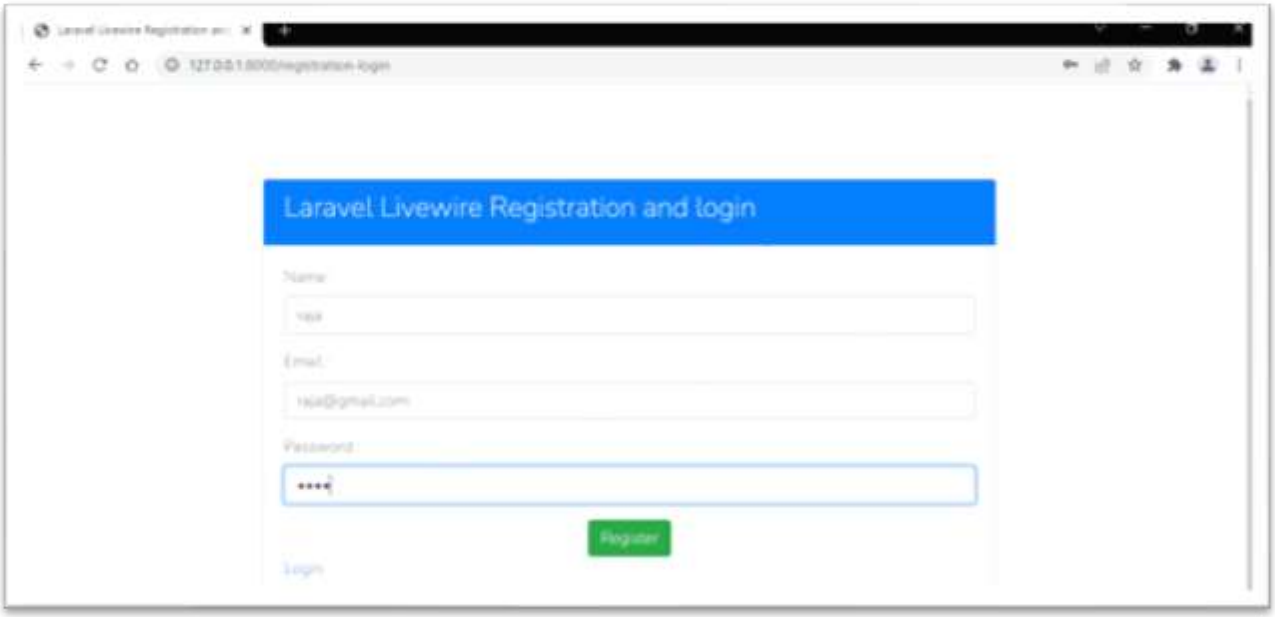
Now, you are ready to run laravel livewire registration and login app. So open your browser and hit the following URL into your browser:

http://localhost:8000/registration-login

Output/Results snippet:



```
C:\Windows\System32\cmd.exe
Publishing complete.

C:\xampp\htdocs\blog>composer -v

Composer version 2.2.7 2022-02-25 11:12:27

Usage:
  command [options] [arguments]

Options:
  -h, --help                     Display this help message
  -q, --quiet                    Do not output any message
  -V, --version                  Display this application version
      --ansi                     Force ANSI output
      --no-ansi                  Disable ANSI output
  -n, --no-interaction           Do not ask any interactive question
      --profile                  Display timing and memory usage information
      --no-plugins               Whether to disable plugins.
      --no-scripts               Skips the execution of all scripts defined in composer.j
son file.
  -d, --working-dir=WORKING-DIR  If specified, use the given directory as working directo
ry.
      --no-cache                 Prevent use of the cache
  -v|vv|vvv, --verbose           Increase the verbosity of messages: 1 for normal output,
```



