MAHATMA GANDHI INSTITUTE OF TECHNOLOGY (Autonomous)
Kokapet(Village), Gandipet, Hyderabad, Telangana – 500075. www.mgit.ac.in
UGC Autonomous | Affiliation to | AICTE approved | Accredited by NAAC Grade: A++ | NBA

MOTIVATE INNOVATE EMPOWER 26 YEARS

CS753PE: Full Stack Development Lab (Professional Elective-III) (Common to CSE & IT)

Prepared By

Mr. G Nagi Reddy
Asst. Professor, CSE

Dr. P Shyam Sunder
Asst. Professor, CSE

Mr. Y.Pavan Narasimha Rao
Asst. Professor, CSE

L	T	P	C
0	0	2	1

VI Semester Syllabus

CS753PE: Full Stack Development Lab

(Professional Elective-III)

(Common to CSE & IT)

Pre-Requisites:

1. Object Oriented Programming
2. Web Technologies

Course Objectives:

- Introduce fast, efficient, interactive and scalable web applications using run time environmentprovided by the full stack components.

Course Outcomes:

- Design flexible and responsive Web applications using Node JS, React, Express and Angular.
- Perform CRUD operations with MongoDB on huge amount of data.
- Develop real time applications using react components.
- Use various full stack modules to handle http requests and responses.

List of Experiments

1. Create an application to setup node JS environment and display "Hello World".
2. Create a Node JS application for user login system.
3. Write a Node JS program to perform read, write and other operations on a file.
4. Write a Node JS program to read form data from query string and generate response usingNodeJS
5. Create a food delivery website where users can order food from a particular restaurant listed in the website for handling http requests and responses using NodeJS.
6. Implement a program with basic commands on databases and collections using MongoDB.
7. Implement CRUD operations on the given dataset using MongoDB.
8. Perform Count, Limit, Sort, and Skip operations on the given collections using MongoDB.
9. Develop an angular JS form to apply CSS and Events.
10. Develop a Job Registration form and validate it using angular JS.

11. Write an angular JS application to access JSON file data of an employee from a server using $http service.
12. Develop a web application to manage student information using Express and Angular JS.
13. Write a program to create a simple calculator Application using React JS.
14. Write a program to create a voting application using React JS
15. Develop a leave management system for an organization where users can apply different typesof leaves such as casual leave and medical leave. They also can view the available number ofdays using react application.
16. Build a music store application using react components and provide routing among the web pages.
17. Create a react application for an online store which consist of registration, login, product information pages and implement routing to navigate through these pages.

TEXT BOOKS:

1. Brad Dayley, Brendan Dayley, Caleb Dayley., Node.js, MongoDB and Angular Web Development, 2nd Edition, Addison-Wesley,2019.
2. Mark Tielens Thomas., React in Action, 1st Edition, Manning Publications.

REFERENCE BOOKS:

1. Vasan Subramanian, Pro MERN Stack, Full Stack Web App Development with Mongo, Express, React, and Node, 2nd Edition, Apress,2019.
2. Chris Northwood, The Full Stack Developer: Your Essential Guide to the Everyday Skills Expected of a Modern Full Stack Web Developer', 1st edition, Apress, 2018.
3. Brad Green& Seshadri. Angular JS. 1st Edition. O'Reilly Media, 2013.
4. Kirupa Chinnathambi, Learning React: A Hands-On Guide to Building Web Applications Using React and Redux, 2nd edition, Addison-Wesley Professional, 2018.

EXP1.

AIM: Create an application to setup node JS environment and display "Hello World."

hello1.js

```
http = require('http');
listener = function (req, res) {
   res.writeHead(200, {'Content-Type': 'text/html'});
     res.end('Hello World.....');
};

server = http.createServer(listener);
server.listen(8080);
```

output:

>node hello1.js

hello2.js

```
//Create an application to setup node JS environment and display "Hello World." on
webpage
var http = require('http');
 http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/html'});
  res.write('<h1>hello world....</h1>');
     res.end();

   }).listen(8080);
```

output:

node hello1.js

EXP2.

AIM: Write a java script to find the factorial of a given number

```
function factorial(num) {
// Base case
 if (num === 0) {
 return 1;
 }
 // Recursive case
 return num * factorial(num - 1);
 }
 console.log(factorial(5));
```

Output:

120

EXP3.

AIM: Write a java script to check the given number is prime or not

```
// program to check if a number is prime or not
// take input from the user
const prompt = require('prompt-sync')();
 const number = parseInt(prompt("Enter a positive number: "));
 let isPrime = true;
 // check if number is equal to 1
 if (number === 1) {
 console.log("1 is neither prime nor composite number.");
 }
 // check if number is greater than 1
 else if (number > 1) {
 // looping through 2 to number-1
 for (let i = 2; i < number; i++) {
 if (number % i == 0) {
 isPrime = false;
 break;
 }
}
 if (isPrime) {
 console.log(`${number} is a prime number`);
 } else {
 console.log(`${number} is a not prime number`);
 }
 }
```

OUTPUT:

Enter a positive number: 11

11 is a prime number

Exp4.

Aim: Write a Node JS program to reverse the given string

```
// program to reverse a string

const prompt = require('prompt-sync')();

 function reverseString(str) {

// empty string

 let newString = "";

 for (let i = str.length - 1; i >= 0; i--) {

 newString += str[i];

 }

 return newString;

 }

// take input from the user

 const string = prompt('Enter a string: ');

 const result = reverseString(string);

 console.log(result)
```

Output:

Enter a string: mgit

tigm

EXP5.

Aim: Write a Node JS program to perform read, write and other operations on a file.

readFile.js

```
//READ FILE
var http = require('http');
var fs = require('fs');
http.createServer(function (req, res) {
  fs.readFile('file1.txt', function(err, data) {
    res.writeHead(200, {'Content-Type': 'text/html'});
    res.write(data);
    return res.end();
  });
}).listen(8080);
```

File1.txt

```
WELCOME TO MGIT CSE1 FULL STACK DEVELOPMENT
```

Output:

file1 content:WELCOME TO MGIT CSE1 FULL STACK DEVELOPMENT

writeFile.js

```
//WRITE FILE ASYNCHRONOUS CALL
var fs = require('fs');
fs.writeFile('file1.txt', 'This is my text', function (err) {
  if (err) throw err;
  console.log('Replaced!');
});
```

Output:

Replaced!

appendFile.js

```
//APPEND FILE ASYNCHRONOUS CALL

var fs = require('fs');

fs.appendFile('file1.txt', 'FULL STACK DEVELOPMENT', function (err) {

  if (err) throw err;

  console.log('Saved!');

});
```

Output:

Saved!

Deletefile.js

```
//DELETE FILE

var fs = require('fs');

fs.unlink('file2.txt', function (err) {

  if (err) throw err;

  console.log('File deleted!');

});
```

Output:

'File deleted!

EXP6.

Aim: Write a Node JS program to demonstrate user defined modules.

