Name: Shital R.Gurule
Std.: Msc (Comp. Sci.)-II
Sub.: Web Framework

---

1. Create an HTML. form that contains the Student Registration details and write a JavaScript to validate Student first and last name as it should not contain other than alphabets and age should be between 18 to 50.

➔

```
<!DOCTYPE html>
 <html>
<head>
<script type="text/javascript" src="validateform.js"></script>
<style>
ul {list-style-type:none;}
form{
background-color: #DCDCDC;
}
</style>
</head>
 <body>
<form action="#" name="StudenSignupForm" onsubmit="return(validateHTMlform());">
<div cellpadding="2" width="20%" bgcolor="99FFFF" align="center"
cellspacing="2">
<ul>
<li>
<center><font size=4><b>Student Registration Form</b></font></center>
</li>
</ul>
 <ul>
<li>First Name</li>
<li><input type=text name=textnames id="textname" size="30"></li>
</ul>
 <ul>
<li>
Last Name</li>
<li><input type=text name=lastnames id="lastname" size="30"></li>
</ul>
 <ul>
<li>Father Name</li>
<li><input type="text" name="full_father_name" id="fathername"
size="30"></li>
</ul>
 <ul>
```

```
<li>Address</li>
<li><input type="text" name="personal_address"
id="personaladdress" size="30"></li>
</ul>
<ul>
<li>Gender</li>
<li><input type="radio" name="sex" value="male" size="10">Male
<input type="radio" name="sex" value="Female" size="10">Female</li>
</ul>
<ul>
<li>City</li>
<li><select name="City">
<option value="-1" selected>select..</option>
<option value="KOLKATA">KOLKATA</option>
<option value="CHENNAI">CHENNAI</option>
<option value="PUNE">PUNE</option>
<option value="JAIPUR">JAIPUR</option>
</select></li>
</ul>
<ul>
<li>Course</li>
<li><select name="Course">
<option value="-1" selected>select..</option>
<option value="B.Tech">B.TECH</option>
<option value="MCA">MCA</option>
<option value="MBA">MBA</option>
<option value="BCA">BCA</option>
</select></li>
</ul>
<ul>
<li>State</li>
<li><select Name="State">
<option value="-1" selected>select..</option>
<option value="New Delhi">NEW DELHI</option>
<option value="Mumbai">MUMBAI</option>
<option value="Goa">GOA</option>
<option value="Bihar">BIHAR</option>
</select></li>
</ul>
<ul>
<li>District</li>
<li><select name="Disulict">
<option value="-1" selected>select..</option>
<option value="Nalanda">NALANDA</option>
```

```
      <option value="UP">UP</option>
      <option value="Goa">GOA</option>
      <option value="Patna">PATNA</option>
      </select></li>
     </ul>
      <ul>
      <li>PinCode</li>
      <li><input type="text" name="pin_code" id="pincode" size="30"></li>
     </ul>
      <ul>
      <li>student email</li>
      <li><input type="text" name="email_id" id="emailid" size="30"></li>
      </ul>
      <ul>
      <li>Date Of Birth</li>
      <li><input type="text" name="date_of_birth" id="dob" size="30"></li>
      </ul>
      <ul>
      <li>Mobile Number</li>
      <li><input type="text" name="mobilenumber" id="mobile_no" size="30"></li>
      </ul>
      <ul>
      <li><input type="reset"></li>
      <li colspan="2"><input type="submit" value="Submit Form" /></li>
      </ul>
      </div>
      </form>
      </body>
</html>
```

validateform.js

```
function validateHTMlform()
{
  let form = document.StudenSignupForm;
   if( form.textnames.value == "" )
   {
     alert( "Enter Your First Name!" );
     form.textnames.focus() ;
     return;
   }
   if( form.lastnames.value == "" )
   {
     alert( "Enter Your Last Name!" );
```

```
    form.textnames.focus() ;
    return;
   }
   if( form.fathername.value == "" )
   {
    alert( "Enter Your Father Name!" );
    form.fathername.focus() ;
    return;
   }
   if( form.paddress.value == "" )
   {
    alert( "Enter Your Postal Address!" );
    form.paddress.focus() ;
    return;
   }
   if( form.personaladdress.value == "" )
   {
    alert( "Enter Your Personal Address!" );
    form.personaladdress.focus() ;
    return;
   }
   if ( ( StudenSignupForm.sex[0].checked == false ) && (
StudenSignupForm.sex[1].checked == false ) )
   {
   alert ( "Choose Your Gender: Male or Female" );
   return false;
   }
   if( form.City.value == "-1" )
   {
    alert( "Enter Your City!" );
    form.City.focus() ;
    return;
   }
   if( form.Course.value == "-1" )
   {
    alert( "Enter Your Course!" );
    return;
   }
   if( form.District.value == "-1" )
   {
    alert( "Select Your District!" );
   return;
   }
   if( form.State.value == "-1" )
```

```
  {
    alert( "Select Your State!" );
   return;
  }
  if( form.pincode.value == "" ||isNaN( form.pincode.value)
||form.pincode.value.length != 6 )
  {
    alert( "Enter your pincode in format ######." );
    form.pincode.focus() ;
    return;
  }
 var email = form.emailid.value;
 atpos = email.indexOf("@");
 dotpos = email.lastIndexOf(".");
 if (email == "" || atpos < 1 || ( dotpos - atpos < 2 ))
 {
    alert("Enter your correct email ID")
    form.emailid.focus() ;
    return;
}
 if( form.dob.value == "" )
  {
    alert( "Enter your DOB!" );
    form.dob.focus() ;
    return;
  }
 if( form.mobileno.value == "" ||
        isNaN( form.mobileno.value) ||
        form.mobileno.value.length != 10 )
  {
    alert( "Enter your Mobile No. in the format 123." );
    form.mobileno.focus() ;
    return;
  }
  return( true );
}
```

2. **Create an HTML form that contain the Employee Registration details and write a JavaScript to validate DOB, Joining Date, and Salary**.

➔

javaScript to validate DOB. joining Date, and Salary.

```
<!DOCTYPE html>
<html lang="en"><head>
<meta charset="utf-8">
<title>JavaScript Form Validation using a sample Employee registration form</title
>
<meta name="keywords" content="example, JavaScript Form Validation, Sample re
gistration form" />
<meta name="description" content="This document is an example of JavaScript For
m Validation using a sample registration form. " />
<link rel='stylesheet' href='employee.css' type='text/css' />
<script src="employee.js">
</script>
</head>
<body onload="document.registration.userid.focus();"bgcolor="orange">
<h1>Employee Registration Form</h1>
<form name='registration' onSubmit="return formValidation();">
<ul>
<li><label for="first">First Name:</label></li>
<li><input type="text" name="first" size="50" /></li>
<li><label for="last">Last Name:</label></li>
<li><input type="text" name="last" size="50" /></li>
<li><label for="empid">Employee id:</label></li>
<li><input type="text"name="empid" size="50"/></li>
<li><label for="birth">Birth of date:</label></li>
<li><input type="date" id="birth" name="birth"></li>
<li><label for="address">Address:</label></li>
<li><input type="text" name="address" size="50" /></li>
<li><label for="country">Country:</label></li>
<li><select name="country">
<option selected="" value="Default">(Please select a country)</option>
<option value="AF">Australia</option>
<option value="AL">Canada</option>
<option value="DZ">India</option>
<option value="AS">Russia</option>
<option value="AD">USA</option>
```

```
</select></li>
<li><label for="no">Contact no:</label></li>
<li><input type="number" id="" name="no"></li>
<li><label for="jdate">Date of joining:</label></li>
<li><input type="date" id="" name="jdate"></li>
<li><label for="email">Email:</label></li>
<li><input type="text" name="email" size="50" /></li>
<li><label id="gender">Gender:</label></li>
<li><input type="radio" name="male" value="Male" /><span>Male</span></li>
<li><input type="radio" name="female" value="Female" /><span>Female</span></
li>
<li><label for="salary">salary:</label></li>
<li><input type="number" id="salary" name="salary"></li>
<li><input type="submit" name="submit" value="Submit" /></li>
</ul>
</form>
</body>
</html>
```