Laravel Livewire Registration and login

Name

raja

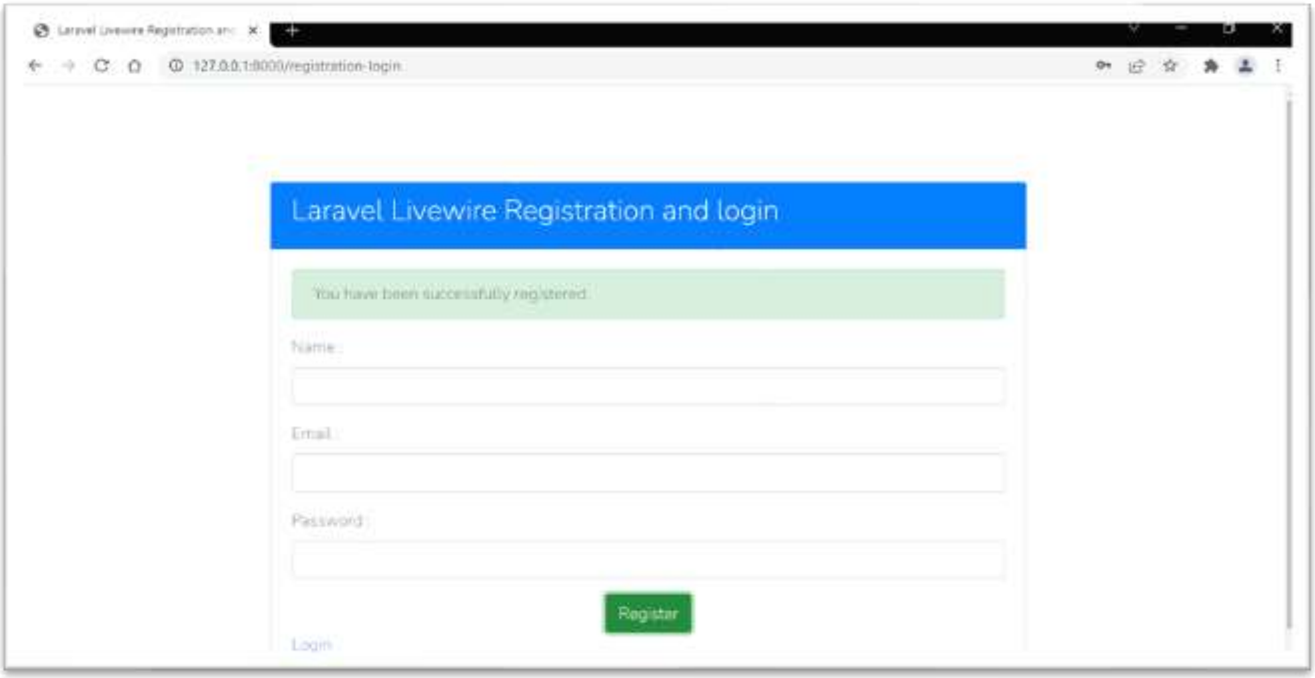
Email

raja@gmail.com

Password

••••

Laravel Livewire Registration an...
127.0.0.1:8000/registration-login
Register
Login