Calc.js

```
//calc functions
exports.add = function (x, y) {
    return x + y;
};
exports.sub = function (x, y) {
   return x - y;
};
exports.mult = function (x, y) {
    return x * y;
};
exports.div = function (x, y) {
    return x / y;
};
```

Module.js

```
//user defined module

const calculator = require('./calc.js');

let x = 50, y = 10;

console.log("Addition of 50 and 10 is"    + calculator.add(x, y));

console.log("Subtraction of 50 and 10 is "  + calculator.sub(x, y));

console.log("Multiplication of 50 and 10 is "+ calculator.mult(x, y));

console.log("Division of 50 and 10 is " + calculator.div(x, y));
```

output:

Addition of 50 and 10 is60

Subtraction of 50 and 10 is 40

Multiplication of 50 and 10 is 500

Division of 50 and 10 is 5

Exp7:

Aim: Write a Node JS program to demonstrate accessing static files using http webserver and client.

Httpstaticfileserver.js

```
//Implementing a basic static file webserver
var fs = require('fs');
var http = require('http');
var url = require('url');
var ROOT_DIR = "html/";
http.createServer(function (req, res) {
var urlObj = url.parse(req.url, true, false);
fs.readFile(ROOT_DIR + urlObj.pathname, function (err,data) {
if (err) {
res.writeHead(404);
res.end(JSON.stringify(err));
return;
}
res.writeHead(200);
res.end(data);
});
}).listen(8080)
```

In termimal1:

modules>node httpstaticfileserver.js

Output:

Httpstaticfileclient.js

```
// Basic web client retrieving static files
var http = require('http');
var options = {
hostname: 'localhost',
 port: '8080',
 path: '/hello.html'
 };
 function handleResponse(response) {
 var serverData = '';
 response.on('data', function (chunk) {
 serverData += chunk;
 });
 response.on('end', function () {
 console.log(serverData);
 });
 }
 http.request(options, function(response){
 handleResponse(response);
 }).end()
```

html/hello.html

```
<html>
 <head>
 <title>Static Example</title>
 </head>
 <body>
 <h1>Hello from a Static File</h1>
 </body>
</html>
```

Output:

In terminal2:

modules>node httpstaticfileclient.js

```
<html>
 <head>
 <title>Static Example</title>
 </head>
 <body>
 <h1>Hello from a Static File</h1>
 </body>
</html>
```

Exp8:

Aim: Write a Node JS program to demonstrate accessing dynamic content through GET method using http webserver and client.

Httpserverget.js

```
//Implementing a basic GET webserver

var http = require('http');

var messages = [

 'Hello World',

 'From a basic Node.js server',

 'Take Luck'];

 http.createServer(function (req, res) {

 res.setHeader("Content-Type", "text/html");

 res.writeHead(200);

 res.write('<html><head><title>Simple HTTP Server</title></head>');

 res.write('<body>');

 for (var idx in messages){

 res.write('\n<h1>' + messages[idx] + '</h1>');

 }

 res.end('\n</body></html>');

 }).listen(8080)
```

Output: In terminal1:

Modules>node httpserverget.js

Httpsclientget.js

```
// Basic web client retrieving
var http = require('http');
var options = {
hostname: 'localhost',
 port: '8080',
  };
 function handleResponse(response) {
 var serverData = '';
 response.on('data', function (chunk) {
 serverData += chunk;
 });
 response.on('end', function () {
    console.log("Response Status:", response.statusCode);
    console.log("Response Headers:", response.headers);
    console.log(serverData);
 });
 }
 http.request(options, function(response){
 handleResponse(response);
 }).end()
```

Output: In Terminal2:

```
modules> node httpclientget.js
Response Status: 200
Response Headers: {
  'content-type': 'text/html',
  date: 'Sat, 30 Nov 2024 10:39:02 GMT',
  connection: 'keep-alive',
  'keep-alive': 'timeout=5',
  'transfer-encoding': 'chunked'
}
<html><head><title>Simple HTTP Server</title></head><body>
<h1>Hello World</h1>
<h1>From a basic Node.js server</h1>
<h1>Take Luck</h1>
</body></html>
```

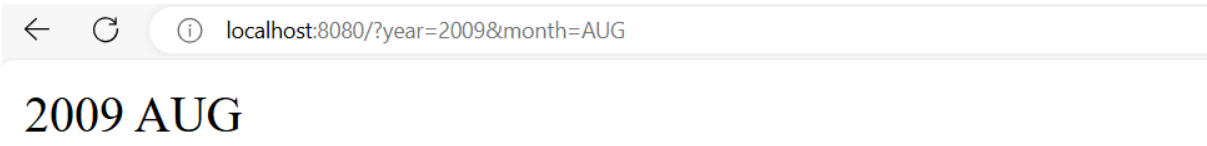
Exp9:

AIM: write a Node JS program to read form data from query string and generate response using NodeJS

Modules/url5.js

```
//write a Node JS program to read form data from query string and generate response using NodeJS

var http = require('http');

var url = require('url');

http.createServer(function (req, res) {

  res.writeHead(200, {'Content-Type': 'text/html'});

  var q= url.parse(req.url,true).query;

  console.log(q);

  var txt = q.year + " " + q.month;

  res.end(txt);

}).listen(8080);
```

OUTPUT:



Exp10:

AIM: write a Node JS program to demonstrate events and call back functions.

//events.js

```
// Importing events
const EventEmitter = require('events');
// Initializing event emitter instances
var eventEmitter = new EventEmitter();
// Registering to myEvent
eventEmitter.on('myEvent', (msg) => {
  console.log(msg);
});
// Triggering myEvent
eventEmitter.emit('myEvent', "First event");
// Triggering myEvent
eventEmitter.emit('myEvent', "Second event");
console.log("program ended...")
```

output:

node events1.js

First event

Second event

program ended...

//callback.js

```
// call back function

var fs= require('fs')

fs.writeFile('file.txt',' welcome to call back functions', function()

   {

   console.log(" data written to file.txt")

})

console.log('End of the program.....')
```

output:

End of the program.....

data written to file.txt

file.text

```
call back functions
```

Exp11: Implement a program with basic commands on databases and collections using MongoDB.