employee.js

```
function formValidation()
{
var first=document.registration.first;
var last=document.registration.last;
var empid=document.registration.empid;
var birth=document.registration.birth;
var uadd =document.registration.address;
var ucountry =document.registration.country;
var no=document.registration.no;
var jdate=document.registration.jdate;
var uemail = document.registration.email;
var umgen = document.registration.umgen;
var ufgen = document.registration.ufgen;
var salary =document.registration.salary;
if(allLetter(first))
{
if(allLetter(last))
{
if(alphanumeric(empid))
```

```
{
if(allb(birth))
{
if(alphanumeric(uadd))
{
if(countryselect(ucountry))
{
if(allnumeric(no))
{
if(allnumeric(jdate))
{
if(ValidateEmail(uemail))
{
if(validgendor(umgen,ufgen))
{
if(allnumeric(salary))
{
}}}}}}}}}}
return false;
}
function allLetter(first)
{
var letters = /^[A-Za-z]+$/;
if(first.value.match(letters))
{
alert('employee name submitted');
return true;
}
else
{
alert('employee name must have alphabet characters only');
first.focus();
return false;
}
}
function allLetter(last)
{
var letters = /^[A-Za-z]+$/;
if(last.value.match(letters))
{
   alert("employee name submitted");
return true;
}
else
```

```
{
alert('employee name must have alphabet characters only');
last.focus();
return false;
}
}
function alphanumeric(empid)
{
var letters = /^[0-9a-zA-Z]+$/;
if(empid.value.match(letters))
{
   alert("employee id submitted");
return true;
}
else
{
alert('employee id must have alphanumeric characters only');
uadd.focus();
return false;
}
}
function allb(birth)
{
var birth_len = birth.value.length;
if (birth_len == 0)
{
alert("birth date should not be empty");
birth.focus();
return false;
}
alert("birth of date submitted");
return true;
}
function alphanumeric(uadd)
{
var letters = /^[0-9a-zA-Z]+$/;
if(uadd.value.match(letters))
{
   alert("address submitted");
return true;
}
else
{
alert('address must have alphanumeric characters only');
```

```
uadd.focus();
return false;
}
}
function countryselect(ucountry)
{
if(ucountry.value == "Default")
{
alert('Select your country from the list');
ucountry.focus();
return false;
}
else
{
   alert("country submitted");
return true;
}
}
function allnumeric(no)
{
var number = /^[0-9]+$/;
if(no.value.match(number))
{
      alert("Contact Number submitted");
   return true;
   }
   else
   {
   alert('Contact no must have numeric numbers only');
   no.focus();
   return false;
   }
}
function allnumeric(jdate)
{
var jdate_len = jdate.value.length;
if (jdate_len == 0)
{
alert("date of joining should not be empty");
birthday.focus();
return false;
}
alert("date of joining submitted");
return true;
```

```
}
function ValidateEmail(uemail)
{
var mailformat = /^\w+([\.-]?\w+)*@\w+([\.-]?\w+)*(\.\w{2,3})+$/;
if(uemail.value.match(mailformat))
{
   alert("email address is submitted");
return true;
}
else
{
alert("You have entered an invalid email address!");
uemail.focus();
return false;
}
}
function validgender(umgen,ufgen)
{
x=0;
if(umgen.checked)
{
x++;
} if(ufgen.checked)
{
x++;
}
if(x==0)
{
alert('Select Male/Female');
umgen.focus();
return false;
}
else
{
window.location.reload()
return true;
}
}
function allnumeric(salary)
{
var sal = /^[0-9]+$/;
if(salary.value.match(sal))
{
alert("salary submitted");
```

```
return true;
}
else
{
alert('salry is not submitted');
salary.focus();
return false;
}
}
/*function underAgeValidate(births){
   // it will accept two types of format yyyy-mm-dd and yyyy/mm/dd
   var optimizedBirthday = births.replace(/-/g, "/");

   //set date based on birthday at 01:00:00 hours GMT+0100 (CET)
   var myBirthday = new Date(optimizedBirthday);

   // set current day on 01:00:00 hours GMT+0100 (CET)
   var currentDate = new Date().toJSON().slice(0,10)+' 01:00:00';

   // calculate age comparing current date and borthday
   var myAge = ~~((Date.now(currentDate) - myBirthday) / (31557600000));

   if(myAge < 18)
   {
   alert("age is not validate");
   }
   else
   {
   alert("age is validate");
   }
}*/
```

employee.css

```
h1
 {
   margin-left: 70px;
   }
   form li {
   list-style: none;
   margin-bottom: 5px;
   }

   form ul li label{
```

```
float: left;
clear: left;
width: 100px;
text-align: right;
margin-right: 10px;
font-family:Verdana, Arial, Helvetica, sans-serif;
font-size:14px;
}

form ul li input, select, span {
float: left;
margin-bottom: 10px;
}

form textarea {
float: left;
width: 350px;
height: 150px;
}

[type="submit"] {
clear: left;
margin: 20px 0 0 230px;
font-size:18px
}

p {
margin-left: 70px;
font-weight: bold;
}
```

3. Create an HTML form for Login and write a Javascript to validate email ID using Regular Expression.

➔

```
function ValidateEmail(mail)
{
 if (/^\w+([\.-]?\w+)*@\w+([\.-
]?\w+)*(\.\w{2,3})+$/.test(myForm.emailAddr.value))
  {
    return (true)
  }
    alert("You have entered an invalid email address!")
    return (false)
}
```

HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>JavaScript form validation - checking email</title>
<link rel='stylesheet' href='form-style.css' type='text/css' />
</head>
<body onload='document.form1.text1.focus()'>
<div class="mail">
<h2>Input an email and Submit</h2>
<form name="form1" action="#">
<ul>
<li><input type='text' name='text1'/></li>
<li> </li>
<li class="submit"><input type="submit" name="submit" value="Submit"
onclick="ValidateEmail(document.form1.text1)"/></li>
<li> </li>
</ul>
</form>
</div>
<script src="email-validation.js"></script>
</body>
</html>
```

JavaScript Code

```
function ValidateEmail(inputText)
{
var mailformat = /^\w+([\.-]?\w+)*@\w+([\.-]?\w+)*(\.\w{2,3})+$/;
if(inputText.value.match(mailformat))
{
alert("Valid email address!");
document.form1.text1.focus();
return true;
}
else
{
alert("You have entered an invalid email address!");
document.form1.text1.focus();
return false;
}
}
```

CSS Code

```
li {list-style-type: none;
font-size: 16pt;
}
.mail {
margin: auto;
padding-top: 10px;
padding-bottom: 10px;
width: 400px;
background : #D8F1F8;
border: 1px soild silver;
}
.mail h2 {
margin-left: 38px;
```

4. Create a Node.js file that will convert the output "Hello Worldr" into upper-case letters

➔

```
 var http = require('http');
 var uc = req
 uire('upper-case');
 http.createServer(function (req, res) {
 res.writeHead(200, {'Content-Type': 'text/html'});
 res.write(uc.upperCase("Hello World!"));
 res.end();
 }).listen(8080);
```

5. Using nodejs create a web page to read two file names from user and append contents of first file into second file.

➔

```
  <!DOCTYPE html>
 <html xmlns="http://www.w3.org/1999/xhtml">
<head>
   <meta charset="utf-8" />
   <title>The Page Returned by Making Http Call to Node.js</title>
   <style type="text/css">
      table, td {
        border:double;
      }
   </style>
</head>
<body>
```

```
  <h1>Product Information Page</h1>
  <table>
    <tr>
      <td>Product Id:</td>
      <td>
        <input type="text" />
      </td>
    </tr>
    <tr>
      <td>Product Name:</td>
      <td>
        <input type="text" />
      </td>
    </tr>
    <tr>
      <td></td>
      <td>
        <input type="button"  value="Save"/>
      </td>
    </tr>
  </table></body></html>
```

This is a simple HTML file which will be sent with the request.

Step 3: Open app.js and add the following code in it

```
//1
var http = require('http');
var fs = require('fs');
//2
var server = http.createServer(function (req, resp) {
//3
if (req.url === "/create") {
    fs.readFile("AppPages/MyPage.html", function (error, pgResp) {
      if (error) {
        resp.writeHead(404);
        resp.write('Contents you are looking are Not Found');
      } else {
        resp.writeHead(200, { 'Content-Type': 'text/html' });
        resp.write(pgResp);
      }

      resp.end();
    });
  } else {
  //4
resp.writeHead(200, { 'Content-Type': 'text/html' });
```

