

RAJALAKSHMI ENGINEERING COLLEGE
[AUTONOMOUS]
RAJALAKSHMI NAGAR, THANDALAM - 602105



Laboratory Record Note Book

Name :

Year / Branch / Section :

Register No :

College Roll No :

Semester :

Academic Year :

RAJALAKSHMI ENGINEERING COLLEGE
RAJALAKSHMI NAGAR, THANDALAM - 602105

BONAFIDE CERTIFICATE

Name : _____

Academic Year : _____ Semester : _____ Branch : _____

Register No :

Certified that is the bonafide record of work done by the

above student in the _____

Laboratory during the year 20 - 20

Signature of Faculty in-charge

Submitted for the practical examination held on _____

Internal Examiner

External Examiner

INDEX

Reg. No. : _____ Name : _____

Year : _____ Branch : _____ Sec : _____

| S. No. | Date | Title | Page No. | Teacher's Signature / Remarks |
|--------|------|--|----------|-------------------------------|
| 1 | | Create a web page to embed a map along with hot spot AND links | | |
| 2 | | Create a web page using an embedded, external, and inline CSS file | | |
| 3 | | Create a registration page along with validations | | |
| 4a | | JSP - Library Management System | | |
| 4b | | Servlet - Bank Application | | |
| 5 | | PHP – Employee Details to connect database and execute queries to retrieve and update data. Prepare report for single and group of employees based on the end user needs | | |
| 6 | | Bootstrap – Web Page | | |
| 7 | | Design a Web page with Navigation menu, Inline editor, Order form, Instant Search & Switchable Grid. | | |
| 8 | | Angular JS – Single Page Application | | |

EXPT NO: Create a web page to embed a map along with hot spot AND links.

DATE:

AIM:

To create a web page which includes a map and display the related information when a hot spot is clicked in the map.

PROCEDURE:

1. Create a html file with map tag.
2. Set the source attribute of the img tag to the location of the image and also set the use map attribute.
3. Specify an area with name, shape and href set to the appropriate values.
4. Repeat step 3 as many hot spots you want to put in the map.
5. Create html files for each and every hot spot the user will select.

Code:

ImageMap.html

```
<HTML>
```

```
<HEAD>
```

```
<TITLE>Image Map</TITLE> </HEAD>
```

```
<BODY>
```

```
 <map  
name="metroid"
```

```
id="metroid">
```

```
<area href="TamilNadu.html" shape="circle" coords="208,606,50"  
title="TamilNadu"/>
```

```
<area href="Karnataka.html" shape="rect" coords = "130,531,164,535" title  
="Karnataka" />
```

```
<area href="AndhraPradesh.html" shape="poly" coords =
```

```
"227,490,238,511,230,536,198,535,202,503" title ="Andhra Pradesh" />
```

```
</BODY>
```

TamilNadu.html

```
<HTML><HEAD>
<TITLE>About Tamil Nadu</TITLE>
</HEAD>
<BODY>
<CENTER><H1>Tamil Nadu</H1></CENTER> <HR>
<UL>
<LI>Area : 1,30,058 Sq. Kms.</LI>
<LI>Capital : Chennai</LI>
<LI>Language : Tamil</LI>
<LI>Population : 6,21,10,839</LI> </UL><hr>
<a href='ImageMap.html'>India Map</a>
</BODY>
</HTML>
```

Karnataka.html

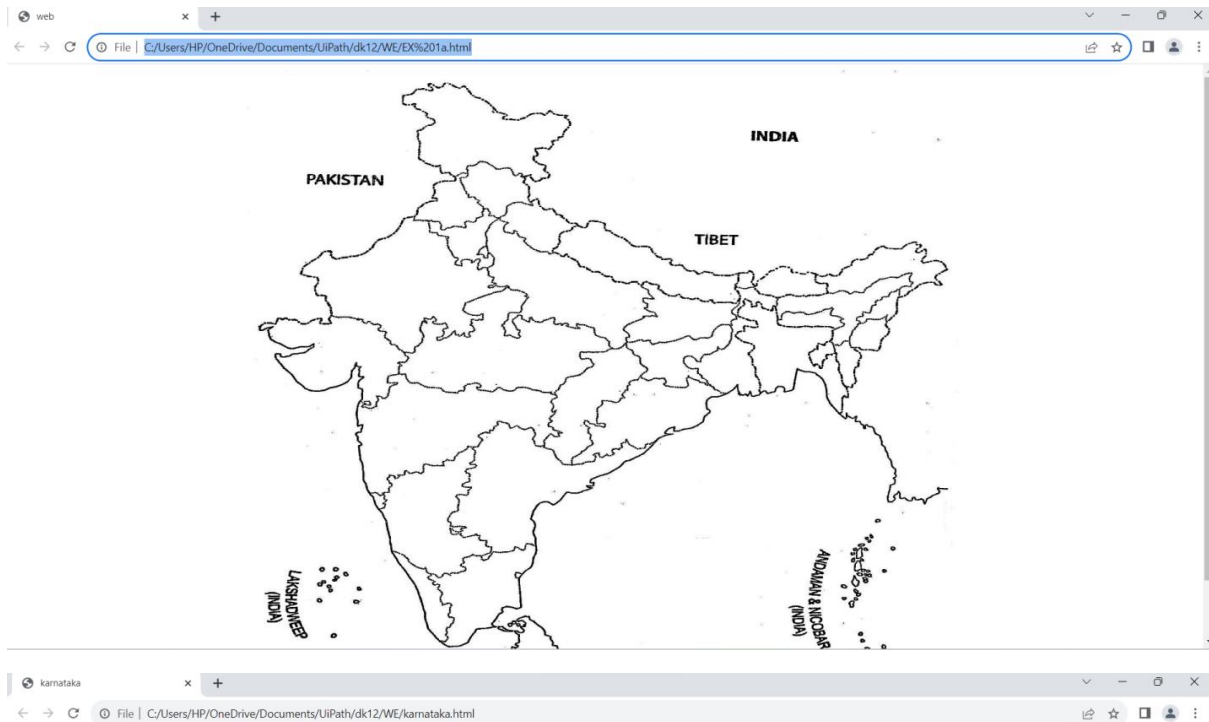
```
<HTML>
<HEAD>
<TITLE>About Karnataka</TITLE> </HEAD>
<BODY>
<CENTER><H1>Karnataka</H1></CENTER>
<HR>
5
<UL>
<LI>Area : 1,91,791 Sq. Kms</LI>
<LI>Capital : Bangalore</LI>
<LI>Language : Kannada</LI>
```

```
<LI>Population : 5,27,33,958</LI>
</UL>
<hr>
<a href='ImageMap.html'>India Map</a>
</BODY>
</HTML>
```

AndhraPradesh.html

```
<HTML>
<HEAD>
<TITLE>About Andhra Pradesh</TITLE> </HEAD>
<BODY>
<CENTER><H1>Andhra Pradesh</H1></CENTER> <HR>
<UL>
<LI>Area : 2,75,068 Sq. Kms</LI>
<LI>Capital : Hyderabad</LI>
<LI>Language : Telugu</LI>
</UL>
<hr>
<a href='ImageMap.html'>India Map</a>
</BODY>
</HTML>
```

Output:



BEST TOURIST PLACES IN KARNATAKA



Karnataka is a state in southwest India with Arabian Sea coastlines. The capital, Bengaluru (formerly Bangalore), is a high-tech hub known for its shopping and nightlife. To the southwest, Mysore is home to lavish temples including Mysore Palace, former seat of the region's maharajas. Hampi, once the medieval Vijayanagara empire's capital, contains ruins of Hindu temples, elephant stables and a stone chariot.

RESULT:

Thus the creation of a web page which includes a map and display the related in-formation when a hot spot is clicked in the map was executed successfully.

EXPT NO: Create a web page using an embedded, external, and inline CSS file.

DATE:

AIM:

To create a web page that displays college information using various style sheet.

PROCEDURE:

1. Create a web page with frame sets consisting two frames
2. In the first frame include the links
3. In the second frame set display the web page of the link
4. Create a external style sheets
5. Create a embedded style sheets
6. Create a inline and internal style sheets and make it link to the external style sheets

Code:

Index.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>MUSCLEWIKI</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      margin: 0;
      padding: 0;
    }
```



```
header {
    background-color: #38004a;
    color: #ffffff;
    padding: 20px;
    text-align: center;
} nav {
    background-color: #525252;
    padding: 10px;
}
nav ul {
    list-style-type: none;
    margin: 0;
    padding: 0;
    text-align: center;
}
nav ul li {
    display: inline;
    margin-right: 20px;
}
nav ul li a {
    color: #ffffff;
    text-decoration: none;
}
nav ul li a:hover {
    text-decoration: underline;
}
main {
```

```
padding: 20px;
}
footer {
background-color: #000000;
color: #ffffff;
padding: 10px;
text-align: center;
position: fixed;
width: 100%;
bottom: 0;
}
</style>
</head>
<body>
<header>
<h1>MUSCLE WIKI</h1>
</header>
<nav>
<ul>
<li><a href="EX 2.html">HOME</a></li>
<li><a href="UPPER.html">UPPERBODY</a></li>
<li><a href="LOWER.html">LOWEBODY</a></li>
<li><a href="ABS.html">ABS</a></li>
</ul>
</nav>
<main>
<h2 align="center">PUSH YOUR LIMITS</h2>
```

<p align="center">The purpose of warming up before physical activity is to prepare mentally and physically for your chosen activity.

Warming up increases your heart rate and therefore your blood flow. This enables more oxygen to reach your muscles.

A warm-up also activates and primes the connections between your nerve and muscles, which improves the efficiency of movement. Additionally, your range of motion (flexibility) should be increased by dynamic stretching.</p>

<div align="center">

<iframe width="560" height="315" src="https://www.youtube.com/embed/VecbXgWY0DI?si=7XsScGR6Tk_3-0Ht" title="YouTube video player" title="YouTube video player" frameborder="0" allow="accelerometer; autoplay; clipboard-write; encrypted-media; gyroscope; picture-in-picture; web-share" allowfullscreen></iframe>

</div>

</main>

<footer>

</footer>

</body>

</html>

Upperbody:

<main>

<h2 align="center">UPPERBODY</h2>

<p align="center">When building your upper body workouts, you'll want to keep in mind what stage of your training you are in. If you are a true beginner who lacks any sort of stability required to perform some of the listed exercises above, you won't be very successful with a more advanced training program.

Below, we're going to lay it all out for you. You'll have an upper body focused program for whatever phase of training you are in. If you are a beginner, the workouts below can be used as a form of progression from one to the next.

We also incorporated lower body workouts into the routines to ensure you have an aesthetic physique. You might even notice you get stronger on your upper body lifts as you get stronger on your lower body workout days.</p>