MONGODB COMMANDS (CRUD OPERATIONS):
C-CREATE
R-READ/RETRIVE
U-UPDATE
D-DELETE

1. D:\MONGODB\DB>mongod --version
db version v8.0.0
Build Info: {
  "version": "8.0.0",
  "gitVersion": "d7cd03b239ac39a3c7d63f7145e91aca36f93db6",
  "modules": [],
  "allocator": "tcmalloc-gperf",
  "environment": {
    "distmod": "windows",
    "distarch": "x86_64",
    "target_arch": "x86_64"
  }
}
2.D:\MONGODB\DB>mongosh
Current Mongosh Log ID: 66f252c9808c7f3e6bc73bf7
Connecting to:
mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.3.1
Using MongoDB: 8.0.0
Using Mongosh: 2.3.1

For mongosh info see: https://www.mongodb.com/docs/mongodb-shell/

  The server generated these startup warnings when booting
  2024-09-23T14:31:32.621+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
---

test>
3.test> show dbs
admin 40.00 KiB
config 72.00 KiB
local 72.00 KiB

4.test> use MYDB1
switched to db MYDB1
MYDB1> show dbs
admin 40.00 KiB
config 72.00 KiB
local 72.00 KiB

5.MYDB1> db.createCollection("students")
{ ok: 1 }

6.MYDB1> show dbs
MYDB1 8.00 KiB
admin 40.00 KiB
config 108.00 KiB
local 72.00 KiB

7.MYDB1> db.students.insertOne({"rollno":501 , "name":"cse1"})
```
{
  acknowledged: true,
  insertedId: ObjectId('66f255ec808c7f3e6bc73bf8')
}
```

8.MYDB1> show collections
Students

9.MYDB1> db.students.find().pretty()
```
[
  {
    _id: ObjectId('66f255ec808c7f3e6bc73bf8'),
    rollno: 501,
    name: 'cse1'
  }
]
```

10.MYDB1> db.students.insertOne({"rollno":502 , "name":"cse2"})
```
{
  acknowledged: true,
  insertedId: ObjectId('66f2577b808c7f3e6bc73bf9')
}
```

11.MYDB1> db.students.find().pretty()
```
[
  {
    _id: ObjectId('66f255ec808c7f3e6bc73bf8'),
    rollno: 501,
    name: 'cse1'
  },
  {
    _id: ObjectId('66f2577b808c7f3e6bc73bf9'),
    rollno: 502,
    name: 'cse2'
  }
]
```

12.MYDB1> db.students.updateOne({rollno:502},{$set:{name:"cse3"}})

```
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

13.MYDB1> db.students.find().pretty()

```
[
  {
    _id: ObjectId('66f255ec808c7f3e6bc73bf8'),
    rollno: 501,
    name: 'cse1'
  },
  {
    _id: ObjectId('66f2577b808c7f3e6bc73bf9'),
    rollno: 502,
    name: 'cse3'
  }
]
```

14. MYDB1> db.students.deleteOne({rollno:111})

```
{ acknowledged: true, deletedCount: 1 }

MYDB1> db.students.find().pretty()
[
  {
    _id: ObjectId('670ca9a53fede232f9c73bf9'),
    rollno: 222,
    name: 'cccccc'
  }
]
```

15.MYDB1> db.students.drop()

True

16.MYDB1> show collections

17.MYDB1> db.dropDatabase()

```
{ ok: 1, dropped: 'MYDB1' }
```

18MYDB1> show dbs

```
admin    40.00 KiB
config  108.00 KiB
local    72.00 KiB
```

<u>inserting documents from java scripts:</u>

db1.js

```
db.students.insertOne({name: "Karthik", rollno: 101, marks: 98 })
db.students.insertOne({name: "Ravi", rollno: 102, marks: 99 })
db.students.insertOne({name: "Shiva", rollno: 103, marks: 100 })
db.students.insertOne({name: "Pavan", rollno: 104, marks: 80 })
```

```
MYDB1> load('d:\db1.js')
True

MYDB1> db.students.find().pretty()
[
  {
    _id: ObjectId('670ca9a53fede232f9c73bf9'),
    rollno: 222,
    name: 'cccccc'
  },
  {
    _id: ObjectId('670ded507a61ecab52c73bf8'),
    name: 'Karthik',
    rollno: 101,
    marks: 98
  },
  {
    _id: ObjectId('670ded507a61ecab52c73bf9'),
    name: 'Ravi',
    rollno: 102,
    marks: 99
  },
  {
    _id: ObjectId('670ded507a61ecab52c73bfa'),
    name: 'Shiva',
    rollno: 103,
    marks: 100
  },
  {
    _id: ObjectId('670ded507a61ecab52c73bfb'),
    name: 'Pavan',
    rollno: 104,
    marks: 80
  }
]

MYDB1> db.students.findOne()
{
  _id: ObjectId('670ca9a53fede232f9c73bf9'),
  rollno: 222,
  name: 'cccccc'
}

MYDB1> db.students.findOne({'name':'cccccc'})
{
  _id: ObjectId('670ca9a53fede232f9c73bf9'),
  rollno: 222,
  name: 'cccccc'
}
```

Exp12: Implement CRUD operations on the given dataset using MongoDB.

<u>Adding the MongoDB Driver to Node.js</u>

1.npm install mongodb

added 12 packages, and audited 30 packages in 2s

found 0 vulnerabilities

Connect.js

```
// Connect to database and close the database connection
const { MongoClient } = require('mongodb')
// Create Instance of MongoClient for mongodb
const client = new MongoClient('mongodb://localhost:27017')
// Connect to database
client.connect()
    .then(() => {
        console.log('Connected Successfully!')

        //Close the database connection
        console.log('Exiting..')
        client.close()
    })
    .catch(error => console.log('Failed to connect!', error))
```

<u>Output:</u>
MONGODB> node connect.js

Connected Successfully!
Exiting..

<u>Insertdb.js</u>

```
// to insert one document
const { MongoClient } = require('mongodb')
// Create Instance of MongoClient for mongodb
const client = new MongoClient('mongodb://localhost:27017')

// Insert to database
client.db('MYDB').collection('students').insertOne({
    name: 'cse1',
    email: 'cse1@example.com'
})
    .then((res) => {
        console.log(res)
        client.close()
    })
    .catch((err) => console.log(err))
```

Output:

MONGODB> node insertdb.js

```
{
  acknowledged: true,
  insertedId: new ObjectId('674aeea8b3bc707da2d4559f')
}
```

Finddb.js

```
//to find one document
const { MongoClient } = require('mongodb')
// Create Instance of MongoClient for mongodb
const client = new MongoClient('mongodb://localhost:27017')

// Insert to database
client.db('MYDB').collection('students')
    .findOne({name:'cse1'})
    .then((res) => {
        console.log(res)
        client.close()
    })
    .catch((err) => console.log(err))
```

Output:

MONGODB>node finddb.js

```
{
  _id: new ObjectId('674aeea8b3bc707da2d4559f'),
  name: 'cse1',
  email: 'cse1@example.com'
}
```

Updatedb.js

```
//to update one document
const { MongoClient } = require('mongodb')
// Create Instance of MongoClient for mongodb
const client = new MongoClient('mongodb://localhost:27017')

// Insert to database
client.db('MYDB').collection('students')
    .updateOne({ name: 'cse1' },
        {
            $set:
                { email: 'cse123@example.com' }
        })
    .then((res) => {
        console.log(res)
        client.close()
    })
    .catch((err) => console.log(err))
```

Output:

MONGODB> node updatedb.js

```
{
  acknowledged: true,
  modifiedCount: 1,
  upsertedId: null,
  upsertedCount: 0,
  matchedCount: 1
}
```

Deletedb.js

```
//to delete one document
const { MongoClient } = require('mongodb')

// Create Instance of MongoClient for mongodb
const client = new MongoClient('mongodb://localhost:27017')

// Insert to database
client.db('MYDB').collection('students')
    .deleteOne({ name: 'cse1' })
    .then((res) => {
        console.log(res)
        client.close()
    })
    .catch((err) => console.log(err))
```

Output:

MONGODB> node deletedb.js

```
{ acknowledged: true, deletedCount: 1 }
```

Exp13: Perform Count, Limit, Sort, and Skip operations on the given collections using MongoDB.