```
      resp.write('<h1>Product Manaager</h1><br /><br />To create product please
enter: ' + req.url);
      resp.end();
   }
});
//5
server.listen(5050);
console.log('Server Started listening on 5050');
```

6. Create a Node.js file that opens the requested file and returns the content to the client. If anything goes wrong throw a 404 error.

➔

```
  <!DOCTYPE html>
   <html>
  <body>
  <h1>Summer</h1>
  <p>I love the sun!</p>
  </body>
  </html>

<!DOCTYPE html>
<html>
<body>
<h1>Winter</h1>
<p>I love the snow!</p>
</body>
</html>

var http = require('http');
var url = require('url');
var fs = require('fs');

http.createServer(function (req, res) {
var q = url.parse(req.url, true);
var filename = "." + q.pathname;
fs.readFile(filename, function(err, data) {
if (err) { res.writeHead(404, {'Content-Type': 'text/html'});
return res.end("404 Not Found");
}
res.writeHead(200, {'Content-Type': 'text/html'});
res.write(data);
return res.end();
});
}).listen(8080);
```

7. Create a Node.js file that writes an HTML form, with an upload field.

➔

```
var http = require('http');
http.createServer(function (req, res) {
res.writeHead(200, {'Content-Type': 'text/html'});
res.write('<form action="fileupload" method="post"    enctype="multipart/form-
data">');
 res.write('<input type="file" name="filetoupload"><br>');
 res.write('<input type="submit">');
 res.write('</form>');
 return res.end();
}).listen(8080);
var http = require('http');
var formidable = require('formidable');

http.createServer(function (req, res) {
 if (req.url == '/fileupload') {
  var form = new formidable.IncomingForm();
  form.parse(req, function (err, fields, files) {
   res.write('File uploaded');
   res.end();
  });
 } else {
  res.writeHead(200, {'Content-Type': 'text/html'});
  res.write('<form action="fileupload" method="post" enctype="multipart/form-
data">');
  res.write('<input type="file" name="filetoupload"><br>');
  res.write('<input type="submit">');
  res.write('</form>');
  return res.end();
 }
}).listen(8080);

var http = require('http');
var formidable = require('formidable');
var fs = require('fs');

http.createServer(function (req, res) {
 if (req.url == '/fileupload') {
  var form = new formidable.IncomingForm();
  form.parse(req, function (err, fields, files) {
   var oldpath = files.filetoupload.filepath;
   var newpath = 'C:/Users/Your Name/' + files.filetoupload.originalFilename;
   fs.rename(oldpath, newpath, function (err) {
```

```
      if (err) throw err;
      res.write('File uploaded and moved!');
      res.end();
     });
 });
  } else {
    res.writeHead(200, {'Content-Type': 'text/html'});
    res.write('<form action="fileupload" method="post" enctype="multipart/form-
data">');
    res.write('<input type="file" name="filetoupload"><br>');
    res.write('<input type="submit">');
    res.write('</form>');
    return res.end();
  }
}).listen(8080);
```

8. Create a Nodejs file that demonstrate create database and table in MySQL.

➔

```
   var mysql = require('mysql');

var con = mysql.createConnection({
  host: "localhost",
  user: "yourusername",
  password: "yourpassword",
  database: "mydb"
});

con.connect(function(err) {
  if (err) throw err;
  con.query("SELECT * FROM customers", function (err, result, fields) {
    if (err) throw err;
    console.log(result);
  });
});
var mysql = require('mysql');

var con = mysql.createConnection({
  host: "localhost",
  user: "yourusername",
  password: "yourpassword",
  database: "mydb"
});

con.connect(function(err) {
```

```
  if (err) throw err;
  con.query("SELECT name, address FROM customers", function (err, result, fields) {
    if (err) throw err;
    console.log(result);
  });
});

var mysql = require('mysql');

var con = mysql.createConnection({
  host: "localhost",
  user: "yourusername",
  password: "yourpassword",
  database: "mydb"
});

con.connect(function(err) {
  if (err) throw err;
  con.query("SELECT name, address FROM customers", function (err, result, fields) {
    if (err) throw err;
    console.log(fields);
  });
});
```

9). Create a node.js file that Insert Multiple Records in "student" table. and display the result object on console.

➔

```
    class Date
    {
    int day, month, year;
    Date(int day, int month, int year)
    {
        this.day = day;
        this.month = month;
        this.year = year;
    }
    Date(){}
    }
    import java.io.*;
    import java.lang.*;
    class Students extends Date
    {
        int id;
        String name;
```

```
    Date d1;
    int marks[] = new int[3];
    Students(int id, String name, Date d, int s1, int s2, int s3)
    {
        this.id = id;
        this.name = name;
        marks[0] = s1;
        marks[1] = s2;
        marks[2] = s3;
        d1 = new Date(d.day, d.month, d.year);
    }
    public void display()
    {
        System.out.println("\n\nID   Name\tDOB\t  Marks of 3 Subjects");
        System.out.println("===  =====\t=======\t  ====================");
        System.out.println(+id+"  "+name+"
  \t"+d1.day+"/"+d1.month+"/"+d1.year+"    "+marks[0]+"  "+marks[1]+"
  "+marks[2]);
        System.out.println("===  =====\t=======\t  ====================");
    }
    public static void main(String ar[])
    {
        Date d = new
  Date(Integer.parseInt(ar[2]),Integer.parseInt(ar[3]),Integer.parseInt(ar[4]));
        Students  s1 = new
  Students(Integer.parseInt(ar[0]),ar[1],d,Integer.parseInt(ar[5]),Integer.parseInt(
  ar[6]),Integer.parseInt(ar[7]));
        s1.display();
    }
}
```

10). Create a node.js file that Select all records from the "customers" table, and delete the specified record.

➔

```
var mysql = require('mysql');

var con = mysql.createConnection({
  host: "localhost",
  user: "yourusername",
  password: "yourpassword",
  database: "mydb"
});

con.connect(function(err) {
```

```
  if (err) throw err;
  var sql = "DELETE FROM customers WHERE address = 'Mountain 21'";
  con.query(sql, function (err, result) {
    if (err) throw err;
    console.log("Number of records deleted: " + result.affectedRows);
  });
});
```

```
{
  fieldCount: 0,
  affectedRows: 1,
  insertId: 0,
  serverStatus: 34,
  warningCount: 0,
  message: '',
  protocol41: true,
  changedRows: 0
}
```

11. Create a Simple Web Server using node js.

➔

```
    var http = require('http'); // 1 - Import Node.js core module
    var server = http.createServer(function (req, res) {   // 2 - creating server
    //handle incomming requests here..
    });
    server.listen(5000); //3 - listen for any incoming requests
   console.log('Node.js web server at port 5000 is running..')
   var http = require('http'); // Import Node.js core module
   var server = http.createServer(function (req, res) {   //create web server
    if (req.url == '/') { //check the URL of the current request

        // set response header
        res.writeHead(200, { 'Content-Type': 'text/html' });
          // set response content
        res.write('<html><body><p>This is home Page.</p></body></html>');
        res.end();
    }
    else if (req.url == "/student") {
        res.writeHead(200, { 'Content-Type': 'text/html' });
        res.write('<html><body><p>This is student Page.</p></body></html>');
        res.end();
    }
    else if (req.url == "/admin") {
        res.writeHead(200, { 'Content-Type': 'text/html' });
```

```
        res.write('<html><body><p>This is admin Page.</p></body></html>');
        res.end();
    }
    else
        res.end('Invalid Request!');

});

server.listen(5000); //6 - listen for any incoming requests
console.log('Node.js web server at port 5000 is running..')
var http = require('http');
var server = http.createServer(function (req, res) {
    if (req.url == '/data') { //check the URL of the current request
            res.writeHead(200, { 'Content-Type': 'application/json' });
            res.write(JSON.stringify({ message: "Hello World"}));
            res.end();
    }
});
server.listen(5000);
console.log('Node.js web server at port 5000 is running..')
```

12). Using node is create a User Login System

➔

```
 <h1>login</h1>
 <form action="/login" method="POST">
   <input type="text" name="username"
      placeholder="username">
   <input type="password" name="password"
      placeholder="password">
   <button>login</button>
</form>

<h1>This is home page</h1>

<li><a href="/register">Sign up!!</a></li>
<li><a href="/login">Login</a></li>
<li><a href="/logout">Logout</a></li>
```

Js file:

```
var express = require("express"),
passport = require("passport"),
   bodyParser = require("body-parser"),
   LocalStrategy = require("passport-local"),
```