```
<div align="center">

    <iframe                                width="560"                height="315"
src="https://www.youtube.com/embed/lBickDLoU8o?si=u_BHwvhDwpiafyNw
" title="YouTube video player" frameborder="0" allow="accelerometer;
autoplay; clipboard-write; encrypted-media; gyroscope; picture-in-picture; web-
share" allowfullscreen></iframe>

</div>

</main>

<footer>

</footer>

</body>

</html>
```

Lowebody:

```
<main>

    <h2 align="center">LOWEBODY</h2>

    <p align="center">"Whether you're adding lower body strength training to
your routine to benefit other types of workouts, like cycling or running, you're
hoping to further define muscles in your lower body, or you want to feel stronger
in your everyday activities like climbing stairs and walking, there are many
benefits to lower-body workouts.</p>

    <div align="center">

        <iframe                                width="560"                height="315"
src="https://www.youtube.com/embed/2HnfQ58Ng_8?si=_8WO8IC7dCiLiThr"
title="YouTube video player" frameborder="0" allow="accelerometer; autoplay;
clipboard-write; encrypted-media; gyroscope; picture-in-picture; web-share"
allowfullscreen></iframe>

    </div>

</main>
```

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

Abdoman workout:

```
<main>

  <h2 align="center">ABDOMAN WORKOUTS</h2>

  <p align="center">We're all for planking your way to a stronger core. And
odds are you have a handful of go-to bodyweight abs exercises that you do on the
mat when you're cooling down.

  But if you've ever wondered how you can work your abs with the equipment
available at your gym, we've got you covered with 17 creative and effective ways
to take your abs routine up a notch.

  We worked with Equinox trainer Gerren Liles to round up killer core moves
using some of the most common gym equipment: a pull-up bar, a cable machine,
a Bosu ball, an abs roller, TRX straps, and a barbell. And you can adjust them to
fit any fitness level.

  Next time you're at the gym (and dread doing another crunch), pick three of
the moves below and do 2 to 3 sets for the recommended number of reps. Stick
to it, and we promise you'll feel the burn — and see results.</p>

  <div align="center">

    <iframe
      width="560"
      height="315"
      src="https://www.youtube.com/embed/fSc9n7C07FU?si=XWj5IQJM0_G9k4U
D" title="YouTube video player" frameborder="0" allow="accelerometer;
autoplay; clipboard-write; encrypted-media; gyroscope; picture-in-picture; web-
share" allowfullscreen></iframe>

  </div>

</main>

<footer>

  </footer>
```

</body>

</html>

Output:



PUSH YOUR LIMITS

The purpose of warming up before physical activity is to prepare mentally and physically for your chosen activity. Warming up increases your heart rate and therefore your blood flow. This enables more oxygen to reach your muscles. A warm-up also activates and primes the connections between your nerve and muscles, which improves the efficiency of movement. Additionally, your range of motion (flexibility) should be increased by dynamic stretching.



UPPERBODY

When building your upper body workouts, you'll want to keep in mind what stage of your training you are in. If you are a true beginner who lacks any sort of stability required to perform some of the listed exercises above, you won't be very successful with a more advanced training program. Below, we're going to lay it all out for you. You'll have an upper body focused program for whatever phase of training you are in. If you are a beginner, the workouts below can be used as a form of progression from one to the next. We also incorporated lower body workouts into the routines to ensure you have an aesthetic physique. You might even notice you get stronger on your upper body lifts as you get stronger on your lower body workout days.





LOWEBODY

"Whether you're adding lower body strength training to your routine to benefit other types of workouts, like cycling or running, you're hoping to further define muscles in your lower body, or you want to feel stronger in your everyday activities like climbing stairs and walking, there are many benefits to lower-body workouts.



ABDOMAN WORKOUTS

We're all for planking your way to a stronger core. And odds are you have a handful of go-to bodyweight abs exercises that you do on the mat when you're cooling down. But if you've ever wondered how you can work your abs with the equipment available at your gym, we've got you covered with 17 creative and effective ways to take your abs routine up a notch. We worked with Equinox trainer Gerren Liles to round up killer core moves using some of the most common gym equipment: a pull-up bar, a cable machine, a Bosu ball, an abs roller, TRX straps, and a barbell. And you can adjust them to fit any fitness level. Next time you're at the gym (and dread doing another crunch), pick three of the moves below and do 2 to 3 sets for the recommended number of reps. Stick to it, and we promise you'll feel the burn — and see results.



RESULT:

Thus the creation of a web page that displays college information using various style sheet was successfully executed and verified.

EXPT NO: Create a registration page along with validations.

DATE:

Aim:

To create a visually appealing registration form with validation for email addresses.

Procedure:

1. Layout Design:

Design the layout of the registration form, including input fields for username, email, and password, along with a submit button.

2. Styling:

Apply CSS to style the form elements, providing appropriate spacing, alignment, and background color to enhance visual appeal.

3. Email Validation:

Implement JavaScript to validate the email address entered by the user. Ensure that the email follows the standard format and contains the "@" symbol.

4. Error Handling:

Display error messages if the email entered by the user is invalid. These messages should provide clear guidance on how to correct the input.

5. Submission Handling:

Handle form submission events, ensuring that the form data is submitted only if all fields are filled correctly, including a valid email address. If any field is invalid, prevent form submission and prompt the user to correct the errors.

Code:**Index.html:**

```
<!DOCTYPE html>
<html lan="en">
<head>
<title>Regform</title>
</head>
<style>
.registration-form {
width: 100%;
max-width: 400px;
margin: 0 auto;
}
.registration-form input[type="text"], .registration-form input[type="email"],
.registration-form input[type="password"] {
width: 100%;
padding: 15px;
margin-bottom: 10px;
border-radius: 5px;
border: 1px solid #ccc;
}
.registration-form input[type="submit"] {
width: 100%;
padding: 15px;
background-color: #4CAF50;
color: white;
border: none;
```

```

border-radius: 5px;
cursor: pointer;
}
@media (max-width: 600px) {
.registration-form input[type="text"], .registration-form
input[type="email"], .registration-form input[type="password"] {
width: 100%;
}
}
p{
text-align: center;
font-weight:bold ;
font-size: larger;
color:blue;
}
</style>
<body>
<form class="registration-form"><p>Registration Form</p>
<label for="first"><b>First Name:</b></label>
<input type="text" name="first" placeholder="First Name" required>
<label for="last"><b>Last Name:</b></label>
<input type="text" name="last" placeholder="Last Name" required>
<label for="dept"><b>Department:</b></label>
<input type="text" name="dept" placeholder="Department" required>
<label for="email"><b>Email Id:</b></label>
<input type="email" name="email" placeholder="Email" required>
<label for="pass"><b>PassWord:</b></label>

```

```
<input type="password" name="pass" placeholder="Password" required>
<input type="checkbox" required> I agree to the terms and conditions.
<br><br><br>
<input type="submit" value="Register">
</form>
</body>
</html>
```

Output:

Registration Form

First Name:

Last Name:

Department:

Email Id:

PassWord:

☒ I agree to the terms and conditions.

Registration Form

First Name:

Last Name:

Department:

Email Id:

PassWord:

☒ I agree to the terms and conditions.

Result:

A visually appealing registration form with email validation ensures user inputs adhere to standard email format, enhancing data accuracy and user experience.

EXPT NO:

JSP - LIBRARY MANAGEMENT SYSTEM

DATE:

Aim:

To develop a JavaScript program that validates the controls in the forms of the Library Management System application, ensuring data integrity and user input correctness.

Procedure:

1. Identify forms requiring validation in the Library Management System.
2. Define validation rules for each form field.
3. Develop JavaScript functions for validation based on defined rules.
4. Integrate validation functions with form submission processes.
5. Prevent form submission if data fails validation, display error messages.
6. Thoroughly test and debug the validation functionality for usability and reliability.

Code:

Index.html:

```
<!DOCTYPE html>
```

```
<html lang="en">
```

```
<head>
```

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

```
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
  <title>Library Management System</title>
```

```
  <link rel="stylesheet" href="styles.css">
```

```
</head>
<body>
  <h2>Library Management System</h2>
  <form id="bookForm">
    <label for="title">Title:</label>
    <input type="text" id="title" name="title" required><br>
    <label for="author">Author:</label>
    <input type="text" id="author" name="author" required><br>
    <label for="year">Year:</label>
    <input type="number" id="year" name="year" required><br>
    <button type="submit">Add Book</button>
  </form>
  <table id="bookTable">
    <tr>
      <th>Title</th>
      <th>Author</th>
      <th>Year</th>
      <th>Action</th>
    </tr>
  </table>

  <script>
    const bookForm = document.getElementById('bookForm');
    const bookTable = document.getElementById('bookTable');

    bookForm.addEventListener('submit', function(event) {
```

```
event.preventDefault();

const title = document.getElementById('title').value;
const author = document.getElementById('author').value;
const year = document.getElementById('year').value;

addBook(title, author, year);
bookForm.reset();
});

function addBook(title, author, year) {
    const row = bookTable.insertRow(-1);
    const titleCell = row.insertCell(0);
    const authorCell = row.insertCell(1);
    const yearCell = row.insertCell(2);
    const actionCell = row.insertCell(3);

    titleCell.textContent = title;
    authorCell.textContent = author;
    yearCell.textContent = year;
    actionCell.innerHTML = '<button
onclick="editBook(this)">Edit</button>';
}

function editBook(button) {
    const row = button.parentElement.parentElement;
    const title = row.cells[0].textContent;
    const author = row.cells[1].textContent;
```

```
const year = row.cells[2].textContent;

const newTitle = prompt('Enter new title:', title);
const newAuthor = prompt('Enter new author:', author);
const newYear = prompt('Enter new year:', year);

if (newTitle && newAuthor && newYear) {
    row.cells[0].textContent = newTitle;
    row.cells[1].textContent = newAuthor;
    row.cells[2].textContent = newYear;
}
}
</script>
</body>
</html>
```

Style.css:

```
body {
    font-family: Arial, sans-serif;
    background-color: #f0f0f0;
    margin: 0;
    padding: 0;
}

.container {
    max-width: 800px;
    margin: 20px auto;
    padding: 20px;
```



```
background-color: #fff;
border-radius: 5px;
box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
}
```