```
test> use MYDB2
switched to db MYDB2

MYDB2> db.createCollection("employees");
{ ok: 1 }

MYDB2> db.employees.insertMany([{'id':111, 'name':'aaaa', 'salary':10000},{'id':222,
'name':'bbbb', 'salary':30000},{'id':333, 'name':'cccc', 'salary':20000},{'id':444,
'name':'dddd', 'salary':10000}])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('6713c7d9b34f42f350c73bfc'),
    '1': ObjectId('6713c7d9b34f42f350c73bfd'),
    '2': ObjectId('6713c7d9b34f42f350c73bfe'),
    '3': ObjectId('6713c7d9b34f42f350c73bff')
  }
}

MYDB2> db.employees.find().pretty()
[
  {
    _id: ObjectId('6713c7d9b34f42f350c73bfc'),
    id: 111,
    name: 'aaaa',
    salary: 10000
  },
  {
    _id: ObjectId('6713c7d9b34f42f350c73bfd'),
    id: 222,
    name: 'bbbb',
    salary: 30000
  },
  {
    _id: ObjectId('6713c7d9b34f42f350c73bfe'),
    id: 333,
    name: 'cccc',
    salary: 20000
  },
  {
    _id: ObjectId('6713c7d9b34f42f350c73bff'),
    id: 444,
    name: 'dddd',
    salary: 10000
  }
]
```

```
MYDB2> db.employees.find().count()
4

MYDB2> db.employees.find({salary:10000}).count()
2


MYDB2> db.employees.find().pretty().limit(1)
[
  {
    _id: ObjectId('6713c7d9b34f42f350c73bfc'),
    id: 111,
    name: 'aaaa',
    salary: 10000
  }
]

MYDB2> db.employees.find().pretty().limit(2)
[
  {
    _id: ObjectId('6713c7d9b34f42f350c73bfc'),
    id: 111,
    name: 'aaaa',
    salary: 10000
  },
  {
    _id: ObjectId('6713c7d9b34f42f350c73bfd'),
    id: 222,
    name: 'bbbb',
    salary: 30000
  }
]

MYDB2> db.employees.find().pretty().skip(2)
[
  {
    _id: ObjectId('6713c7d9b34f42f350c73bfe'),
    id: 333,
    name: 'cccc',
    salary: 20000
  },
  {
    _id: ObjectId('6713c7d9b34f42f350c73bff'),
    id: 444,
    name: 'dddd',
    salary: 10000
  }
]

MYDB2> db.employees.find().pretty().skip(3)
```

```
[
  {
    _id: ObjectId('6713c7d9b34f42f350c73bff'),
    id: 444,
    name: 'dddd',
    salary: 10000
  }
]
```

MYDB2> db.employees.find().pretty().sort({id:1})

```
[
  {
    _id: ObjectId('6713c7d9b34f42f350c73bfc'),
    id: 111,
    name: 'aaaa',
    salary: 10000
  },
  {
    _id: ObjectId('6713c7d9b34f42f350c73bfd'),
    id: 222,
    name: 'bbbb',
    salary: 30000
  },
  {
    _id: ObjectId('6713c7d9b34f42f350c73bfe'),
    id: 333,
    name: 'cccc',
    salary: 20000
  },
  {
    _id: ObjectId('6713c7d9b34f42f350c73bff'),
    id: 444,
    name: 'dddd',
    salary: 10000
  }
]
```

MYDB2> db.employees.find().pretty().sort({id:-1})

```
[
  {
    _id: ObjectId('6713c7d9b34f42f350c73bff'),
    id: 444,
    name: 'dddd',
    salary: 10000
  },
  {
    _id: ObjectId('6713c7d9b34f42f350c73bfe'),
    id: 333,
    name: 'cccc',
```

```
    salary: 20000
  },
  {
    _id: ObjectId('6713c7d9b34f42f350c73bfd'),
    id: 222,
    name: 'bbbb',
    salary: 30000
  },
  {
    _id: ObjectId('6713c7d9b34f42f350c73bfc'),
    id: 111,
    name: 'aaaa',
    salary: 10000
  }
]
```

Exp.14:

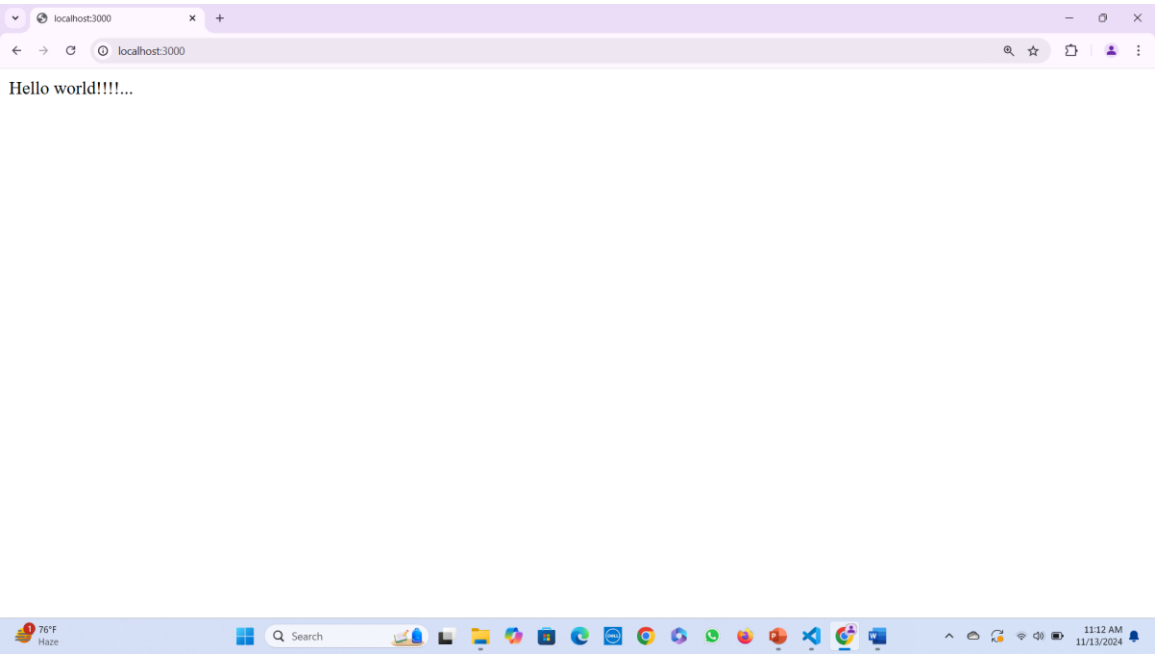
AIM: Program to print the msg ‘Hello world!’ using Express JS
Note: use the following commands to create the application:
>npm init
>npm install -g express

Index.js