```
var app = express();
app.set("view engine", "ejs");
app.use(bodyParser.urlencoded({ extended: true }));

app.use(require("express-session")({
   secret: "Rusty is a dog",
   resave: false,
   saveUninitialized: false
}));

app.use(passport.initialize());
app.use(passport.session());

passport.use(new LocalStrategy(User.authenticate()));
passport.serializeUser(User.serializeUser());
passport.deserializeUser(User.deserializeUser());

//Handling user login
app.post("/login", passport.authenticate("local", {
   successRedirect: "/secret",
   failureRedirect: "/login"
}), function (req, res) {
});

//Handling user logout
app.get("/logout", function (req, res) {
   req.logout();
   res.redirect("/");
});

function isLoggedIn(req, res, next) {
   if (req.isAuthenticated()) return next();
   res.redirect("/login");
}

var port = process.env.PORT || 3000;
app.listen(port, function () {
   console.log("Server Has Started!");
});
```

14. using node is create a Recipe Book.

➔

```
cd chef-repo/cookbooks
 chef generate cookbook lamp_stack
cd lamp_stack
cd recipes
knife cookbook upload lamp_stack
package "apache2" do
  action :install
endpackage "apache2" do
  action :install
end
knife cookbook upload lamp_stack
knife node run_list add nodename "recipe[lamp_stack::apache]"
knife ssh 'name:nodename' 'systemctl status apache2' -x root
chef generate attribute ~/chef-repo/cookbooks/lamp_stack default
default["lamp_stack"]["sites"]["example.com"] = { "port" => 80, "servername" =>
"example.com", "serveradmin" => "webmaster@example.com" }
default["lamp_stack"]["sites"]["example.com"] = { "port" => 80, "servername" =>
"example.com", "serveradmin" => "webmaster@example.com" }
default["lamp_stack"]["sites"]["example.org"] = { "port" => 80, "servername" =>
"example.org", "serveradmin" => "webmaster@example.org" }
#Install & enable Apache
package "apache2" do
  action :install
end
service "apache2" do
  action [:enable, :start]
end
# Virtual Host Files
node["lamp_stack"]["sites"].each do |sitename, data|
end
node["lamp_stack"]["sites"].each do |sitename, data|
  document_root = "/var/www/html/#{sitename}"
end
node["lamp_stack"]["sites"].each do |sitename, data|
  document_root = "/var/www/html/#{sitename}"
  directory document_root do
    mode "0755"
    recursive true
  end
end
chef generate template ~/chef-repo/cookbooks/lamp_stack virtualhosts
[client]
```

```
knife cookbook site install mysql
mysqlpass = data_bag_item("mysql", "rtpass.json")

mysql_service "mysqldefault" do
 version '5.7'
 initial_root_password mysqlpass["password"]
 action [:create, :start]
end
socket=/run/mysql-mysqldefault/mysqld.sock
cookbook_file "/etc/my.cnf" do
 source "my.cnf"
 mode "0644"
end

package "php" do
 action :install
end

package "php-pear" do
 action :install
end

package "php-mysql" do
 action :install
end

package "libapache2-mod-php" do
 action :install
end
```

15. write node js script to interact with the filesystem. and serve a web page from a file

➔

```
var http = require('http');
var fs = require('fs');
fs.readFile('index.html', function (err, html) {
   if (err) {
      throw err;
   }
   http.createServer(function(request, response) {
      response.writeHeader(200, {"Content-Type": "text/html"});
      response.write(html);
      response.end();
   }).listen(1337, '127.0.0.1');
});
```

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset='utf-8'>
    <title>Node.js test</title>
    <link rel="stylesheet" media="screen" type="text/css" href="css/plugin.css" />
    <link rel="stylesheet" media="screen" type="text/css" href="css/xGrid.css" />
    <link rel="stylesheet" media="screen" type="text/css" href="css/jquery-ui/jquery-ui-
1.10.1.custom.min.css" />
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.6/jquery.min.js"></script>
    <script src="https://ajax.googleapis.com/ajax/libs/jqueryui/1.8/jquery-ui.min.js"></script>
    <script src="js/slate.js"></script>
    <script src="js/slate.portlet.js"></script>
    <script src="js/slate.message.js"></script>
    <script src="js/plugin.js"></script>
  </head>
  <body>
    <h1 class="styled-h1">Test</h1>
  </body>
</html>

fs.readFile('index.html', function (err, html) {
  if (err) {
    throw err;
  }
  http.createServer(function(request, response) {
    response.writeHeader(200, {"Content-Type": "text/html"});  // <-- HERE!
    response.write(html);  // <-- HERE!
    response.end();
  }).listen(1337, '127.0.0.1');
});
var filePath = req.url;
if (filePath == '/')
 filePath = '/index.html';

filePath = __dirname+filePath;
var extname = path.extname(filePath);
var contentType = 'text/html';
switch (extname) {
   case '.js':
      contentType = 'text/javascript';
      break;
   case '.css':
```

```
        contentType = 'text/css';
        break;
}
fs.exists(filePath, function(exists) {

   if (exists) {
      fs.readFile(filePath, function(error, content) {
         if (error) {
            res.writeHead(500);
            res.end();
         }
         else {
            res.writeHead(200, { 'Content-Type': contentType });
            res.end(content, 'utf-8');
         }
      });
   }
```

16. Write node js script to build Your Own Node.js Module. Use require (http) module us a built-in Node module that invokes the functionality of the HTTP library to create a local server. Also use the export statement to make functions in your module available externally. Create a new text file to contain the functions in your module called. "modules.js" and add this function to return n av's date and time.

➔

```
var http = require('http');
http.createServer(function (req, res) {
 res.writeHead(200, {'Content-Type': 'text/html'});
 res.end('Hello World!');
}).listen(8080);
 exports.myDateTime = function () {
  return Date();
};
 var http = require('http');
 var dt = require('./myfirstmodule');

 http.createServer(function (req, res) {
   res.writeHead(200, {'Content-Type': 'text/html'});
   res.write("The date and time are currently: " + dt.myDateTime());
   res.end();
 }).listen(8080);
```

19.Write node js application that transfer a file as an attachment on web and enable browser to prompt the user to download file using express js.

➔

```
   <form action="fileupload" method="post" enctype="multipart/form-data">
   <input type="file" name="filetoupload">
   <input type="submit" value="Upload">
</form>
```

```
http.createServer(function (req, res) {
   if (req.url == '/uploadform') {
      // if request URL contains '/uploadform'
      // fill the response with the HTML file containing upload form
   } else if (req.url == '/fileupload') {
      // if request URL contains '/fileupload'
      // using formiddable module,
      // read the form data (which includes uploaded file)
      // and save the file to a location.
   }
}).listen(8086);
```

```
var form = new formidable.IncomingForm();
form.parse(req, function (err, fields, files) {
   // oldpath : temporary folder to which file is saved to
   var oldpath = files.filetoupload.path;
   var newpath = upload_path + files.filetoupload.name;
   // copy the file to a new location
   fs.rename(oldpath, newpath, function (err) {
      if (err) throw err;
      // you may respond with another html page
      res.write('File uploaded and moved!');
      res.end();
   });
});
<!DOCTYPE html>
<html>
<head>
<title>Upload File</title>
<style>
   body{text-align:center;}
   form{display:block;border:1px solid black;padding:20px;}
</style>
</head>
<body>
   <h1>Upload files to Node.js Server</h1>
```

```
    <form action="fileupload" method="post" enctype="multipart/form-data">
     <input type="file" name="filetoupload">
     <input type="submit" value="Upload">
   </form>
</body>
</html
```

```
var http = require('http');
var fs = require('fs');
var formidable = require('formidable');
 // html file containing upload form
var upload_html = fs.readFileSync("upload_file.html");
 // replace this with the location to save uploaded files
var upload_path = "/home/arjun/workspace/nodejs/upload_file/";

http.createServer(function (req, res) {
   if (req.url == '/uploadform') {
     res.writeHead(200);
     res.write(upload_html);
     return res.end();
   } else if (req.url == '/fileupload') {
       var form = new formidable.IncomingForm();
       form.parse(req, function (err, fields, files) {
          // oldpath : temporary folder to which file is saved to
          var oldpath = files.filetoupload.path;
          var newpath = upload_path + files.filetoupload.name;
          // copy the file to a new location
          fs.rename(oldpath, newpath, function (err) {
             if (err) throw err;
             // you may respond with another html page
             res.write('File uploaded and moved!');
             res.end();
          });
       });
   } }).listen(8086);
```