```
h2 {
  text-align: center;
}
```

```
form {
  margin-bottom: 20px;
}
```

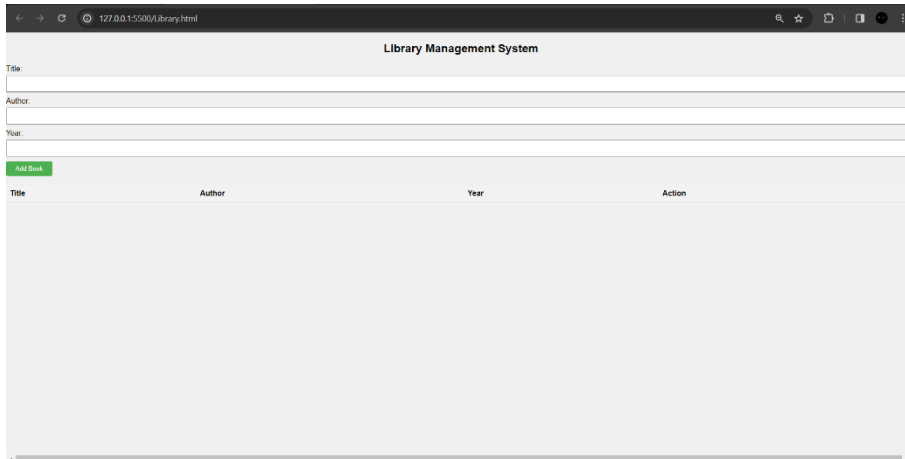
```
form label {
  display: block;
  margin-bottom: 5px;
}
```

```
form input[type="text"],
form input[type="number"] {
  width: calc(100% - 12px);
  padding: 8px;
  margin-bottom: 10px;
}
```

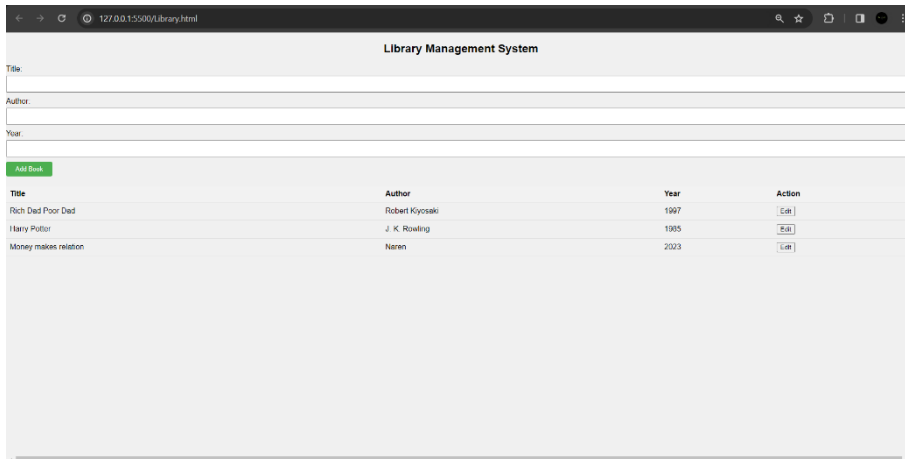
```
form button[type="submit"] {
  padding: 8px 20px;
  background-color: #4CAF50;
  color: white;
  border: none;
  border-radius: 4px;
  cursor: pointer;
```

```
        transition: background-color 0.3s;
    }
    form button[type="submit"]:hover {
        background-color: #45a049;
    }
    table {
        width: 100%;
        border-collapse: collapse;
    }
    th, td {
        padding: 8px;
        border-bottom: 1px solid #ddd;
        text-align: left;
    }
    th {
        background-color: #f2f2f2;
    }
    tr:hover {
        background-color: #f5f5f5;
    }
```

Output:

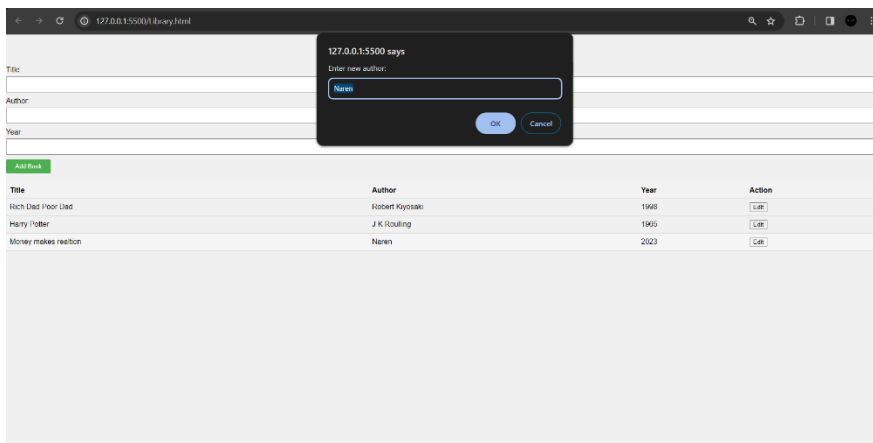


A browser window showing a web application titled "Library Management System". The address bar displays "127.0.0.1:5500/Library.html". The form contains three input fields: "Title:", "Author:", and "Year:". Below these fields is a green button labeled "Add Book". Underneath the button is a table with four columns: "Title", "Author", "Year", and "Action". The table is currently empty.



The same browser window showing the "Library Management System". The "Title:", "Author:", and "Year:" input fields are now populated with "Rich Dad Poor Dad", "Robert Kiyosaki", and "1997" respectively. The "Add Book" button remains green. The table below now contains three rows of data:

| Title | Author | Year | Action |
|----------------------|-----------------|------|----------------------|
| Rich Dad Poor Dad | Robert Kiyosaki | 1997 | Edit |
| Harry Potter | J. K. Rowling | 1995 | Edit |
| Money makes relation | Naren | 2023 | Edit |



The same browser window showing the "Library Management System". A modal dialog box is open in the center of the screen. The dialog has a title "127.0.0.1:5500 says" and a message "Enter new author:". Below the message is a text input field containing the name "Naren". At the bottom of the dialog are two buttons: "OK" and "Cancel". The background form is slightly dimmed.

Result:

Developed JavaScript program to validate Library Management System forms, ensuring data integrity and input correctness.

EXPT NO:

PHP – Employee Details

DATE:

Aim:

PHP program for Employee Details, which includes EmpID, Name, Designation, Salary, DOJ, etc., to connect with the database and execute queries to retrieve and update data.

Procedure:

Relations using MYSQL for a banking application given below enforcing primary key and

foreign key constraints:

EMPDETAILS (EMPID, ENAME, DESIG, DEPT, DOJ, SALARY)

1. Open MySQL.

2. Create a database.

```
mysql> create database rec;
```

Query OK, 1 row affected (0.05 sec)

3. Connect to the database.

```
mysql> use rec;
```

Database changed

4. Create the following tables:

```
mysql> create table empdetails(empid int primary key,
```

```
-> ename varchar(20), desig varchar(20), dept varchar(20),
```

```
-> doj date, salary int);
```

Query OK, 0 rows affected (0.08 sec)

PROGRAM:

config.php

```
<?php
$databaseHost = 'localhost';
$databaseName = 'rec';
$databaseUsername = 'root';
$databasePassword = 'admin';
$mysqli = mysqli_connect($databaseHost, $databaseUsername,
$databasePassword, $databaseName);
?>
```

index.php

```
<?php
//including the database connection file
include_once("config.php");
//fetching data in descending order (lastest entry first)
$result=mysqli_query($mysqli, "SELECT * FROM empdetails ORDER BY
empid DESC");
?>
<html>
<head>
<title>Homepage</title>
</head>
<body>
<h1 align="center">Employee Details</h1>
<hr />
<a href="add.html">Add New Data</a><br/><br/>
<table width='100%' border=0>
<tr bgcolor='#CCCCCC'>
```

```
<td>Employee Id.</td>
<td>Name</td>
<td>Designation</td>
<td>Department</td>
<td>DOJ</td>
<td>Salary</td>
<td>Edit / Delete</td>
</tr>

<?php
while($res = mysqli_fetch_array($result)) {echo
"<tr>";
echo "<td>".$res['empid']."</td>";
echo "<td>".$res['ename']."</td>";
echo "<td>".$res['desig']."</td>";
echo "<td>".$res['dept']."</td>";
echo "<td>".$res['doj']."</td>";
echo "<td>".$res['salary']."</td>";
echo "<td><a href='edit.php?empid=$res[empid]'>Edit</a>";
echo " | <a href='delete.php?empid=$res[empid]'>Delete</a></td>";echo
"</tr>";
}
?>

</table>

</body>

</html>
```

add.html

```
<html>
<head>
<title>Add Employee Details</title>
</head>
<body>
<h1 align="center">Add Employee Details</h1>
<hr />
<a href="index.php">Home</a>
<br /><br />
<form action="add.php" method="post" name="form1">
<table width="25%" border="0">
<tr>
<td>Employee Id. : </td>
<td><input type="text" name="empid"></td>
</tr>
<tr>
<td>Name : </td>
<td><input type="text" name="ename"></td>
</tr>
<tr>
<td>Designation : </td>
<td><input type="text" name="desig"></td>
</tr>
<tr>
<td>Department</td>
<td><input type="text" name="dept"></td>
```

```
</tr>

<tr>
<td>DOJ</td>
<td><input type="text" name="doj"></td>

</tr>

<tr>
<td>Salary</td>
<td><input type="text" name="salary"></td>
</tr>

<tr>
<td colspan="2" align="center"><input type="submit"
name="Submit" value="Add"></td>
</tr>
</table>
</form>
</body>
</html>
add.php
<html>
<head>
<title>Add Employee Details</title>
</head>
<body>
<?php
//including the database connection file
include_once("config.php");
```



```

$empid = $_POST['empid'];
$name = $_POST['ename'];
$desig = $_POST['desig'];
$dept = $_POST['dept'];
$doj = $_POST['doj'];
$salary = $_POST['salary'];
if(isset($_POST['Submit'])) {
//insert data to database
$result = mysqli_query($mysqli, "INSERT INTO empdetails values ($empid,
'$name','$desig','$dept','$doj','$salary)");
//display success message
echo "<h1 align='center'>Add Employee Details</h1>";echo
"<hr />";
echo "<font color='green'>Data added successfully.";echo
"<br/><a href='index.php'>View Result</a>";
}
?>
</body>
</html>
edit.php
<?php
// including the database connection file
include_once("config.php");
if(isset($_POST['update']))
{
$empid = $_POST['empid'];
$name = $_POST['ename'];

```