```
// to print the msg ‘Hello world!’
var express = require('express');
var app = express();
const PORT=3000
app.get('/', function(req, res){
res.send("Hello world!");
});
app.listen(PORT,()=>
{
  console.log('server is running at port:'+PORT)
});
```

Output:
>node index.js

Localhost:3000

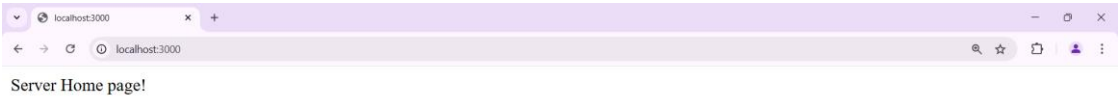


Exp.15:

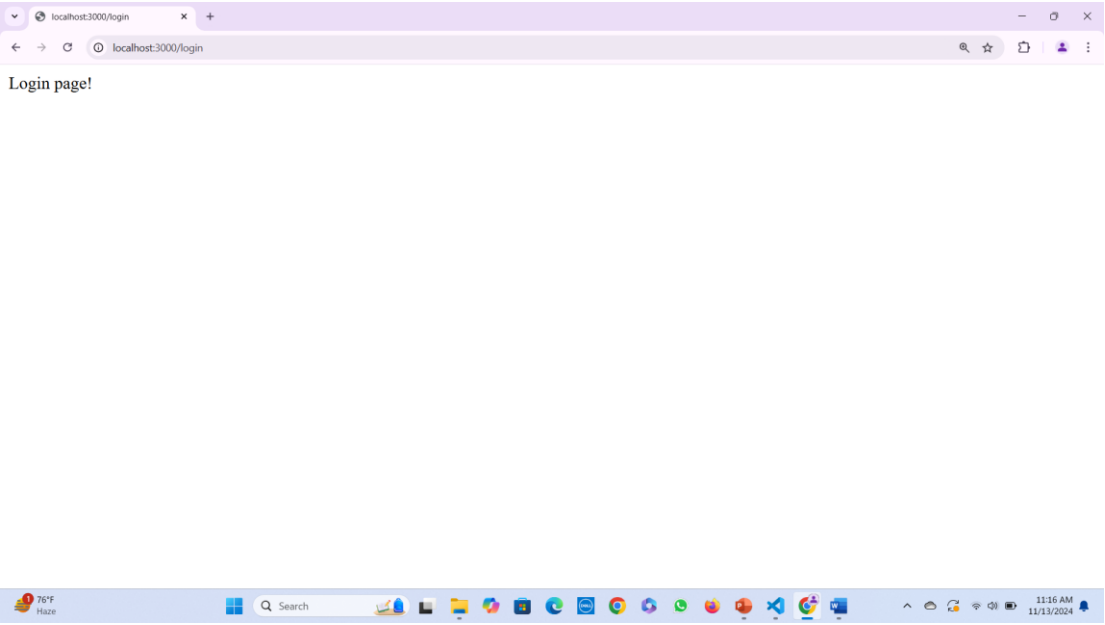
AIM: Program to demonstrate configuring and implementing routes using Express JS
Index.js

```
// to demonstrate configuring and implementing routes using Express Js
var express = require('express');
var app = express();
const PORT=3000
app.get('/', function(req, res){
   res.send("Server Home page!");
   });
app.get('/login', function(req, res){
res.send("Login page!");
});
app.get('/save', function(req, res){
   res.send("Save page!");
   });
app.listen(PORT,()=>
{
   console.log('server is running at port:'+PORT)
});
```

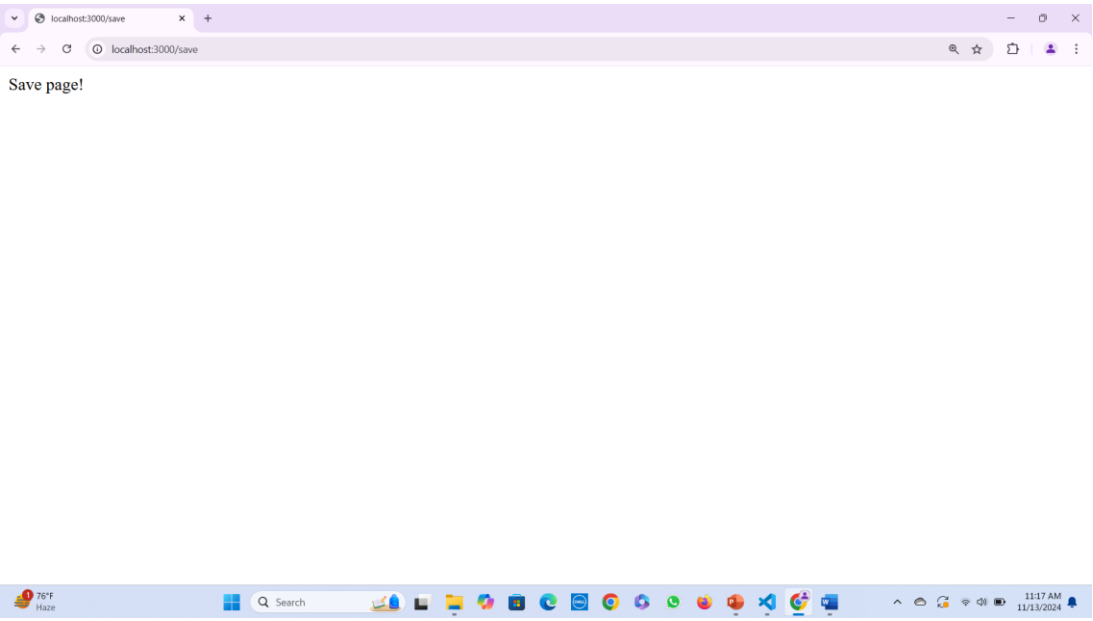
Output:
Localhost:3000



Localhost:3000/login



Localhost:3000/save



Exp.16:

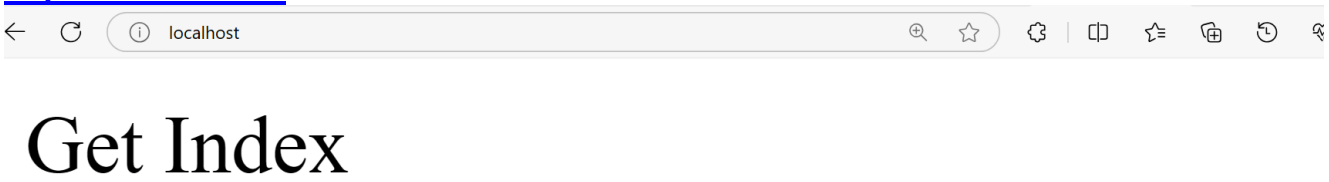
AIM: Program to demonstrate Applying Route Parameters in Express JS

Index.js