20.Create your Django app in which after running the server, you should see on the browser, the text "Hello! I am learning Django", which you defined in the index view

➔

```
    $ python -m django --version
   $ django-admin startproject mysite
   $ python manage.py runserver
   $ python manage.py startapp polls
```

```
from django.http import HttpResponse
def index(request):
   return HttpResponse("Hello! I am learning Django.")
from django.urls import path
from . import views
urlpatterns = [
   path('', views.index, name='index'),
]
from django.contrib import admin
from django.urls import include, path
urlpatterns = [
   path('polls/', include('polls.urls')),
   path('admin/', admin.site.urls),
]
```

21. Design a Django application that adds web pages with views and templates.

➔

```
 import os
# Build paths inside the project like this: os.path.join(BASE_DIR,...)
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
Template_DIR = os.path.join(BASE_DIR,'Templates')
TEMPLATES = [
{
'BACKEND': 'django.template.backends.django.DjangoTemplates',
'DIRS': [Template_DIR,],
'APP_DIRS': True,
'OPTIONS': {
'context_processors': [
'django.template.context_processors.debug',
'django.template.context_processors.request',
'django.contrib.auth.context_processors.auth',
'django.contrib.messages.context_processors.messages',
,
},
},
]Template_DIR = os.path.join(BASE_DIR,'Templates')

<!DOCTYPE html>
<html lang="en" dir="ltr">
<head>
<meta charset="utf-8">
<title>Django App1</title>
</head>
```

```
    <body>
    <h1> Hello world from HTML page <h1>
    </body>
    </html>

    from django.shortcuts import render
    from django.http import  HttpResponse
    def index(request_iter):
    return  render(request_iter,'design.html')

    from django.contrib import admin
    from django.conf.urls import url
    from Django_app1 import views
    urlpatterns = [
    url(r'^$',views.index,name='index'),
    url(r'admin/',admin.site.urls),]
```

```
<!DOCTYPE html>
<html lang="en" dir="ltr">
<head>
<meta charset="utf-8">
<title>Django App1</title>
</head>
<body>
<h1> <u> All valid Technical tutorials </u> </h1>
{% if Entity_type == 'tutorial' %}
{{ Entity_name }}
{% else %}
{{ Error_Message }}
{% endif %}
<h2> <u> Django filters Explained <u> </h2>
<p> Student Count: {{ Entity_students_count | add:"230"}}  <p>
<p> Entity Type: {{ Entity_type | capfirst}}  <p>
</body>
</html></html></html>
```

22. Write and run Django code to add data to your site using relational databases Django's Object Relational Mapper.

➔

```
 from django.db import models
class Team(models.Model):
name = ...
class Player(models.Model):
height= ...
```

```
name = ...
team = ...
class Game(models.Model):
date = ...
home_team = ...
home_team_points = ...
rival_team = ...
rival_team_points = ...
class Team(models.Model):
name = models.CharField(max_length=64)
class Player(models.Model):
height = models.FloatField()
name = models.CharField(max_length=64)
team = models.ForeignKey(Team, on_delete=models.CASCADE
class Game(models.Model):
home_team = models.ForeignKey(Team, related_name='game_at_home',
on_delete=models.CASCADE)
home_team_points = models.IntegerField()
rival_team = models.ForeignKey(Team, related_name='rival_game',
on_delete=models.CASCADE)
rival_team_points = models.IntegerField()
date = models.DateField()
from django.db import models
from django.db.models import CharField, ForeignKey, IntegerField, CASCADE
class Team(models.Model):
name = models.CharField(max_length=64)
class Meta:
app_label = 'tournament'
class League(models.Model):
name = CharField(max_length=32)
champion = ForeignKey(Team, related_name='champion_of', on_delete=CASCADE)
number_of_teams = IntegerField()
class Meta:
app_label = 'tournament'
```

23. Develop a basic poll application (app). it should consist of two parts:

a) A public site in which user can pick their favorite programming language and votes.

b) An admin site that lets you add, change and delete programming languages.

➔

```
$ python -m django --version
$ django-admin startproject mysite
$ python manage.py runserver
$ python manage.py startapp polls
from django.http import HttpResponse
def index(request):
   return HttpResponse("Hello! I am learning Django.")
from django.urls import path
from . import views
urlpatterns = [path('', views.index, name='index'),]
from django.contrib import admin
from django.urls import include, path
urlpatterns = [
   path('polls/', include('polls.urls')),
   path('admin/', admin.site.urls),]
from django.db import models


class Question(models.Model):
   question_text = models.CharField(max_length=200)
   pub_date = models.DateTimeField('date published')

  class Choice(models.Model):
   question = models.ForeignKey(Question, on_delete=models.CASCADE)
   choice_text = models.CharField(max_length=200)
   votes = models.IntegerField(default=0)
$ python manage.py makemigrations polls

$ python manage.py sqlmigrate polls 0001

BEGIN;
--
-- Create model Question
--
CREATE TABLE "polls_question" (
   "id" serial NOT NULL PRIMARY KEY,
   "question_text" varchar(200) NOT NULL,
   "pub_date" timestamp with time zone NOT NULL
);
--
```

```
-- Create model Choice
--
CREATE TABLE "polls_choice" (
    "id" serial NOT NULL PRIMARY KEY,
    "choice_text" varchar(200) NOT NULL,
    "votes" integer NOT NULL,
    "question_id" integer NOT NULL
);
ALTER TABLE "polls_choice"
  ADD CONSTRAINT "polls_choice_question_id_c5b4b260_fk_polls_question_id"
    FOREIGN KEY ("question_id")
    REFERENCES "polls_question" ("id")
    DEFERRABLE INITIALLY DEFERRED;
CREATE INDEX "polls_choice_question_id_c5b4b260" ON "polls_choice" ("question_id");

COMMIT;

$ python manage.py shell

from django.db import models

class Question(models.Model):
    # ...
    def __str__(self):
        return self.question_text

class Choice(models.Model):
    # ...
    def __str__(self):
return self.choice_text

import datetime

from django.db import models
from django.utils import timezone


class Question(models.Model):
    # ...
    def was_published_recently(self):
        return self.pub_date >= timezone.now() - datetime.timedelta(days=1)

>>> from polls.models import Choice, Question
```

```
# Make sure our __str__() addition worked.
>>> Question.objects.all()
<QuerySet [<Question: What's up?>]>

# Django provides a rich database lookup API that's entirely driven by
# keyword arguments.
>>> Question.objects.filter(id=1)
<QuerySet [<Question: What's up?>]>
>>> Question.objects.filter(question_text__startswith='What')
<QuerySet [<Question: What's up?>]>

# Get the question that was published this year.
>>> from django.utils import timezone
>>> current_year = timezone.now().year
>>> Question.objects.get(pub_date__year=current_year)
<Question: What's up?>

# Request an ID that doesn't exist, this will raise an exception.
>>> Question.objects.get(id=2)
Traceback (most recent call last):
    ...
DoesNotExist: Question matching query does not exist.

# Lookup by a primary key is the most common case, so Django provides a
# shortcut for primary-key exact lookups.
# The following is identical to Question.objects.get(id=1).
>>> Question.objects.get(pk=1)
<Question: What's up?>

# Make sure our custom method worked.
>>> q = Question.objects.get(pk=1)
>>> q.was_published_recently()
True

# Give the Question a couple of Choices. The create call constructs a new
# Choice object, does the INSERT statement, adds the choice to the set
# of available choices and returns the new Choice object. Django creates
# a set to hold the "other side" of a ForeignKey relation
# (e.g. a question's choice) which can be accessed via the API.
>>> q = Question.objects.get(pk=1)

# Display any choices from the related object set -- none so far.
>>> q.choice_set.all()
<QuerySet []>
```

```
    # Create three choices.
    >>> q.choice_set.create(choice_text='Not much', votes=0)
    <Choice: Not much>
    >>> q.choice_set.create(choice_text='The sky', votes=0)
    <Choice: The sky>
    >>> c = q.choice_set.create(choice_text='Just hacking again', votes=0)

    # Choice objects have API access to their related Question objects.
    >>> c.question
    <Question: What's up?>

    # And vice versa: Question objects get access to Choice objects.
    >>> q.choice_set.all()
    <QuerySet [<Choice: Not much>, <Choice: The sky>, <Choice: Just hacking again>]>
    >>> q.choice_set.count()
    3

    # The API automatically follows relationships as far as you need.
    # Use double underscores to separate relationships.
    # This works as many levels deep as you want; there's no limit.
    # Find all Choices for any question whose pub_date is in this year
    # (reusing the 'current_year' variable we created above).
    >>> Choice.objects.filter(question__pub_date__year=current_year)
    <QuerySet [<Choice: Not much>, <Choice: The sky>, <Choice: Just hacking again>]>

    # Let's delete one of the choices. Use delete() for that.
    >>> c = q.choice_set.filter(choice_text__startswith='Just hacking')
    >>> c.delete()
```

26.implement a simple Django application for portfolio management.