Laravel Livewire Registration an
127.0.0.1:8000/registration-login

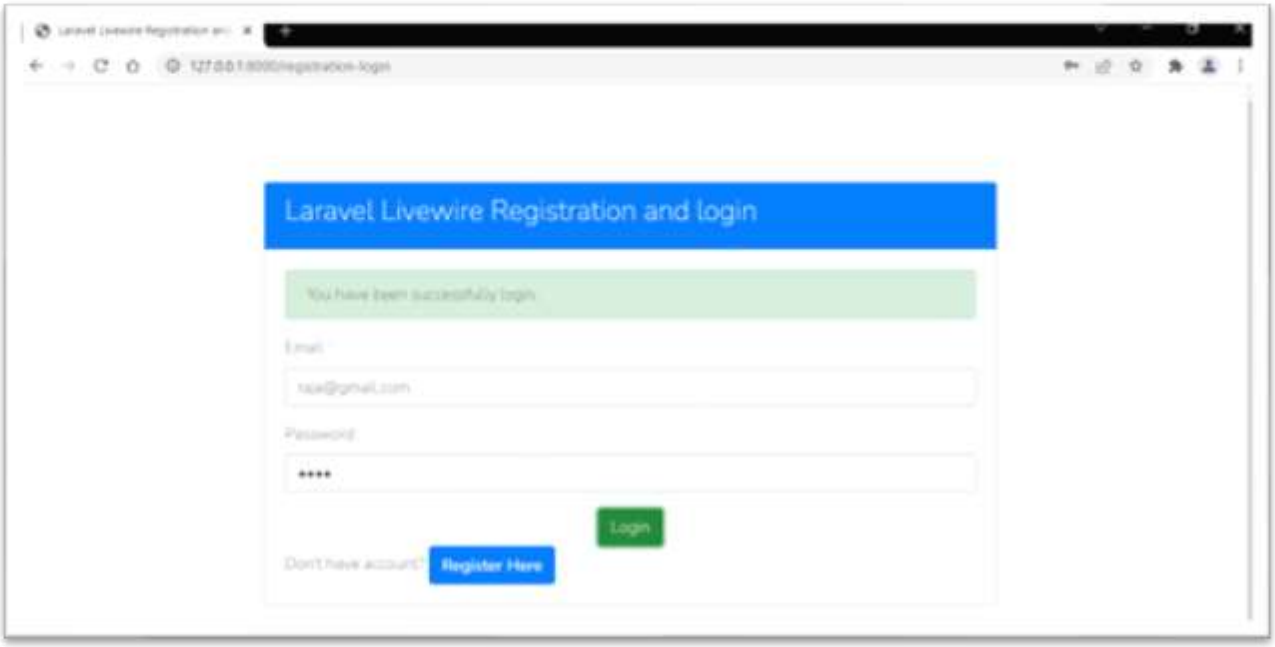
Laravel Livewire Registration and login

You have been successfully registered.

Name :

Email :

Password :


Laravel Livewire Registration an
127.0.0.1:8000/registration-login

Laravel Livewire Registration and login

You have been successfully login.

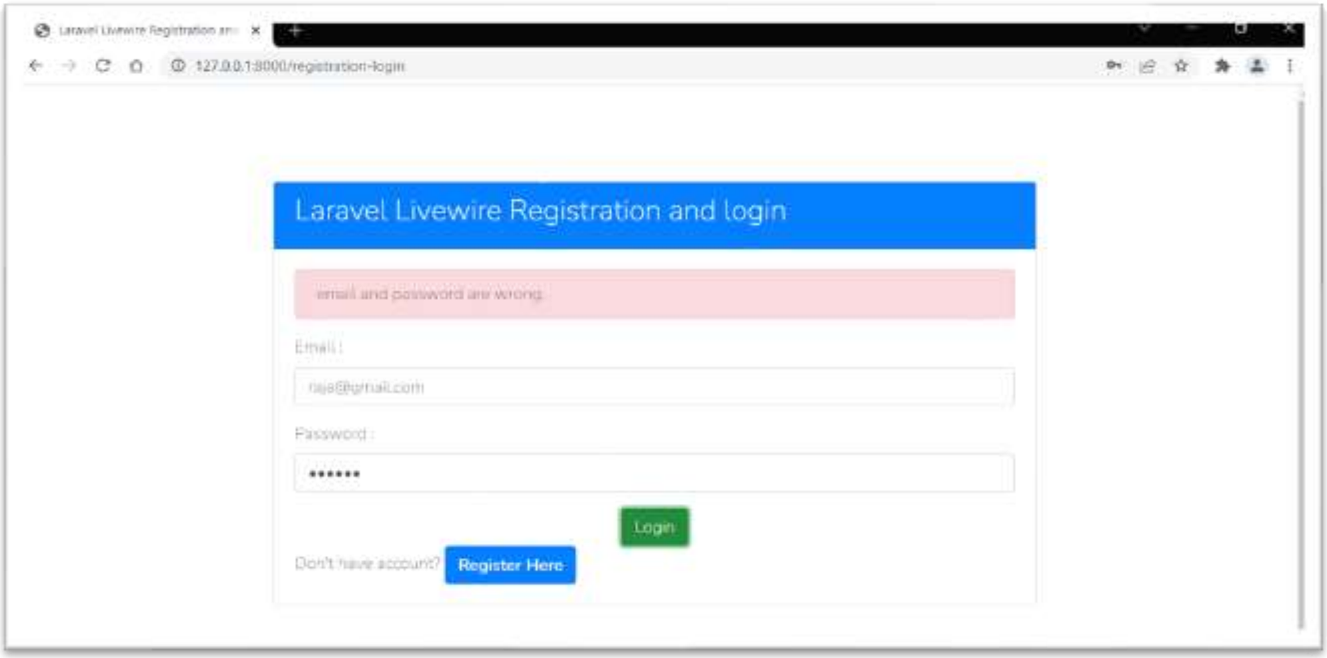
Email :

raja@gmail.com

Password :

••••

Don't have account?