```
//to demonstrate Applying Route Parameters in Express JS
var express = require('express');
 var url = require('url');
var app = express();
 app.listen(4000);
app.get('/', function (req, res) {
 res.send("Get Index");
 });
 app.get('/find', function(req, res){
 var url_parts = url.parse(req.url, true);
 var query = url_parts.query;
 var response = 'Finding Book: Author: ' + query.author +
' Title: ' + query.title;
console.log('\nQuery URL: ' + req.originalUrl);
console.log(response);
res.send(response);
 });
 app.get(/^\/book\/(\w+)\:(\w+)?$/, function(req, res){
    var response = 'Get Book: Chapter: ' + req.params[0] +
     ' Page: ' + req.params[1];
     console.log('\nRegex URL: ' + req.originalUrl);
     console.log(response);
     res.send(response);
     });
     app.get('/user/:userid', function (req, res) {
        res.send("Get User: " + req.param("userid"));
       });
```

Output:

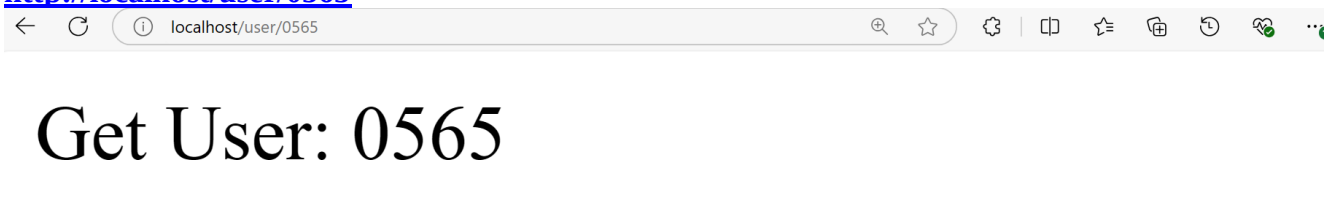
http://localhost:4000



http://localhost/find?author=shyam&title=FSD

http://localhost/book/05:65

http://localhost/user/0565



Exp.17:

AIM: Commands For Angular Installation, Create and Running Angular Apps.

ANGULAR JS

- Install node.js from https://nodejs.org/en/download/
  - Check if node installed by typing node –v and npm -v on command prompt
- Installing TypeScript using npm
  - npm install –g typescript
- Installing the Angular CLI using npm
  - npm install -g @angular/cli
  - ng –v [to test if angular installed]
- Select editor of your choice to start creating angular apps
  - We will be using VSCode
  - Download from : https://code.visualstudio.com

D:\ANGULAR>npm install -g typescript

added 1 package in 1s

D:\ANGULAR>tsc -v
Version 5.6.3

D:\ANGULAR>npm install -g @angular/cli

added 266 packages in 54s

49 packages are looking for funding
  run `npm fund` for details

D:\ANGULAR>ng v

```
     _                      _             ____ _     ___
    / \   _ __   __ _ _   _| | __ _ _ __     / ___| |   |_ _|
   / △ \ | '_ \ / _` | | | |/ _` | '__|   | |   | |    | |
  / ___ \| | | (_| | |_| | (_| | |      | |___| |___ | |
 /_/   \_\_| |_|\__, |\__,_|_|\__,_|_|       \____|_____|___|
                |___/
```

```
Angular CLI: 18.2.10
Node: 20.15.0
Package Manager: npm 10.7.0
OS: win32 x64

Angular:
...
Package                      Version
---------------------------------------------------------
@angular-devkit/architect    0.1802.10 (cli-only)
@angular-devkit/core         18.2.10 (cli-only)
@angular-devkit/schematics   18.2.10 (cli-only)
@schematics/angular          18.2.10 (cli-only)
```

D:\ANGULAR>ng new angulardemo

```
Would you like to share pseudonymous usage data about this project with the Angular Team
at Google under Google's Privacy Policy at https://policies.google.com/privacy. For more
details and how to change this setting, see https://angular.dev/cli/analytics.

  yes

Thank you for sharing pseudonymous usage data. Should you change your mind, the following
command will disable this feature entirely:

   ng analytics disable --global

Global setting: enabled
Local setting: No local workspace configuration file.
Effective status: enabled
? Which stylesheet format would you like to use? CSS          [
https://developer.mozilla.org/docs/Web/CSS                    ]
? Do you want to enable Server-Side Rendering (SSR) and Static Site Generation (SSG/Prerendering)?
yes
CREATE angulardemo/angular.json (2857 bytes)
CREATE angulardemo/package.json (1282 bytes)
CREATE angulardemo/README.md (1100 bytes)
CREATE angulardemo/tsconfig.json (1045 bytes)
CREATE angulardemo/.editorconfig (331 bytes)
CREATE angulardemo/.gitignore (629 bytes)
CREATE angulardemo/tsconfig.app.json (504 bytes)
CREATE angulardemo/tsconfig.spec.json (449 bytes)
CREATE angulardemo/server.ts (1786 bytes)
```

```
CREATE angulardemo/.vscode/extensions.json (134 bytes)
CREATE angulardemo/.vscode/launch.json (490 bytes)
CREATE angulardemo/.vscode/tasks.json (980 bytes)
CREATE angulardemo/src/main.ts (256 bytes)
CREATE angulardemo/src/index.html (310 bytes)
CREATE angulardemo/src/styles.css (81 bytes)
CREATE angulardemo/src/main.server.ts (271 bytes)
CREATE angulardemo/src/app/app.component.html (20239 bytes)
CREATE angulardemo/src/app/app.component.spec.ts (960 bytes)
CREATE angulardemo/src/app/app.component.ts (320 bytes)
CREATE angulardemo/src/app/app.component.css (0 bytes)
CREATE angulardemo/src/app/app.config.ts (413 bytes)
CREATE angulardemo/src/app/app.routes.ts (80 bytes)
CREATE angulardemo/src/app/app.config.server.ts (361 bytes)
CREATE angulardemo/public/favicon.ico (15086 bytes)
√ Packages installed successfully.
'git' is not recognized as an internal or external command,
operable program or batch file.
```
PS D:\ANGULAR\angulardemo> ng serve

```
Would you like to share pseudonymous usage data about this project with the Angular
Team
at Google under Google's Privacy Policy at https://policies.google.com/privacy. For
more
details and how to change this setting, see https://angular.dev/cli/analytics.