➔

```
    INSTALLED_APPS = (
  'django.contrib.auth'
  'django.contrib.contenttypes'
  'django.contrib.sessions',
  'django.contrib.sites',
     'djangorocks.blog',
)
class Blog(models.Model):
  title = models.CharField(max_length=100, unique=True)
  slug = models.SlugField(max_length=100, unique=True)
  body = models.TextField()
  posted = models.DateField(db_index=True, auto_now_add=True)
```

```
  category = models.ForeignKey('blog.Category')
class Category(models.Model):
  title = models.CharField(max_length=100, db_index=True)
  slug = models.SlugField(max_length=100, db_index=True)
class Blog(models.Model):
title = models.CharField(max_length=100, db_index=True)
slug = models.SlugField(max_length=100, db_index=True)
body = models.TextField()
posted = models.DateTimeField(db_index=True, auto_now_add=True)
category = models.ForeignKey('blog.Category')
from django.db import models
from django.db.models import permalink

# Create your models here.
class Blog(models.Model):
  title = models.CharField(max_length=100, unique=True)
  slug = models.SlugField(max_length=100, unique=True)
  body = models.TextField()
  posted = models.DateTimeField(db_index=True, auto_now_add=True)
  category = models.ForeignKey('blog.Category')

  def __unicode__(self):
    return '%s' % self.title

  @permalink
  def get_absolute_url(self):
    return ('view_blog_post', None, { 'slug': self.slug })

class Category(models.Model):
  title = models.CharField(max_length=100, db_index=True)
  slug = models.SlugField(max_length=100, db_index=True)
  def __unicode__(self):
    return '%s' % self.title
  @permalink
  def get_absolute_url(self):
    return ('view_blog_category', None, { 'slug': self.slug })
return ('view_blog_post', None, { 'slug': self.slug })
from django.contrib import admin
from blog.models import Blog, Category

admin.site.register(Blog)
admin.site.register(Category)
from django.contrib import admin
from blog.models import Blog, Category
```

```
admin.site.register(Blog)
admin.site.register(Category)
from django.contrib import admin
from djangorocks.blog.models import Blog, Category

class BlogAdmin(admin.ModelAdmin):
  exclude = ['posted']
  prepopulated_fields = {'slug': ('title',)}

class CategoryAdmin(admin.ModelAdmin):
  prepopulated_fields = {'slug': ('title',)}

admin.site.register(Blog, BlogAdmin)
admin.site.register(Category, CategoryAdmin)
# Create your views here.

from djangorocks.blog.models import Blog, Category
from django.shortcuts import render_to_response, get_object_or_404

def index(request):
  return render_to_response('index.html', {
    'categories': Category.objects.all(),
    'posts': Blog.objects.all()[:5]
  })

def view_post(request, slug):
  return render_to_response('view_post.html', {
    'post': get_object_or_404(Blog, slug=slug)
  })

def view_category(request, slug):
  category = get_object_or_404(Category, slug=slug)
  return render_to_response('view_category.html', {
    'category': category,
    'posts': Blog.objects.filter(category=category)[:5]
  })
from djangorocks.blog.models import Blog, Category
from django.shortcuts import render_to_response, get_object_or_404
def view_category(request, slug):
render_to_response('test.html', {
'categories': Category.objects.all(),
'posts': Blog.objects.all()[:5]
get_object_or_404(Blog, slug=slug)
(r'^$', 'djangorocks.blog.views.index'),
```

```
url(
  r'^blog/view/(?P<slug>[^\.]+).html',
  'djangorocks.blog.views.view_post',
  name='view_blog_post'),
url(
  r'^blog/category/(?P<slug>[^\.]+).html',
  'djangorocks.blog.views.view_category',
  name='view_blog_category'),
(?P<slug>[^\.]+)
```

base.html

```
<html>
  <head>
    <title>{% block head_title %}Welcome to my blog{% endblock %}</title>
  </head>
  <body>
    <h1>{% block title %}Welcome to my block{% endblock %}</h1>
    {% block content %}

    {% endblock %}
  </body>
</html>
```

index.html

```
{% extends 'base.html' %}
{% block title %}Welcome to my blog{% endblock %}

{% block content %}
  <h2>Categories</h2>
  {% if categories %}
    <ul>
    {% for category in categories %}
      <li><a href="{{ category.get_absolute_url }}">{{ category.title }}</a></li>
    {% endfor %}
    </ul>
  {% else %}
    <p>There are no posts.</p>
  {% endif %}

  <h2>Posts</h2>
  {% if posts %}
    <ul>
    {% for post in posts %}
      <li><a href="{{ post.get_absolute_url }}">{{ post.title }}</a></li>
```

```
    {% endfor %}
    </ul>
  {% else %}
    <p>There are no posts.</p>
  {% endif %}

{% endblock %}
```

view_post.html

```
{% extends 'base.html' %}
{% block head_title %}{{ post.title }}{% endblock %}
{% block title %}{{ post.title }}{% endblock %}

{% block content %}
  {{ post.body }}
{% endblock %}
```

view_category.html

```
{% extends 'base.html' %}
{% block head_title %}Viewing category {{ category.title }}{% endblock %}
{% block title %}{{ category.title }}{% endblock %}
{% block content %}
  {% if posts %}
    <ul>
    {% for post in posts %}
      <li><a href="{{ post.get_absolute_url }}">{{ post.title }}</a></li>
    {% endfor %}
    </ul>
  {% else %}
    <p>There are no posts.</p>
  {% endif %}
{% endblock %}
```

28. Build your own To-Do app in Django.

➔

```
django-admin startproject todoproject
cd todoproject
python manage.py runserver
python manage.py startapp todoapp
mkdir templates
<h1>My To Do List</h1>
from django.shortcuts import render
def todoappView(request):
   return render(request, 'todolist.html')
from django.contrib import admin
```

```
from django.urls import path
from todoapp.views import todoappView

urlpatterns = [
   path('admin/', admin.site.urls),
   path('todoapp/', todoappView),
]
from django.db import models
class TodoListItem(models.Model):
   content = models.TextField()
python manage.py makemigrations
python manage.py migrate
from .models import TodoListItem
def todoappView(request):
   all_todo_items = TodoListItem.objects.all()
   return render(request, 'todolist.html',
   {'all_items':all_todo_items})
<ul>
   {% for i in all_items %}
   <li>{{i.content}}
   </li>
   {% endfor %}
</ul>
<form action="/addTodoItem/" method = "post">{% csrf_token %}
   <input type="text" name="content">
   <input type="submit" value="Add Todo Item">
</form>
def addTodoView(request):
   x = request.POST['content']
   new_item = TodoListItem(content = x)
   new_item.save()
   return HttpResponseRedirect('/todoapp/')
from django.http import HttpResponseRedirect
path('addTodoItem/',addTodoView),
from todoapp.views import todoappView, addTodoView
<form action="/deleteTodoItem/{{i.id}}/" method = "post">{% csrf_token %}
        <input type="submit" value="Delete">
</form>
path('deleteTodoItem/<int:i>/', deleteTodoView),
from todoapp.views import todoappView, addTodoView, deleteTodoView
def deleteTodoView(request, i):
   y = TodoListItem.objects.get(id= i)
   y.delete()
   return HttpResponseRedirect('/todoapp/')
```