```

$desig = $_POST['desig'];
$dept = $_POST['dept'];
$doj = $_POST['doj'];
$salary = $_POST['salary'];
//updating the table
$result = mysqli_query($mysqli, "UPDATE empdetails SET ename='$ename',
desig='$desig',dept='$dept',doj='$doj',salary=$salary WHERE empid=$empid");
//redirectig to the display page. In our case, it is index.phpheader("Location:
index.php");
}
?>
<?php
echo "<h1 align='center'>Edit Employee Details</h1>";echo "<hr
/>";
//getting id from url
$empid = $_GET['empid'];
//selecting data associated with this particular eid
$result = mysqli_query($mysqli, "SELECT * FROM empdetails WHERE
empid=$empid");
while($res = mysqli_fetch_array($result))
{
$empid = $res['empid'];
$ename = $res['ename'];
$desig = $res['desig'];
$dept = $res['dept'];
$doj = $res['doj'];

```

```
$salary = $res['salary'];
}
?>

<html>
<head>
<title>Edit Employee Details</title>
</head>
<body>
<a href="index.php">Home</a>
<br/><br/>
<form name="empform" method="post" action="edit.php">
<table border="0">
<tr>
<td>Name : </td>
<td><input type="text" name="ename" value="<?php echo $ename;?>"></td>
</tr>
<tr>
<td>
<tr>
<td>Designation : </td>
<td><input type="text" name="desig" value="<?php echo $desig;?>"></td>
</tr>
<tr>
<td>Department : </td>
<td><input type="text" name="dept" value="<?php echo $dept;?>"></td>
</tr>

<td>DOJ : </td>
```

```

<td><input type="text" name="doj" value="<?php echo $doj;?>"></td>
</tr>
<tr>
<td>Salary</td>
<td><input type="text" name="salary" value="<?php echo $salary;?>"></td>
</tr>
<tr>
<td><input type="hidden" name="empid" value=<?php echo
$_GET['empid'];?>></td>
<td><input type="submit" name="update" value="Update"></td>
</tr>
</table>
</form>
</body>
</html>

```

delete.php

```

<?php
//including the database connection file
include("config.php");
//getting id of the data from url
$empid = $_GET['empid'];
//deleting the row from table
$result = mysqli_query($mysqli, "DELETE FROM empdetails WHERE
empid=$empid");
//redirecting to the display page (index.php in our case)
header("Location:index.php");
?>

```

Output:

Add Employee Details

[Home](#)

Employee Id. :

Name :

Designation :

Department :

DOJ :

Salary :

Add Employee Details

[Home](#)

Employee Id. :

Name :

Designation :

Department :

DOJ :

Salary :



Add Employee Details

Data added successfully.

[View Result](#)

Result:

Hence, PHP program for Employee Details, which includes EmpID, Name, Designation, Salary, DOJ, etc., to connect with the database and execute queries to retrieve and update data is executed successfully.

EXPT NO: Servlet - Bank Application

DATE:

Aim:

Program to develop a Banking application accessing a database using Servlet.

Procedure:

Relations using MYSQL for a banking application given below enforcing primary key and

foreign key constraints:

CUSTOMER (CID, CNAME)

ACCOUNT (ANO, ATYPE, BALANCE, CID)

An account can be a savings account or a current account. Check ATYPE in 'S' or 'C'.

A customer can have both types of accounts.

TRANSACTION (TID, ANO, TTYPE, TDATE, TAMOUNT)

TTYPE can be 'D' or 'W'

D- Deposit; W – Withdrawal

1. Open MySQL.

2. Create a database.

```
mysql> create database banking;
```

Query OK, 1 row affected (0.05 sec)

3. Connect to the database.

```
mysql> use banking;
```

Database changed

4. Create the following tables:

```
mysql> create table customer (cid integer, cname varchar(20),
```

```
-> primary key (cid));
```

index.html:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Banking Application</title>
</head>
<body>
<h1 align="center">Banking Application</h1>
<hr />
<a href="Customer.html">Customer Details</a>
<br />
<br />
<a href="Account.html">Account Details</a>
</body>
</html>
```

Customer.html:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Customer Details</title>
</head>
<body>
<h1 align="center">Customer Details</h1>
<hr />
<form action="AddCustomer" method="post">
```

```

<table>
<tr>
</tr>
<tr>
</tr>
<tr>
</tr>
<tr>
</tr>
</table>
<td>Customer Id. :</td>
<td><input type="text" name="cid"></td>
<td>Customer Name :</td>
<td><input type="text" name="cname"></td>
<td colspan="2" align="center"><input type="submit"
value="Add Customer"></td>
<br /> <a href="ViewCustomers">View All Customers</a>
</form>
</body>
</html>

```

AddCustomer.java:

```

import java.io.IOException; import
java.io.PrintWriter; import
java.sql.Connection; import
java.sql.DriverManager;
import java.sql.PreparedStatement;
import javax.servlet.ServletException; import
javax.servlet.annotation.WebServlet; import
javax.servlet.http.HttpServlet; import

```



```

javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
/**
 * Servlet implementation class AddCustomer
 */
@WebServlet("/AddCustomer")
public class AddCustomer extends HttpServlet { private static
final long serialVersionUID = 1L;Connection conn = null;
PreparedStatement ps = null;
/**
 * @see HttpServlet#doPost(HttpServletRequest request,
HttpServletResponse
 * response)
 */
protected void doPost(HttpServletRequest request,
HttpServletResponse response)
throws ServletException, IOException {
// TODO Auto-generated method stub
response.setContentType("text/html");
PrintWriter out = response.getWriter();
out.println("<html>");
out.println("<head><title>Add Customer Details</title></head>");
out.println("<body>");
out.println("<h1 align='center'>Add Customer Details</h1>");
out.println("<hr />");
try {
Class.forName("com.mysql.cj.jdbc.Driver");
String URL = "jdbc:mysql://localhost:3306/banking";

```

```
conn = DriverManager.getConnection(URL, "root", "admin");
ps = conn.prepareStatement("insert customer values (?, ?)"); ps.setInt(1,
Integer.parseInt(request.getParameter("cid"))); ps.setString(2,
request.getParameter("cname"));
int res = ps.executeUpdate();
if (res != 0)
out.println("Customer Details Inserted
Successfully...");
```

```
else
out.println("Customer Details Insertion Failure...");
ps.close();
conn.close();
} catch (Exception e) {
out.println(e);
}
out.println("<br />");
out.println("<a href='Customer.html'>Back</a>");
out.println("</body></html>");
}
}
```

ViewCustomers.java:

```
import java.io.IOException; import
java.io.PrintWriter; import
java.sql.Connection; import
java.sql.DriverManager;
import java.sql.PreparedStatement;
```

```

import java.sql.ResultSet;
import javax.servlet.ServletException; import
javax.servlet.annotation.WebServlet; import
javax.servlet.http.HttpServlet; import
javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
/**
 * Servlet implementation class ViewCustomers
 */
@WebServlet("/ViewCustomers")
public class ViewCustomers extends HttpServlet { private static
final long serialVersionUID = 1L;Connection conn = null;
PreparedStatement ps = null;
ResultSet rs = null;
/**
 * @see HttpServlet#doPost(HttpServletRequest request,
HttpServletResponse
 * response)
 */
protected void doGet(HttpServletRequest request,
HttpServletResponse response)
throws ServletException, IOException {
// TODO Auto-generated method stub
response.setContentType("text/html"); PrintWriter
out = response.getWriter(); out.println("<html>");
out.println("<head><title>View All Customer
Details</title></head>"); out.println("<body>");
out.println("<h1 align='center'>View All Customer Details</h1>");

```

```
out.println("<hr />");

try {
    Class.forName("com.mysql.cj.jdbc.Driver");
    String URL = "jdbc:mysql://localhost:3306/banking";
    conn = DriverManager.getConnection(URL, "root", "admin"); ps =
    conn.prepareStatement("select * from customer order bycid");
    rs = ps.executeQuery(); out.println("<table
border='1'>"); out.println("<tr>");
    out.println("<td>Customer Id.</td>");
    out.println("<td>Customer Name</td>");
    out.println("<td>Edit</td>");
    out.println("<td>Delete</td>");
    out.println("</tr>");
    while (rs.next()) {
        out.println("<tr>");
        out.println("<td>" + rs.getInt("cid") + "</td>"); out.println("<td>"
+ rs.getString("cname") + "</td>");out.println("<td><a
href='EditCustomer?cid=" + rs.getInt("cid") + "'>Edit</a></td>");
        out.println("<td><a href='DeleteCustomer?cid=" + rs.getInt("cid")
+ "'>Delete</a></td>"); out.println("</tr>");
    }
    out.println("</table>");
    ps.close();
    conn.close();
} catch (Exception e) {
    out.println(e);
}
```

```

    }
    out.println("<br />");
    out.println("<a href='Customer.html'>Back</a>");
    out.println("</body></html>");
    }
}

```

EditCsutomer.java:

```

import java.io.IOException; import
java.io.PrintWriter; import
java.sql.Connection; import
java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import javax.servlet.ServletException; import
javax.servlet.annotation.WebServlet; import
javax.servlet.http.HttpServlet; import
javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
/**
 * Servlet implementation class EditCustomer
 */
@WebServlet("/EditCustomer")
public class EditCustomer extends HttpServlet { private static
final long serialVersionUID = 1L;Connection conn = null;

PreparedStatement ps = null;
ResultSet rs = null;

```

```

/**
 * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse
 * response)
 */

protected void doGet(HttpServletRequest request,
HttpServletResponse response)
throws ServletException, IOException {
// TODO Auto-generated method stub
response.setContentType("text/html");
PrintWriter out = response.getWriter();
out.println("<html>");
out.println("<head><title>Edit Customer Details</title></head>");
out.println("<body>");
out.println("<h1 align='center'>Edit All Customer Details</h1>");
out.println("<hr />");
try {
Class.forName("com.mysql.cj.jdbc.Driver");
String URL = "jdbc:mysql://localhost:3306/banking";
conn = DriverManager.getConnection(URL, "root", "admin");ps =
conn.prepareStatement("select * from customer where cid = ?");
ps.setInt(1,Integer.parseInt(request.getParameter(("cid"))));rs =
ps.executeQuery();
rs.next();
out.println("<form action='UpdateCustomer' method='post'>");
out.println("<table>");
out.println("<tr>"); out.println("<td>Customer Id.
:</td>");
out.println("<td><input type='text' name='cid' value='" +rs.getInt("cid")

```

```

+ " readonly></td>"); out.println("</tr>");
out.println("<tr>");
out.println("<td>Customer Name :</td>"); out.println("<td><input
type='text' name='cname' value='" +rs.getString("cname") + "'></td>");
out.println("</tr>");
out.println("<tr>");
out.println("<td colspan='2' align='center'><input type='submit'
value='Update Customer'></td>"); out.println("</tr>");
out.println("</table>");
out.println("</form>");
ps.close();
conn.close();
} catch (Exception e) {
out.println(e);
}
out.println("<br />");
out.println("<a href='ViewCustomers'>Back</a>");
out.println("</body></html>");
}
}

```

UpdateCustomer.java:

```

import java.io.IOException; import

java.io.PrintWriter; import
java.sql.Connection; import
java.sql.DriverManager;
import java.sql.PreparedStatement;

```

```

import javax.servlet.ServletException; import
javax.servlet.annotation.WebServlet; import
javax.servlet.http.HttpServlet; import
javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
/**
 * Servlet implementation class UpdateCustomer
 */
@WebServlet("/UpdateCustomer")
public class UpdateCustomer extends HttpServlet { private static
final long serialVersionUID = 1L;Connection conn = null;
PreparedStatement ps = null;
/**
 * @see HttpServlet#doPost(HttpServletRequest request,
HttpServletResponse response)
 */
protected void doPost(HttpServletRequest request,
HttpServletResponse response)
throws ServletException, IOException {

// TODO Auto-generated method stub
response.setContentType("text/html");
PrintWriter out = response.getWriter();
out.println("<html>");
out.println("<head><title>Update Customer Details</title></head>");
out.println("<body>");
out.println("<h1 align='center'>Update Customer Details</h1>");
out.println("<hr />");

```



```

try {
    Class.forName("com.mysql.cj.jdbc.Driver");
    String URL = "jdbc:mysql://localhost:3306/banking";
    conn = DriverManager.getConnection(URL, "root", "admin"); ps =
    conn.prepareStatement("update customer set cname = ?where cid = ?");
    ps.setString(1, request.getParameter("cname")); ps.setInt(2,
    Integer.parseInt(request.getParameter("cid")));int res =
    ps.executeUpdate();
    if (res != 0)
        out.println("Customer Details Updated
        Successfully...");
    else
        out.println("Customer Details Updation Failure...");
    ps.close();
    conn.close();
} catch (Exception e) {
    out.println(e);
}
out.println("<br />");
out.println("<a href='ViewCustomers'>Back</a>");
out.println("</body></html>");

}

}

```

DeleteCustomer.java:

```

import java.io.IOException; import
java.io.PrintWriter; import

```

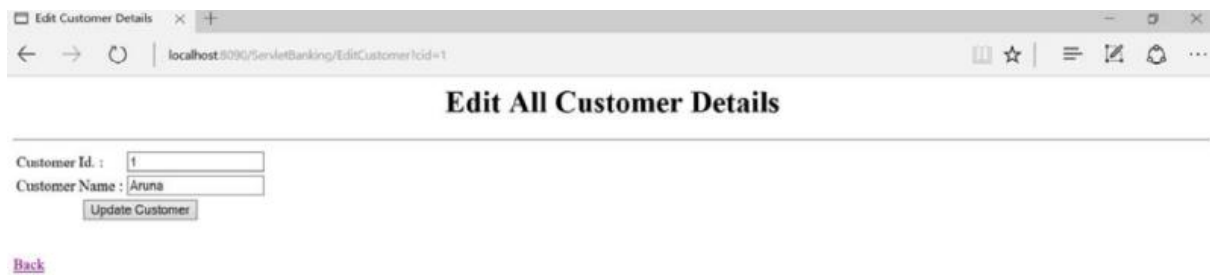
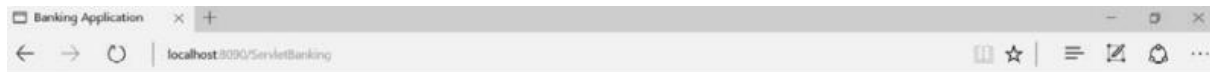
```
java.sql.Connection; import
java.sql.DriverManager;
import java.sql.PreparedStatement;
import javax.servlet.ServletException; import
javax.servlet.annotation.WebServlet; import
javax.servlet.http.HttpServlet; import
javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
/**
 * Servlet implementation class DeleteCustomer
 */
@WebServlet("/DeleteCustomer")
public class DeleteCustomer extends HttpServlet { private static
final long serialVersionUID = 1L;Connection conn = null;
PreparedStatement ps = null;
/**
 * @see HttpServlet#doGet(HttpServletRequest request,
HttpServletResponse response)
 */
protected void doGet(HttpServletRequest request,
HttpServletResponse response)
throws ServletException, IOException {

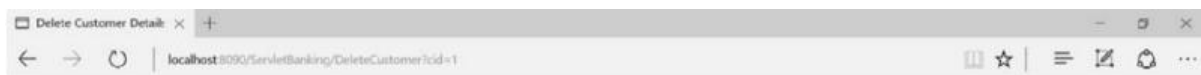
// TODO Auto-generated method stub
response.setContentType("text/html");
PrintWriter out = response.getWriter();
out.println("<html>");
out.println("<head><title>Delete Customer Details</title></head>");
```

```
out.println("<body>");
out.println("<h1 align='center'>Delete Customer Details</h1>");
out.println("<hr />");
try {
    Class.forName("com.mysql.cj.jdbc.Driver");
    String URL = "jdbc:mysql://localhost:3306/banking";
    conn = DriverManager.getConnection(URL, "root", "admin");
    ps = conn.prepareStatement("delete from customer where cid =
?");
    ps.setInt(1, Integer.parseInt(request.getParameter("cid")));
    int res = ps.executeUpdate();
    if (res != 0)
        out.println("Customer Details Deleted uccessfully...");
    else
        out.println("Customer Details Deletion Failure...");
    ps.close();
    conn.close();
} catch (Exception e) {
    out.println(e);
}
out.println("<br />");

out.println("<a href='ViewCustomers'>Back</a>");
out.println("</body></html>");
}
}
```

Output:





Delete Customer Details

Customer Details Deleted successfully...

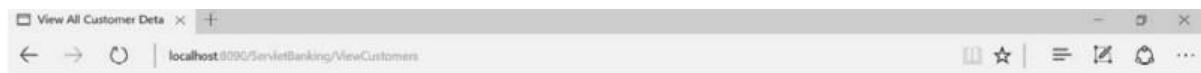
[Back](#)



Update Customer Details

Customer Details Updated Successfully...

[Back](#)



View All Customer Details

| Customer Id | Customer Name | Edit | Delete |
|-------------|---------------|------|--------|
|-------------|---------------|------|--------|

[Back](#)

Result:

Hence developed a Banking application accessing a database using Servlet is implemented successfully.

EXPT NO:

BOOTSTRAP – WEB PAGE

DATE:

Aim:

Program to develop an attractive web pages using Bootstrap.

Procedure:

1. File Setup:

- Create `index.html`, `styles.css`, and `scripts.js` files.
- Ensure they are in the same directory for easy access.

2. Bootstrap Integration:

- Link Bootstrap CSS in the HTML file using the CDN (Content Delivery Network)

```
<link  
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css"  
rel="stylesheet">
```

3. HTML Structure:

- Set up the basic structure of `index.html`.
- Organize sections such as "About Me," "Portfolio," and "Contact."

4. Content Integration:

- Add relevant content to each section.
- Include text, images, and links to showcase your work and skills.

5. Customization and Styling:

- Customize the appearance using custom CSS in `styles.css`.
- Adjust Bootstrap classes and add additional styles as needed for a personalized look.

PROGRAM:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>My Portfolio</title>
  <!-- Bootstrap CSS -->
  <link
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css"
rel="stylesheet">
  <!-- Font Awesome -->
  <link href="https://cdn.jsdelivr.net/npm/font-awesome@5.15.3/css/all.min.css" rel="stylesheet">
  <!-- Custom CSS -->
  <link href="styles.css" rel="stylesheet">
</head>
<body>
  <!-- Navigation -->
  <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
    <div class="container">
      <a class="navbar-brand" href="#">My Portfolio</a>
      <button class="navbar-toggler" type="button" data-toggle="collapse" data-
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 ml-auto">
```

```
<li class="nav-item">
  <a class="nav-link" href="#about">About</a>
</li>
<li class="nav-item">
  <a class="nav-link" href="#portfolio">Portfolio</a>
</li>
<li class="nav-item">
  <a class="nav-link" href="#contact">Contact</a>
</li>
</ul>
</nav>
<!-- About Section -->
<section id="about" class="py-5">
  <div class="container">
    <div class="row">
      <div class="col-lg-6">
        <h2>About Me</h2>
        <p>Hi there! I'm Naren, currently a student at Rajalakshmi Engineering College, pursuing Computer Science and Design. I'm passionate about exploring the intersection of technology and design to create innovative solutions. My journey in the field of computer science has equipped me with a strong foundation in programming, problem-solving, and software development. Additionally, my interest in design allows me to approach challenges with a creative mindset, striving to craft user-centric experiences.</p>
      </div>
      <div class="col-lg-6">
        
      </div>
    </div>
  </div>
```



```
</div>
</section>
<!-- Portfolio Section -->
<!-- Portfolio Section -->
<section id="portfolio" class="bg-light py-5">
  <div class="container">
    <h2>Portfolio</h2>
    <div class="row">
      <div class="col-md-4">
        <div class="card">
          
          <div class="card-body">
            <h5 class="card-title">E-commerce Website</h5>
            <p class="card-text">Developed a fully functional e-commerce website
using HTML, CSS, and JavaScript, integrated with payment gateway and user
authentication.</p>
            <a href="#" class="btn btn-primary">View Project</a>
          </div>
        </div>
      </div>
      </div>
      </div>
      <div class="col-md-4">
        <div class="card">
          
          <div class="card-body">
            <h5 class="card-title">Portfolio Website</h5>
```

```
<p class="card-text">Designed and built a responsive portfolio website to showcase personal projects, skills, and experiences using Bootstrap and custom CSS.</p>
```

```
<a href="#" class="btn btn-primary">View Project</a>
```

```
</div>
```

```
</div>
```

```
</div>
```

```
<div class="col-md-4">
```

```
<div class="card">
```

```