Would you like to share pseudonymous usage data about this project with the Angular
Team
at Google under Google's Privacy Policy at https://policies.google.com/privacy. For
more
details and how to change this setting, see https://angular.dev/cli/analytics.

  yes

Thank you for sharing pseudonymous usage data. Should you change your mind, the following
command will disable this feature entirely:

   ng analytics disable

Global setting: enabled
Local setting: enabled
Effective status: enabled
Browser bundles
Initial chunk files     | Names              |  Raw size
polyfills.js            | polyfills          |  90.20 kB |
```

main.js | main | 22.79 kB |
styles.css | styles | 95 bytes |

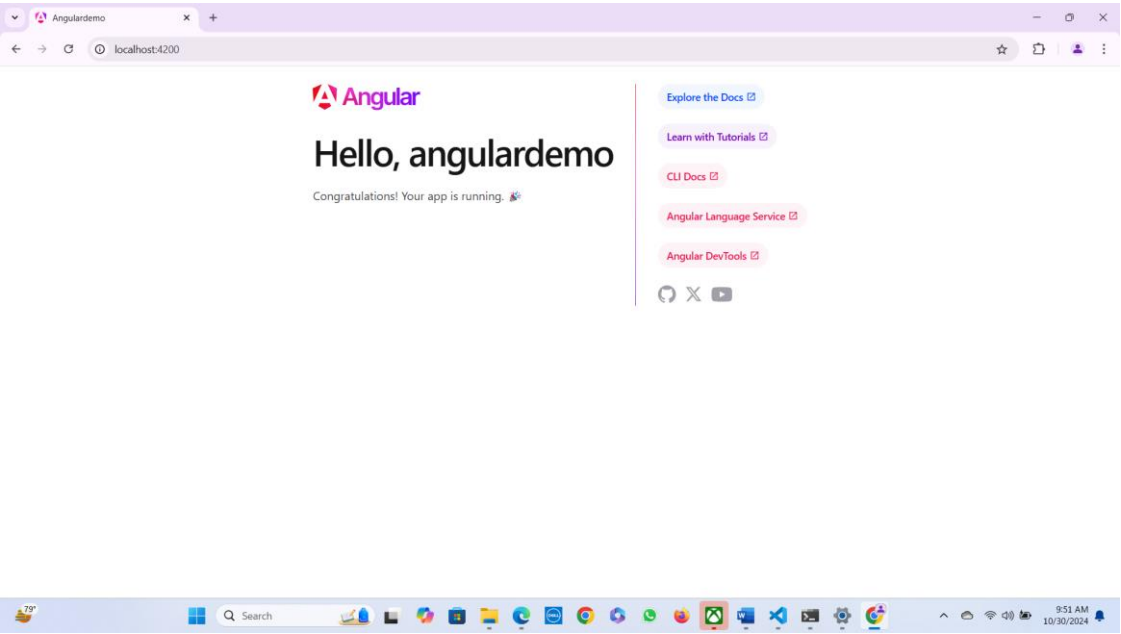
| Initial total | 113.08 kB

Server bundles
Initial chunk files | Names | Raw size
polyfills.server.mjs | polyfills.server | 572.91 kB |
Server bundles
Initial chunk files | Names | Raw size
polyfills.server.mjs | polyfills.server | 572.91 kB |
Initial chunk files | Names | Raw size
polyfills.server.mjs | polyfills.server | 572.91 kB |
polyfills.server.mjs | polyfills.server | 572.91 kB |
main.server.mjs | main.server | 23.23 kB |
main.server.mjs | main.server | 23.23 kB |
render-utils.server.mjs | render-utils.server | 472 bytes |
render-utils.server.mjs | render-utils.server | 472 bytes |

Application bundle generation complete. [3.384 seconds]
Watch mode enabled. Watching for file changes...
Watch mode enabled. Watching for file changes...
NOTE: Raw file sizes do not reflect development server per-request transformations.
NOTE: Raw file sizes do not reflect development server per-request transformations.

➜ Local: http://localhost:4200/
➜ press h + enter to show help

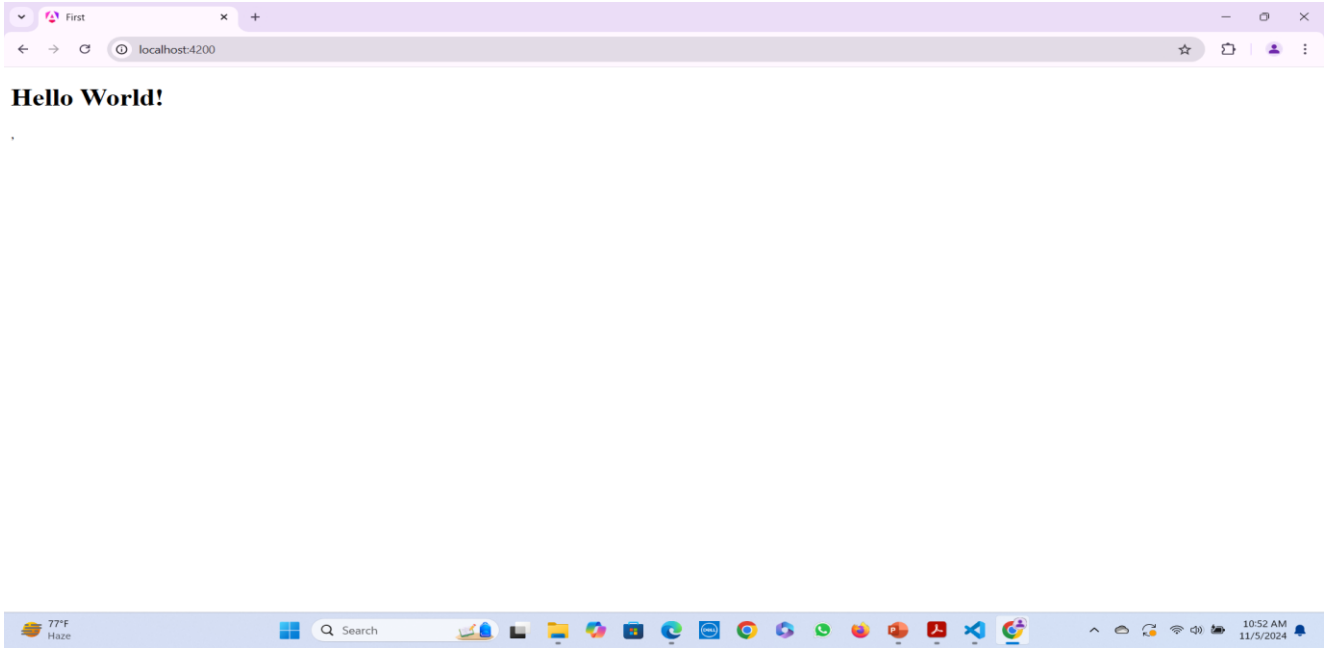
Localhost:4200



First/src/app/app.component.ts