127.0.0.1:8000/registration-login

Laravel Livewire Registration and login

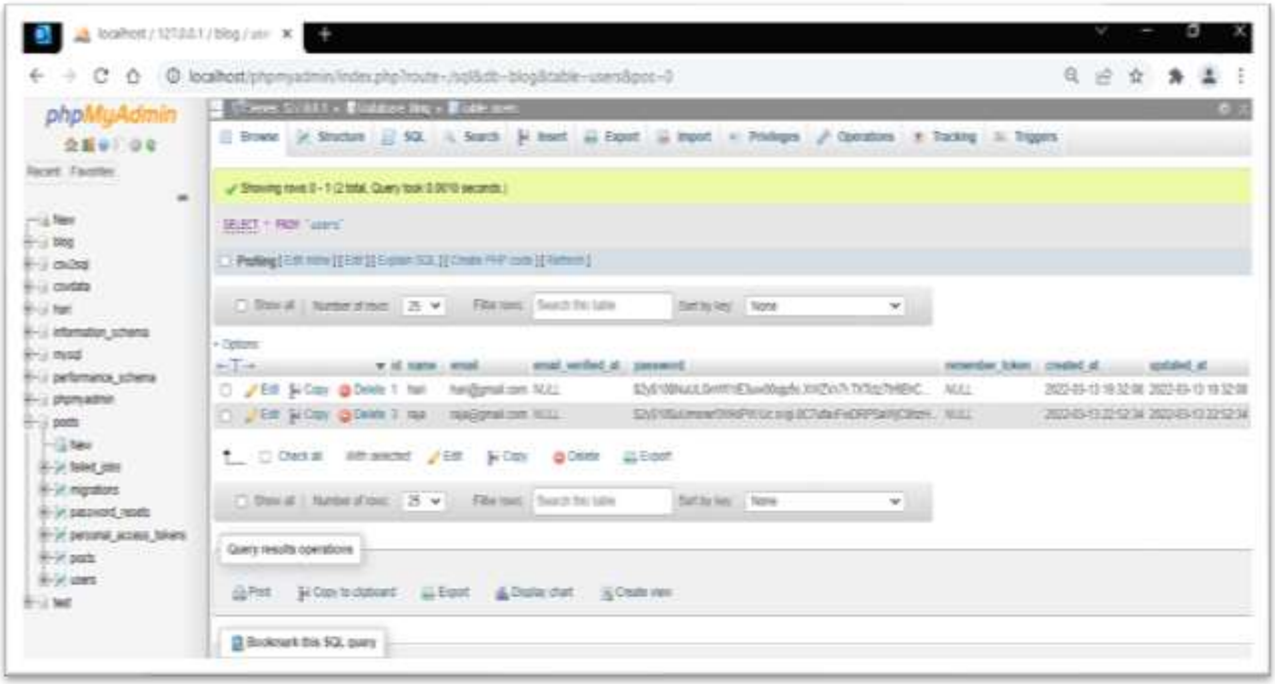
email and password are wrong

Email :

raja@gmail.com

Password :

•••••••



localhost/phpmyadmin/index.php?route=/sql&db=blog&table=users&pos=0

✓ Showing rows 0 - 1 (2 total, Query took 0.0010 seconds.)

```
SELECT * FROM `users`
```

id	name	email	email_verified_at	password	remember_token	created_at	updated_at
1	hari	hari@gmail.com	NULL	[illegible]	NULL	2022-03-13 18:32:08	2022-03-13 18:32:08
3	raja	raja@gmail.com	NULL	[illegible]	NULL	2022-03-13 22:52:34	2022-03-13 22:52:34

References:

- https://www.nicesnippets.com/blog/laravel-livewire-login-register-example-tutorial