```

```
<div class="card-body">
```

```
<h5 class="card-title">Blog Website</h5>
```

```
<p class="card-text">Created a dynamic blog website using WordPress, customized themes, and plugins to enhance functionality and user experience.</p>
```

```
<a href="#" class="btn btn-primary">View Project</a>
```

```
</div>
```

```
</div>
```

```
</section>
```

```
<!-- Contact Section -->
```

```
<section id="contact" class="py-5">
```

```
<div class="container">
```

```
<h2>Contact Me</h2>
```

```
<div class="row">
```

```
<div class="col-md-6">
```

```
<form>
```

```
<div class="form-group">
```

```
<label for="name">Name</label>
```

```

        <input type="text" class="form-control" id="name">
    </div>
    <div class="form-group">
        <label for="email">Email address</label>
        <input type="email" class="form-control" id="email">
    </div>
    <div class="form-group">
        <label for="message">Message</label>
        <textarea class="form-control" id="message" rows="3"></textarea>
    </div>
    <button type="submit" class="btn btn-primary">Submit</button>
</form>
</div>
<div class="col-md-6">
    <div class="card bg-primary text-white">
        <div class="card-body">
            <h5 class="card-title">Get In Touch</h5>
            <p class="card-text">Feel free to contact me if you have any questions
or inquiries!</p>
        </div>
    </div>
</div>
</div>
</div>
</section>

<!-- Bootstrap JS and dependencies -->
<script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>

```

```

<script
src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.5.4/dist/umd/popper.min.js
"></script>

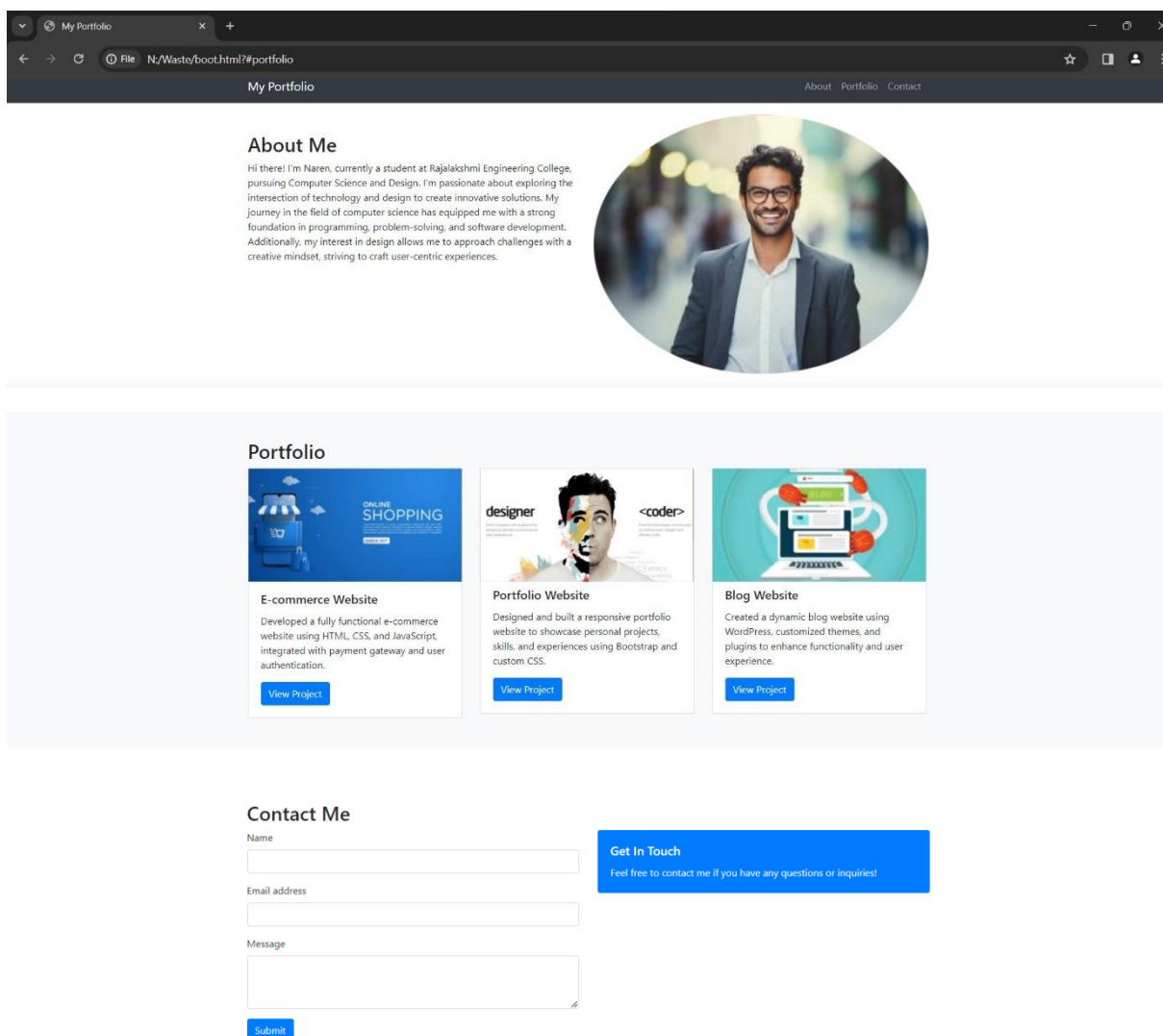
<script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></s
cript>

</body>

</html>

```

OUTPUT:



RESULT:

Hence developed an attractive web pages using Bootstrap.

EXPT NO: DESIGN A WEB PAGE WITH - NAVIGATION MENU

DATE:

AIM:

Program to design a web page with navigation menus using Angular JS.

PROCEDURE:

1. Using Angular's directives to set and read the active variable.
2. When it changes, it causes the HTML that uses it to be updated automatically.
3. In Angular's terminology, this variable is called a model. It is available to all directives in the current scope, and can be accessed in your controllers (more on that in the nextexample).
4. JavaScript templates are with the `{{var}}` syntax, the framework sees such a string, it replaces it with the contents of the variable.
5. This operation is repeated every time var is changed.

PROGRAM:

Index.html:

```
<!DOCTYPE html>

<html>

  <head>

    <link rel="stylesheet" href="style.css">

    <script src="https://code.angularjs.org/1.2.13/angular.js"></script>

    <script src="//cdnjs.cloudflare.com/ajax/libs/angular-material-icons/0.7.1/
      angular-material-icons.min.js"></script>

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

```
</head>
```

```
<body>
```

```
<!-- Adding the ng-app declaration to initialize AngularJS -->
```

```
<div id="main" ng-app="navApp">
```

```
  <nav class="{ { active} }" ng-click="$event.preventDefault()">
```

```
    <h2>Shopping Site</h2>
```

```
    <a href="#" class="home" ng-click="active='home'">
```

```
      <ng-md-icon icon="home" style="fill:white"></ng-md-icon>
```

```
    </a>
```

```
    <a href="#" class="electronics" ng-click="active='electronics'">
```

```
      Electronics</a>
```

```
    <a href="#" class="appliances" ng-click="active='appliances'">
```

```
      Appliances</a>
```

```
    <a href="#" class="clothing" ng-click="active='clothing'">
```

```
      Clothing</a>
```

```
  </nav>
```

```
  <p ng-hide="active">Please click a menu item</p>
```

```
  <p ng-show="active">You chose <b>{ { active} }</b></p>
```

```
</div>
```

```
</body>
```

```
</html>
```

STYLES.CSS:

```
*{  
  
    margin:0;  
  
    padding:0;  
  
}  
  
body{  
  
    font:15px/1.3 'Open Sans', sans-serif;  
  
    color: #9e32a8;  
  
    text-align:center;  
  
}  
  
a, a:visited {  
  
    outline:none;  
  
    color:#389dc1;  
  
}  
  
a:hover{  
  
    text-decoration:none;  
  
}  
  
a img{  
  
    display:inline-block;  
  
}  
  
section, footer, header, aside, nav{  
  
    display: block;}
```

```
nav{  
    display:inline-block;  
    margin:60px auto 45px;  
    background-color: #e887f5;  
    box-shadow:0 1px 1px #ccc;  
    border-radius:2px;  
}  
nav h2{  
    color:#fff !important;  
}  
nav a{  
    display:inline-block;  
    padding: 18px 30px;  
    color:#fff !important;  
    font-weight:bold;  
    font-size:16px;  
    text-decoration:none !important;  
    line-height:1;  
    text-transform: uppercase;  
    background-color:transparent;
```



```
-webkit-transition:background-color 0.25s;

-moz-transition:background-color 0.25s;

transition:background-color 0.25s;

}

nav a:first-child{

    border-radius:2px 0 0 2px;

}

nav a:last-child{

    border-radius:0 2px 2px 0;

}

nav.home .home,

nav.electronics .electronics,

nav.appliances .appliances,

nav.clothing .clothing{

    color:yellow !important;

}

p{

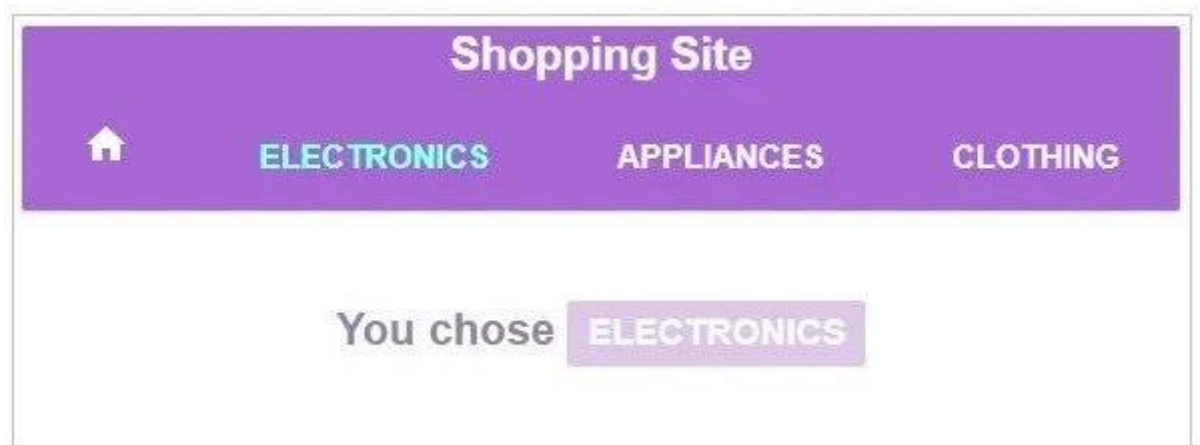
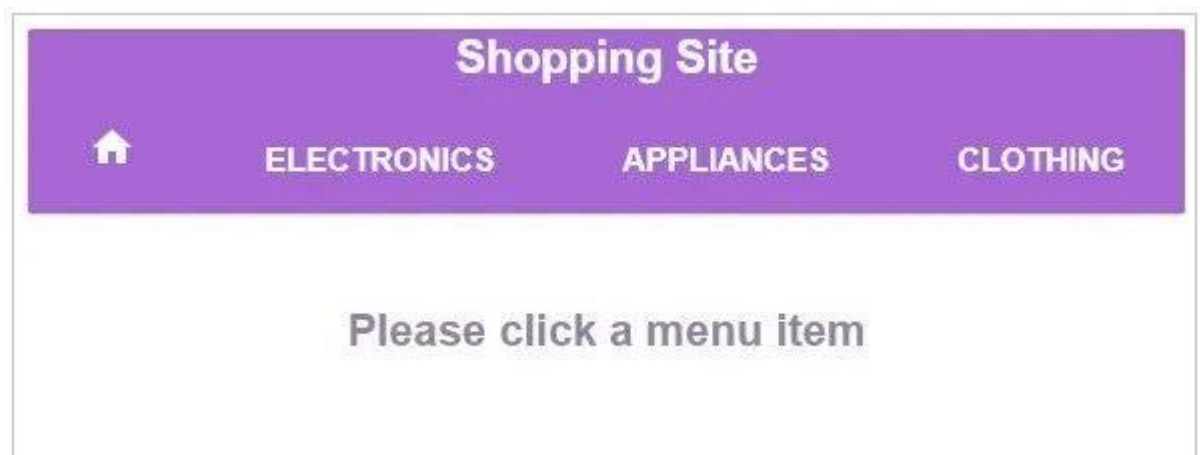
    font-size:22px;

    font-weight:bold;

    color:#7d9098;