29. Create a clone of the "Hacker News" website.

➔

npx create-react-app hackernews-clone-react-app

yarn add axios@0.21.0 bootstrap@4.6.0 node-sass@4.14.1 react-bootstrap@1.4.0 react-router-dom@5.2.

```
import React from 'react';
import { NavLink } from 'react-router-dom';

const Header = () => {
  return (
    <React.Fragment>
      <h1>Hacker News Clone</h1>
      <div className="nav-link">
        <NavLink to="/top" activeClassName="active">
          Top Stories
        </NavLink>
        <NavLink to="/new" activeClassName="active">
          Latest Stories
        </NavLink>
        <NavLink to="/best" activeClassName="active">
          Best Stories
        </NavLink>
      </div>
    </React.Fragment>
  );
};

export default Header;

import React from 'react';

const HomePage = () => {
  return <React.Fragment>Home Page</React.Fragment>;
};

export default HomePage;

import React from 'react';
import { Link } from 'react-router-dom';

const PageNotFound = () => {
```

```
  return (
    <p>
      Page Not found. Go to <Link to="/">Home</Link>
    </p>
  );
};

export default PageNotFound;

import React from 'react';
import { BrowserRouter, Route, Switch } from 'react-router-dom';
import Header from '../components/Header';
import HomePage from '../components/HomePage';
import PageNotFound from '../components/PageNotFound';

const AppRouter = () => {
  return (
    <BrowserRouter>
      <div className="container">
        <Header />
        <Switch>
          <Route path="/" component={HomePage} exact={true} />
          <Route component={PageNotFound} />
        </Switch>
      </div>
    </BrowserRouter>
  );
};

export default AppRouter;
import React from 'react';
import ReactDOM from 'react-dom';
import AppRouter from './router/AppRouter';
import 'bootstrap/dist/css/bootstrap.min.css';
import './styles.scss';

ReactDOM.render(<AppRouter />, document.getElementById('root'));
```

30. Develop Online School System using Django.

➔

```
django-admin startproject OnlineSchoolMgmt
cd OnlineSchoolMgmt
django-admin startapp app
```

```
from django.db import models
from django.contrib.auth.models import User
# Create your models here.
class Attendance(models.Model):
StudentName = models.CharField(max_length=200,null=True)
StudentId = models.CharField(max_length=50,null=True)
LecturesAttended = models.IntegerField(null=True)
TotalLectures = models.IntegerField(null=True)
def __str__(self):
return self.StudentName
class Marks(models.Model):
StudentName = models.CharField(max_length=200,null=True)
StudentId = models.CharField(max_length=50,null=True)
PhysicsMarks = models.IntegerField(null=True)
ChemistryMarks = models.IntegerField(null=True)
MathsMarks = models.IntegerField(null=True)
EnglishMarks = models.IntegerField(null=True)
ComputerMarks = models.IntegerField(null=True)
def __str__(self):
return self.StudentName
class Notice(models.Model):
Message = models.CharField(max_length=200,null=True)
date_created = models.DateTimeField(auto_now_add=True)
def __str__(self):
return self.Message

from django.forms import ModelForm
from .models import *
from django.contrib.auth.forms import UserCreationForm
from django.contrib.auth.models import User
class createuserform(UserCreationForm):
class Meta:
model=User
fields=['username','password']
class addAttendanceform(ModelForm):
class Meta:
model=Attendance
fields="__all__"
class addMarksform(ModelForm):
class Meta:
model=Marks
fields="__all__"
class addNoticeform(ModelForm):
class Meta:
model=Notice
fields="__all__"

from django.contrib import admin
```

```
from .models import *
# Register your models here.
admin.site.register(Atttendance)
admin.site.register(Notice)
admin.site.register(Marks

py manage.py createsuperuser

"""OnlineSchoolMgmt URL Configuration
The `urlpatterns` list routes URLs to views. For more information please see:
https://docs.djangoproject.com/en/3.1/topics/http/urls/
Examples:
Function views
1. Add an import: from my_app import views
2. Add a URL to urlpatterns: path('', views.home, name='home')
Class-based views
1. Add an import: from other_app.views import Home
2. Add a URL to urlpatterns: path('', Home.as_view(), name='home')
Including another URLconf
1. Import the include() function: from django.urls import include, path
2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))
"""
from django.contrib import admin
from django.urls import path
from app.views import*
urlpatterns = [
path('admin/', admin.site.urls),
path('',home ,name='home'),
path('addAttendance/', addAttendance,name='addAttendance'),
path('addMarks/', addMarks,name='addMarks'),
path('addNotice/', addNotice,name='addNotice'),
path('login/', loginPage,name='login'),
path('logout/', logoutPage,name='logout'),
path('register/', registerPage,name='register'),
]

from django.shortcuts import redirect,render
from django.contrib.auth import login,logout,authenticate
from django.http import HttpResponse
from .forms import *
def home(request):
notice = Notice.objects.all()
attendance = Attendance.objects.all()
marks = Marks.objects.all()
context = {
'notice':notice,
'marks':marks,
'attendance':attendance,
```

```
}
return render(request,'app/home.html',context)
def addAttendance(request):
if request.user.is_authenticated:
form=addAttendanceform()
if(request.method=='POST'):
form=addAttendanceform(request.POST)
if(form.is_valid()):
form.save()
return redirect('/')
context={'form':form}
return render(request,'app/addAttendance.html',context)
else:
return redirect('home')
def addMarks(request):
if request.user.is_authenticated:
form=addMarksform()
if(request.method=='POST'):
form=addMarksform(request.POST)
if(form.is_valid()):
form.save()
return redirect('/')
context={'form':form}
return render(request,'app/addMarks.html',context)
else:
return redirect('home')
def addNotice(request):
if request.user.is_authenticated:
form=addNoticeform()
if(request.method=='POST'):
form=addNoticeform(request.POST)
if(form.is_valid()):
form.save()
return redirect('/')
context={'form':form}
return render(request,'app/addNotice.html',context)
else:
return redirect('home')
def registerPage(request):
if request.user.is_authenticated:
return redirect('home')
else:
form=createuserform()
if request.method=='POST':
form=createuserform(request.POST)
if form.is_valid() :
user=form.save()
return redirect('login')
```

```
context={
'form':form,
}
return render(request,'app/register.html',context)
def loginPage(request):
if request.user.is_authenticated:
return redirect('home')
else:
if request.method=="POST":
username=request.POST.get('username')
password=request.POST.get('password')
user=authenticate(request,username=username,password=password)
if user is not None:
login(request,user)
return redirect('/')
context={}
return render(request,'app/login.html',context)
def logoutPage(request):
logout(request)
return redirect('/')
```

```
{% extends 'app/Links.html' %}
{% block content %}
<div class="container ">
<br><br>
<div class="row">
<div class="col-md-9">
<h5>Notice:</h5>
<div class="card card-body">
<table class="table table-sm">
<tr>
<th>Notice</th>
<th>Date</th>
</tr>
{% for n in notice %}
<tr>
<td>{{n.Message}} </td>
<td>{{n.date_created}} </td>
</tr>
{% endfor %}
</table>
</div>
</div>
</div>
<br><br>
<div class="row">
<div class="col-md-9">
```

```
<h5>Attendance:</h5>
<div class="card card-body">
<table class="table table-sm">
<tr>
<th>Student Id</th>
<th>Student Name</th>
<th>Attended Lectures</th>
<th>Total Lectures</th>
</tr>
{% for a in attendance %}
<tr>
<td>{{a.StudentId}} </td>
<td>{{a.StudentName}} </td>
<td>{{a.LecturesAttended}} </td>
<td> {{a.TotalLectures}} </td>
</tr>
{% endfor %}
</table>
</div>
</div>
</div>
<br><br>
<div class="row">
<div class="col-md-9">
<h5>Marks:</h5>
<div class="card card-body">
<table class="table table-sm">
<tr>
<th>Student Id</th>
<th>Student Name</th>
<th>Physics Marks</th>
<th>Chemistry Marks</th>
<th>Maths Marks</th>
<th>English Marks</th>
<th>Computer Marks</th>
</tr>
{% for m in marks %}
<tr>
<td>{{m.StudentId}} </td>
<td>{{m.StudentName}} </td>
<td>{{m.PhysicsMarks}} </td>
<td>{{m.ChemistryMarks}} </td>
<td>{{m.MathsMarks}} </td>
<td>{{m.EnglishMarks}} </td>
<td>{{m.ComputerMarks}} </td>
</tr>
{% endfor %}
</table>
```

```
<p> <b>Note:</b> Marks are out of 100</p>
</div>
</div>
</div>
{% endblock %}

{% extends 'app/Links.html' %}
{% block content %}
<div class="jumbotron container row">
<div class="col-md-6">
<h1>Add Attendance</h1>
<div class="card card-body">
<form action="" method="POST">
{% csrf_token %}
{{form.as_p}}
<br>
<input type="submit" name="Submit">
</form>
</div>
</div>
</div>
</div>
{% endblock %}


{% extends 'app/Links.html' %}
{% block content %}
<div class="jumbotron container row">
<div class="col-md-6">
<h1>Add Notice</h1>
<div class="card card-body">
<form action="" method="POST">
{% csrf_token %}
{{form.as_p}}
<br>
<input type="submit" name="Submit">
</form>
</div>
</div>
</div>
</div>
{% endblock %}


{% extends 'app/Links.html' %}
{% block content %}
<div class="jumbotron container row">
<div class="col-md-6">
```

```
<h1>Add Marks</h1>
<div class="card card-body">
<form action="" method="POST">
{% csrf_token %}
{{form.as_p}}
<br>
<input type="submit" name="Submit">
</form>
</div>
</div>
</div>
</div>
{% endblock %}


{% extends 'app/Links.html' %}
{% load static%}
{% block content %}
<div class="container jumbotron">
<form method="POST" action="">
{% csrf_token %}
<p><input type="text" name="username" placeholder="Username..."></p>
<p><input type="password" name="password" placeholder="Password..." ></p>
<input class="btn btn-success" type="submit" value="Login">
<p>Do not have an account<a href='{% url 'register' %}'>Register</a></p>
</form>
</div>
{% endblock %}


% extends 'app/Links.html' %}
{% load static%}
{% block content %}
<div class="container jumbotron">
<form method="POST" action="" >
{% csrf_token %}
{{form.as_p}}
<input class="btn btn-success" type="submit" value="Register Account">
</form>
</div>
{% endblock %}


{% load static %}
<html>
<head>
<title>
TechVidvan Online School Management Project
```

```
</title>
<link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css" integrity="sha384-
9aIt2nRpC12Uk9gS9baDl411NQApFmC26EwAOH8WgZl5MYYxFfc+NcPb1dKGj7Sk"
crossorigin="anonymous">
</head>
<body>
{% include 'app/navbar.html' %}
{% block content %}
{% endblock %}
<br>
<script src="https://code.jquery.com/jquery-3.5.1.slim.min.js" integrity="sha384-
DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj"
crossorigin="anonymous"></script>
<script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js"
integrity="sha384-Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfooAo"
crossorigin="anonymous"></script>
<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.min.js"
integrity="sha384-OgVRvuATP1z7JjHLkuOU7Xw704+h835Lr+6QL9UvYjZE3Ipu6Tp75j7Bh/kR0JKI"
crossorigin="anonymous"></script>
</body>
</html>

{% load static %}
<style>
.greet{
font-size: 18px;
color: #fff;
margin-right: 20px;
}
</style>
<nav class="navbar navbar-expand-lg navbar-dark bg-dark">
<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">
<li class="nav-item active">
<a class="nav-link" href="{% url 'home' %}">Home</a>
</li>
{% if request.user.is_staff %}
</li>
<li class="nav-item active">
<a class="nav-link" href="{% url 'addAttendance' %}">Add Attendance</a>
</li>
<li class="nav-item active">
<a class="nav-link" href="{% url 'addNotice' %}">Add Notice</a>
```

```
</li>
<li class="nav-item active">
<a class="nav-link" href="{% url 'addMarks' %}">Add Marks</a>
</li>
{% endif %}
<li class="nav-item">
<a class="nav-link" href="{% url 'login' %}">Login</a>
</li>
</ul>
</div>
{% if request.user.is_staff %}
<span class="greet">Hello, {{request.user}}</span>
<span ><a class="greet" href="{% url 'logout' %}">Logout</a></span>
{% endif %}
</nav>
```

31. Implement your E-commerce Website using Django

➔

```
 $ conda create -name <my_env>
$ conda activate <my_env>
$ conda deactivate <my_env>
$ pip install Django
$ django-admin –version
$ django-admin startproject <project_name>
$ cd <project_name>
$ python manage.py migrate
$ python manage.py makemigrations
$ python manage.py createsuperuser
$ python manage.py startapp core
python manage.py runserver
$ code core/urls.p
$ mkdir templates
$ mkdir staic

from django.shortcuts import render
        from django.http import HttpResponse


        # Create your views here.


        def home(request) :
           return render(request, 'home.html')
INSTALLED_APPS = [
            'django.contrib.admin',
```

```
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',


    'django.contrib.sites',
    'allauth',
    'allauth.account',
    'allauth.socialaccount',

    'core'
]


#Static
STATIC_URL = '/static/'
STATICFILES_DIRS = [
    os.path.join(BASE_DIR, 'static')
]
STATIC_ROOT = os.path.join(BASE_DIR, 'assets')

# Auth
AUTHENTICATION_BACKENDS = (
    'django.contrib.auth.backends.ModelBackend',
    'allauth.account.auth_backends.AuthenticationBackend'
)
SITE_ID = 1


LOGIN_REDIRECT_URL = '/'
LOGOUT_REDIRECT_URL = '/'
```

```
$ python manage.py collectstatic
```

```
from django.urls import path
from . import views app_name = 'core' urlpatterns = [
    path('', views.home, name='home')
]
```

```
from django.contrib import admin
from django.urls import path, include urlpatterns = [
    path('', include('core.urls', namespace='core')),
```

```
    path('admin/', admin.site.urls),
    path('accounts/', include('django.contrib.auth.urls'))
]
```

32. Implement Login System using Django.

➔

```
 $ cd ~/Desktop
$ mkdir accounts && cd accounts
$ pipenv install django~=3.1.0
$ pipenv shell
(accounts) $ django-admin.py startproject config .
(accounts) $ python manage.py migrate
(accounts) $ python manage.py runserver
# config/settings.py
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth', # Yoohoo!!!!
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
]
# config/urls.py
from django.contrib import admin
from django.urls import path, include # new

urlpatterns = [
    path('admin/', admin.site.urls),
    path('accounts/', include('django.contrib.auth.urls')), # new
]


accounts/login/ [name='login']
accounts/logout/ [name='logout']
accounts/password_change/ [name='password_change']
accounts/password_change/done/ [name='password_change_done']
accounts/password_reset/ [name='password_reset']
accounts/password_reset/done/ [name='password_reset_done']
accounts/reset/<uidb64>/<token>/ [name='password_reset_confirm']
accounts/reset/done/ [name='password_reset_complete']
```

```
(accounts) $ mkdir templates
(accounts) $ mkdir templates/registration
(accounts) $ touch templates/registration/login.html

<!-- templates/registration/login.html -->
<h2>Log In</h2>
<form method="post">
  {% csrf_token %}
  {{ form.as_p }}
  <button type="submit">Log In</button>
</form>
# config/settings.py
TEMPLATES = [
    {
        ...
        'DIRS': [str(BASE_DIR.joinpath('templates'))],
        ...
    },
]
# config/settings.py
LOGIN_REDIRECT_URL = '/'
(accounts) $ python manage.py createsuperuser
Username (leave blank to use 'wsv'):
Email address: will@learndjango.com
Password:
Password (again):
Superuser created successfully.

accounts) $ touch templates/base.html
(accounts) $ touch templates/home.html

<!-- templates/base.html -->
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>{% block title %}Django Auth Tutorial{% endblock %}</title>
</head>
<body>
  <main>
```

```
    {% block content %}
    {% endblock %}
  </main>
</body>
</html>


<!-- templates/home.html -->
{% extends 'base.html' %}

{% block title %}Home{% endblock %}

{% block content %}
{% if user.is_authenticated %}
  Hi {{ user.username }}!
{% else %}
  <p>You are not logged in</p>
  <a href="{% url 'login' %}">Log In</a>
{% endif %}
{% endblock %}

<!-- templates/registration/login.html -->
{% extends 'base.html' %}

{% block title %}Login{% endblock %}

{% block content %}
<h2>Log In</h2>
<form method="post">
  {% csrf_token %}
  {{ form.as_p }}
  <button type="submit">Log In</button>
</form>
{% endblock %}

# config/urls.py
from django.contrib import admin
from django.urls import path, include
from django.views.generic.base import TemplateView # new
```

```
urlpatterns = [
    path('admin/', admin.site.urls),
    path('accounts/', include('django.contrib.auth.urls')),
    path('', TemplateView.as_view(template_name='home.html'), name='home'), # new
]
```