}
```

OUTPUT:



RESULT:

Hence designed a web page with navigation menus using Angular JS.

DESIGN A WEB PAGE WITH – INLINE EDITOR

AIM:

Program to design a web page with inline editor using Angular JS.

PROCEDURE:

1. Clicking a paragraph will show a tooltip with a text field.
2. Use a controller that will initialize the models and declare two methods for toggling the visibility of the tooltip.
3. Controllers are regular JavaScript functions which are executed automatically by Angular, and which are associated with your page using the ng-controller directive.
4. When the controller function is executed, it gets the special \$scope object as a parameter.
5. Adding properties or functions to it makes them available to the view.
6. Using the ng-model binding on the text field tells Angular to update that variable when the value of the field changes (this in turn re-renders the paragraph with the value).

PROGRAM:

```
<!DOCTYPE html>

<html lang="en" ng-app="inlineEditorApp">
<head>

  <meta charset="UTF-8">

  <title>Bootstrap & AngularJS Inline Editor</title>

  <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css"
>

  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/5.15.3/css/all.min.css">

  <style>

    .center {
```

```

        display: flex;
        justify-content: center;
        align-items: center;
        height: 100vh;
    }
    .title-container {
        display: flex;
        align-items: center;
    }
    .title {
        margin: 0;
    }
</style>
</head>
<body>
    <div ng-controller="InlineEditorController as ctrl" class="container center">
        <div class="title-container">
            <h1 class="title" ng-hide="ctrl.editingTitle">{{ ctrl.title }}</h1>
            <div ng-show="ctrl.editingTitle">
                <input type="text" class="form-control" ng-model="ctrl.title" id="title">
                <button class="btn btn-outline-secondary ml-2" type="button" ng-
click="ctrl.saveField('title')"><i class="fas fa-check"></i></button>
            </div>
        </div>
        <button class="btn btn-outline-secondary mt-3" type="button" ng-
click="ctrl.editField('title')"><i class="fas fa-pencil-alt"></i> Edit
Title</button>
    </div>

```

```
<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></scri
pt>

<script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>

<script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></s
cript>

<script>

angular.module('inlineEditorApp', [])

.controller('InlineEditorController', function() {

  var ctrl = this;

  ctrl.title = "Click to edit me";

  ctrl.editingTitle = false;

  ctrl.editField = function(field) {

    ctrl.editingTitle = true;

    // Set a timeout to focus the input after it's shown

    setTimeout(function() {

      document.getElementById(field).focus();

    });

  };

  ctrl.saveField = function(field) {

    ctrl.editingTitle = false;

  };

});



</script>

</body>

</html>
```

OUTPUT:

Click to edit me 

WEB ESSENTIALS 


WEB ESSENTIALS 

RESULT:

Hence designed a web page with inline editor using Angular JS.

DESIGN A WEB PAGE WITH – ORDER FORM

AIM:

Program to design a web page with order form using Angular JS.

PROCEDURE:

1. Code an order form with a total price updated in real time, using another one of Angular's useful features - filters.
2. Filters let modify models and can be chained together using the pipe character.
3. Use the currency filter, to turn a number into a properly formatted price, complete with a dollar sign and cents. You can easily make your own filters.
4. The ng-repeat binding (docs) is another useful feature of the framework. It lets loop through an array of items and generate markup for them. It is intelligently updated when an item is changed or deleted.

PROGRAM:

```
<!DOCTYPE html>

<html lang="en" ng-app="orderFormApp">
<head>
  <meta charset="UTF-8">
  <title>Order Form</title>
  <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css"
>
  <style>
    body, html {
      height: 100%;
    }
    .container {
      display: flex;
      justify-content: center;
```

```
    align-items: center;
    height: 100%;
}
order-form {
    max-width: 400px;
    width: 100%;
}
form-group {
    margin-bottom: 20px;
}
form-label {
    font-weight: bold;
}
</style>
</head>
<body>
<div ng-controller="OrderFormController as formCtrl" class="container">
    <form class="order-form">
        <div class="form-group">
            <label for="products" class="form-label">Products:</label>
            <select multiple class="form-control" id="products" ng-
model="formCtrl.selectedProducts" ng-options="product as product.name for
product in formCtrl.products">
</select>
        </div>
        <div class="form-group">
            <label for="quantity" class="form-label">Quantity:</label>
```



```

        <input type="number" class="form-control" id="quantity" ng-
model="formCtrl.quantity">
    </div>

    <button type="button" class="btn btn-primary" ng-
click="formCtrl.addToCart()">Add to Cart</button>

    <div class="form-group mt-3">
        <label class="form-label">Cart:</label>

        <ul class="list-group">
            <li class="list-group-item" ng-repeat="item in formCtrl.cart">{{
item.product.name }} - Quantity: {{ item.quantity }} - Total: ${{ item.total
}}</li>
        </ul>

        <p class="mt-3">Grand Total: ${{ formCtrl.grandTotal }}</p>
    </div>
</form>
</div>

<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></scri
pt>

<script>
angular.module('orderFormApp', [])

.controller('OrderFormController', function() {
    var formCtrl = this;
    formCtrl.products = [
        { name: 'Web Essentials', cost: 100 },
        { name: 'Web Hosting', cost: 50 },
        { name: 'Domain Registration', cost: 20 }
    ];

```

```
formCtrl.selectedProducts = [];
formCtrl.quantity = "";
formCtrl.cart = [];
formCtrl.grandTotal = 0;

// Function to add item to cart
formCtrl.addToCart = function() {
  if (formCtrl.selectedProducts.length && formCtrl.quantity) {
    angular.forEach(formCtrl.selectedProducts, function(product) {
      var total = product.cost * formCtrl.quantity;
      formCtrl.cart.push({ product: product, quantity: formCtrl.quantity, total:
total });
      formCtrl.grandTotal += total;
    });
    // Clear input fields after adding to cart
    formCtrl.selectedProducts = [];
    formCtrl.quantity = "";
  } else {
    console.log("Please select product(s) and enter a quantity.");
  }
};

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

OUTPUT:

Products:

Web Essentials

Web Hosting

Domain Registration

Quantity:

Add to Cart

Cart:

Grand Total: \$0

Products:

Web Essentials

Web Hosting

Domain Registration

Quantity:

Add to Cart

Cart:

Web Essentials - Quantity: 2 - Total: \$200

Grand Total: \$200

Products:

Web Essentials

Web Hosting

Domain Registration

Quantity:

Add to Cart

Cart:

Web Essentials - Quantity: 2 - Total: \$200

Web Hosting - Quantity: 1 - Total: \$50

Domain Registration - Quantity: 2 - Total: \$40

Grand Total: \$290

RESULT:

Hence designed a web page with order form using Angular JS.

DESIGN A WEB PAGE WITH – INSTANT SEARCH

AIM:

Program to design a web page with instant search using Angular JS.

PROCEDURE:

1. To filter a list of items by typing into a text field.
2. First have to turn the application into a module.
3. Modules are a way of organizing JavaScript applications into self-contained components that can be combined in new and interesting ways.
4. Angular relies on this technique for code isolation and requires that your application follows it before you can create a filter.
5. There are only two things that you need to do to turn your app into a module:
 1. Use the `angular.module("name",[])` function call in your JS. This will instantiate and return a new module;
 2. Pass the name of the module as the value of the `ng-app` directive.
6. Creating a filter then is as simple as calling the `filter()` method on the module object returned by `angular.module("name", [])`.

PROGRAM:

```
<!DOCTYPE html>

<html lang="en" ng-app="instantSearchApp">
<head>

  <meta charset="UTF-8">

  <title>Instant Search - Engineering Departments</title>

  <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css"
>

</head>

<body>

  <div ng-controller="SearchController as searchCtrl" class="container mt-5">

    <h1>Instant Search - Engineering Departments</h1>
```

```

    <input type="text" class="form-control mt-3" placeholder="Search..." ng-
model="searchCtrl.query">

    <ul class="list-group mt-3">

        <li class="list-group-item" ng-repeat="department in searchCtrl.departments
| filter:searchCtrl.query">{{ department }}</li> </ul>

    </div>

<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></scri
pt>

<script>
angular.module('instantSearchApp', [])
.controller('SearchController', function() {
    var searchCtrl = this;
    searchCtrl.departments = [
        'Computer Science and Engineering',
        'Computer Science and Design',
        'Electrical and Electronics Engineering',
        'Mechanical Engineering',
        'Civil Engineering',
        'Electronics and Communication Engineering',
        'Chemical Engineering',
        'Biomedical Engineering',
        'Aerospace Engineering',
        'Environmental Engineering',
        'Materials Science and Engineering'
    ];
});
</script>
</body>
</html>

```

OUTPUT:

Instant Search - Engineering Departments

| |
|---|
| Search... |
| Computer Science and Engineering |
| Computer Science and Design |
| Electrical and Electronics Engineering |
| Mechanical Engineering |
| Civil Engineering |
| Electronics and Communication Engineering |
| Chemical Engineering |
| Biomedical Engineering |
| Aerospace Engineering |
| Environmental Engineering |
| Materials Science and Engineering |

Instant Search - Engineering Departments

| |
|-----------------------------|
| Computer science and design |
| Computer Science and Design |

Instant Search - Engineering Departments

| |
|------------------------|
| Bio |
| Biomedical Engineering |

RESULT:

Hence designed a web page with instant search using Angular JS.

DESIGN A WEB PAGE WITH – SWITCHABLE GRID

Aim:

Program to design a web page with Switchable grid using Angular JS.

Procedure:

1. Create the HTML structure for the switchable grid interface. Include buttons for switching between grid and list views and define containers for displaying the grid and list.
2. Set up an AngularJS module and controller to manage the functionality of the switchable grid. Inject the AngularJS library into the HTML file.
3. Initialize data and view: Define a sample list of items and initialize the default view to be displayed (in this case, the grid view).
4. Create a function in the AngularJS controller to switch between grid and list views. This function will change the value of a variable representing the current view.
5. Use AngularJS directives such as ng-show and ng-repeat to bind the data to the grid and list views. Show or hide each view based on the current view variable, and iterate over the list of items to display them in the respective views.

PROGRAM:

```
<!DOCTYPE html>

<html lang="en" ng-app="switchableViewApp">
<head>

  <meta charset="UTF-8">

  <title>Switchable Grid</title>
```

```

    <linkrel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css"
>

</head>

<body>

    <div ng-controller="SwitchableGridController as gridCtrl" class="container
mt-5">

        <h1>Switchable Grid</h1>

        <div class="btn-group mb-3">

            <button                class="btn                btn-primary"                ng-
click="gridCtrl.switchView('grid')">Grid View</button>

            <button class="btn btn-primary" ng-click="gridCtrl.switchView('list')">List
View</button>

        </div>

        <div ng-show="gridCtrl.currentView === 'grid'">

            <div class="row">

                <div class="col-md-4 mb-3" ng-repeat="item in gridCtrl.items">

                    <div class="card">

                        <div class="card-body">

                            {{ item }}

                        </div>

                    </div>

                </div>

            </div>

        </div>

        <div ng-show="gridCtrl.currentView === 'list'">

            <ul class="list-group">

```



```
<li class="list-group-item" ng-repeat="item in gridCtrl.items">{{ item }}</li>
  </ul>
</div>
</div>
```

```
<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></scri
pt>
```

```
<script>
```

```
angular.module('switchableViewApp', [])
.controller('SwitchableGridController', function() {
  var gridCtrl = this;
  gridCtrl.items = [
    'Item 1', 'Item 2', 'Item 3', 'Item 4', 'Item 5'
  ];
  gridCtrl.currentView = 'grid';
  gridCtrl.switchView = function(view) {
    gridCtrl.currentView = view;
  };
});
```

```
</script>
```

```
</body>
```

```
</html>
```

OUTPUT:

Switchable Grid

Grid View List View

| | | |
|--------|--------|--------|
| Item 1 | Item 2 | Item 3 |
| Item 4 | Item 5 | |

Switchable Grid

Grid View List View

| |
|--------|
| Item 1 |
| Item 2 |
| Item 3 |
| Item 4 |
| Item 5 |

RESULT:

Hence designed a web page with Switchable grid using Angular JS.

EXPT NO: SINGLE PAGE APPLICATION

DATE:

AIM:

Program to develop an single page application using angular js.

PROCEDURE:

1. Make a single page application and don't want any page refreshes, use Angular's routing capabilities.
2. Include angular-route script after the main angular script. 8. Specify that the module depends on ngRoute module to be able to use it.
3. The next thing is to distinguish common HTML for every page. This HTML will be layout of the website.
4. Then specify the place where HTML of each page will be placed in our layout. There is a ng-view directive for that.
5. ng-view is an Angular directive that will include the template of the current route (for example, /blog or /about) in the main layout file.
6. Configure the routes. Use \$routeProvider service from the ngRoute module.
7. For each route, specify templateUrl and controller.
8. If user will try to go to the route that does not exist, handle this by using otherwise function. In our case, we will redirect user to the "/" route:
9. Build controllers for every route (already specified their names in routeProvider).

PROGRAM:

```
<!DOCTYPE html>
<html ng-app="myApp">
<head>
  <meta charset="UTF-8">
  <title>Single Page Application</title>
  <style>
    body {
```

```
font-family: Arial, Times new roman;
margin: 0;
padding: 0;
background-color: #f4f4f4;
display: flex;
flex-direction: column;
justify-content: center;
align-items: center;
height: 100vh;
}
#navbar {
background-color: #333;
overflow: hidden;
display: flex;
justify-content: center;
align-items: center;
width: 100%;
position: fixed;
top: 0;
}
#navbar a {
display: block;
color: white;
text-align: center;
padding: 14px 16px;
text-decoration: none;
}
```

```
#navbar a:hover {
    background-color: #ddd;
    color: black;
}

#navbar a.active {
    background-color: #4CAF50;
    color: white;
}

.content {
    padding: 20px;
    text-align: center;
    margin-top: 60px; /* Adjusted margin to make space for navbar */
}

h1 {
    margin-top: 0;
}

img {
    max-width: 100%;
    height: auto;
}

</style>

<script
src="https://cdnjs.cloudflare.com/ajax/libs/angular.js/1.4.7/angular.min.js"></sc
ript>

<script src="https://cdnjs.cloudflare.com/ajax/libs/angular.js/1.4.7/angular-
route.min.js"></script>

</head>
```

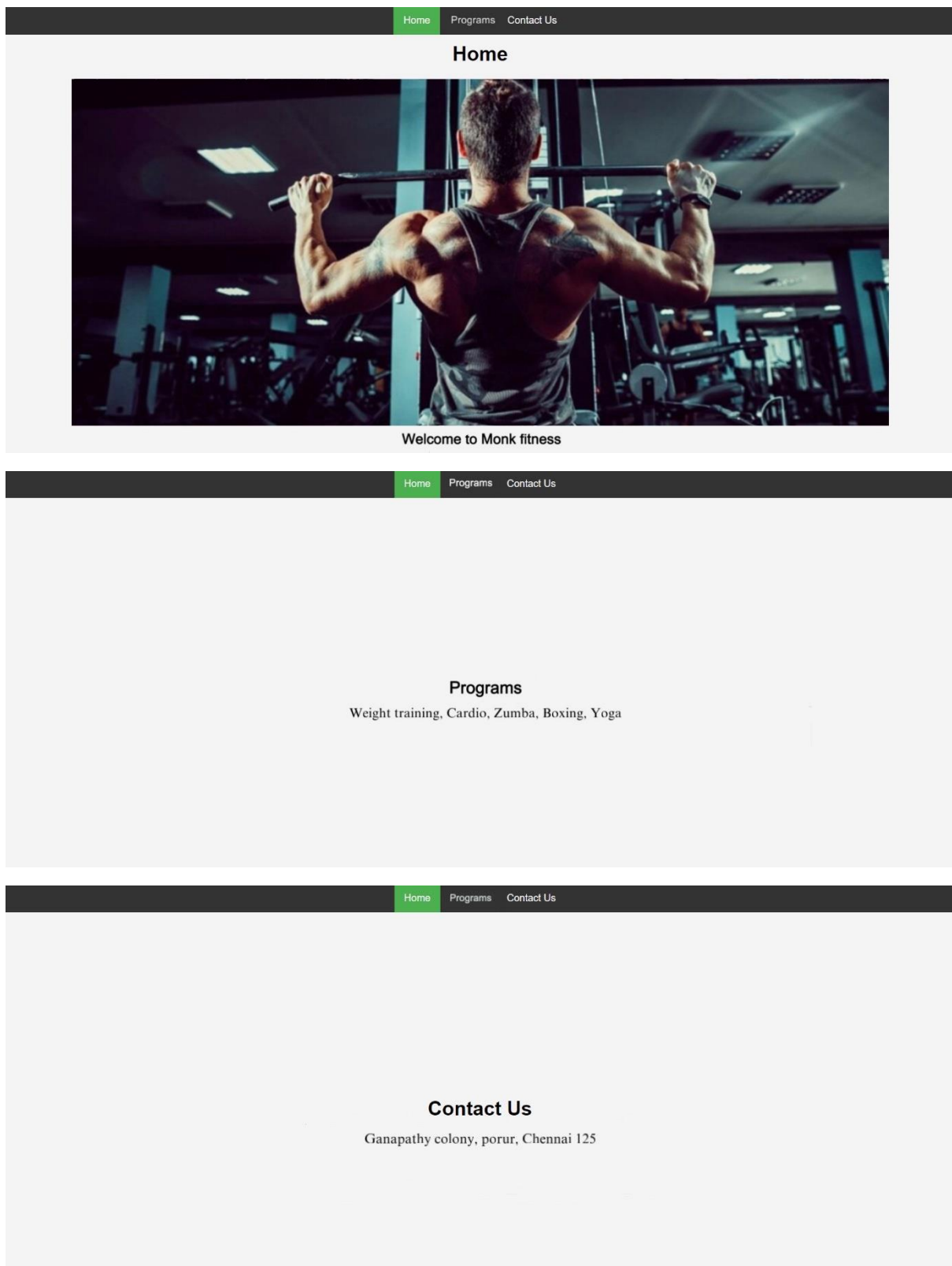
```

<body>
  <div id="navbar">
    <a href="#" class="active">Home</a>
    <a href="#/courses">Programs</a>
    <a href="#/contactus">Contact Us</a>
  </div>
  <div ng-view class="content"></div>
<script type="text/ng-template" id="pages/home.html">
  <h1>Home</h1>
  
  <h3>{{message}}</h3>
</script>
<script type="text/ng-template" id="pages/courses.html">
  <h1>Courses</h1>
  <h3>{{message}}</h3>
</script>
<script type="text/ng-template" id="pages/contactus.html">
  <h1>Contact Us</h1>
  <h3>{{message}}</h3>
</script>
<script>
  var app = angular.module('myApp', ['ngRoute']);
  app.config(function($routeProvider) {
    $routeProvider
      .when('/', {
        templateUrl : 'pages/home.html',
        controller : 'HomeController'
      })
  });

```

```
    })
    .when('/courses', {
        templateUrl : 'pages/courses.html',
        controller : 'CoursesController'
    })
    .when('/contactus', {
        templateUrl : 'pages/contactus.html',
        controller : 'ContactUsController'
    })
    .otherwise({redirectTo: '/'});
});
app.controller('HomeController', function($scope) {
    $scope.message = 'Welcome to Monk Fitness';
});
app.controller('CoursesController', function($scope) {
    $scope.message = 'Weight Training, Cardio, Zumba, Boxing, Yoga';
});
app.controller('ContactUsController', function($scope) {
    $scope.message = 'Ganapathy colony, porur, chennai 125';
});
</script>
</body>
</html>
```

OUTPUT:



RESULT:

Hence developed a single page application using Angular JS.