```
//first angular application
import { Component } from '@angular/core';
@Component({
  selector: 'app-root',
  standalone: true,
  template: `
 <h1>Hello World!</h1>,
  `
})
export class AppComponent {
  title = 'My Fist Angular app';
}
```

Output:



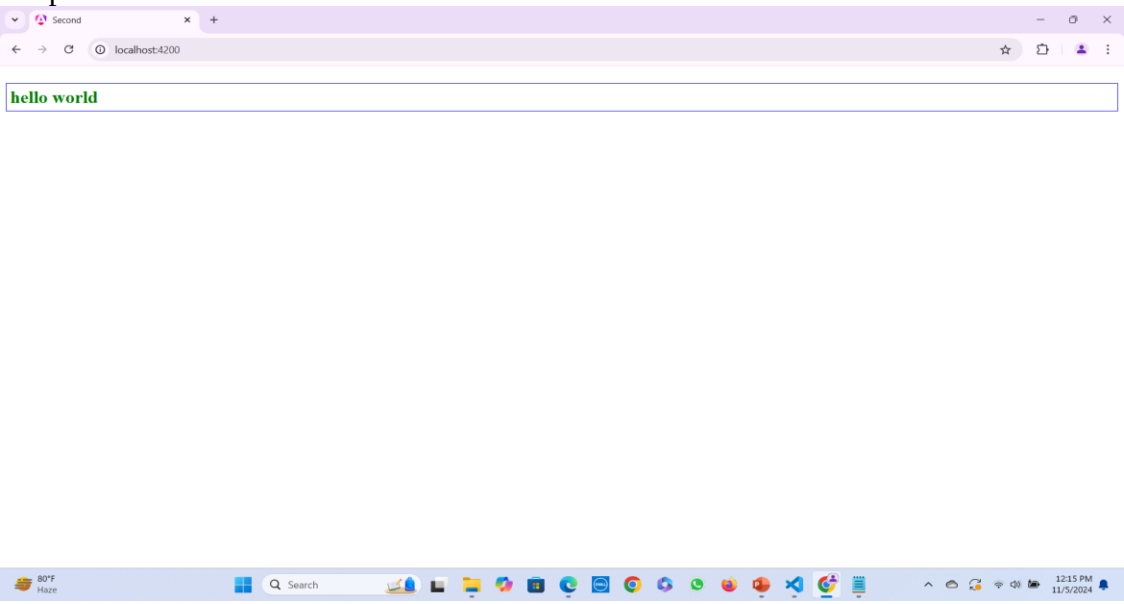
Exp.18

AIM: : angular component configuration

Second/src/app/app.component.ts

```
// angular component configuration
import { Component } from '@angular/core';
import { RouterOutlet } from '@angular/router';
@Component ({
  selector: 'app-root',
  standalone: true,
  template: '<p>hello world</p>',
  styles: [`
  P {
  color: green;
  font-size: 25px;
  font-weight: bold;
  border: 1px ridge blue;
 padding: 5px;
  }
  `]
  })
export class AppComponent {
  title = 'second';
}
```

Output:



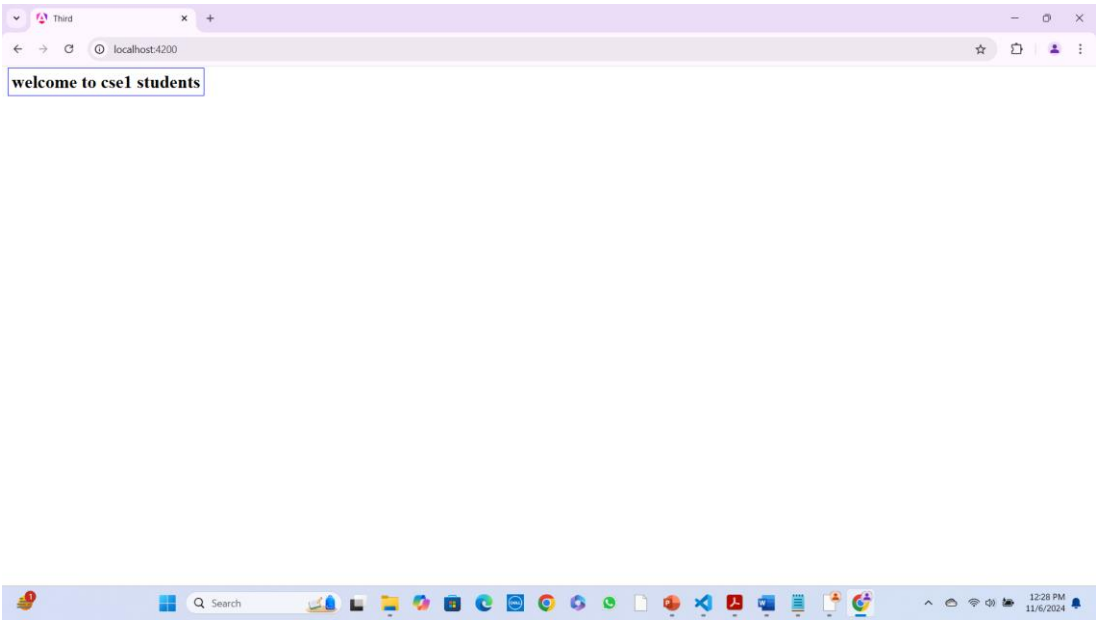
Exp.19

AIM: : To demonstrate Using Inline CSS and HTML in Angular Applications

Third/src/app/app.component.ts

```
//to demonstrate Using Inline CSS and HTML in Angular Applications
import { Component } from '@angular/core';
import { RouterOutlet } from '@angular/router';

 @Component({
 selector: 'app-root',
 standalone: true,
 template: `
 <span>welcome to cse1 students</span>
 `,
 styles:[`
 span {
 font-weight: bold;
 font-size: 25px;
 border: 1px ridge blue;
 padding: 5px;
 }
 `]
 })
 export class AppComponent {
 title = 'third';
 }
```

Output:



Exp.20

AIM: : To demonstrate Using external CSS and HTML in Angular Applications

Fourth/src/app/app.component.ts

```
import { Component } from '@angular/core';
import { RouterOutlet } from '@angular/router';

@Component({
  selector: 'app-root',
  standalone: true,
  imports: [RouterOutlet],
  templateUrl: './app.component.html',
  styleUrl: './app.component.css'
})
export class AppComponent {
  title = 'fourth';
}
```

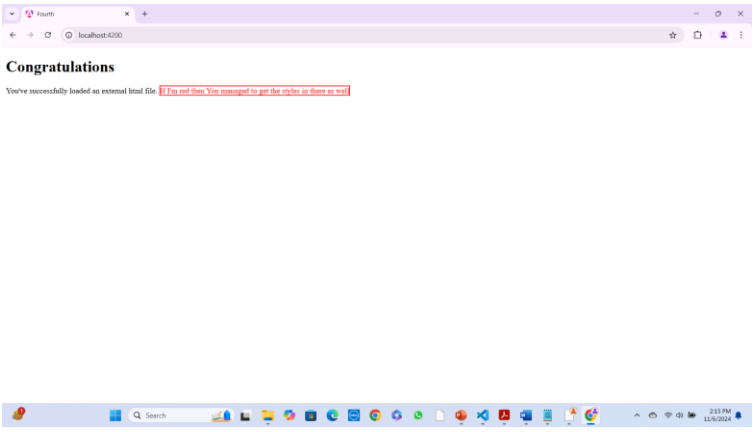
Fourth/src/app/app.component.html

```
<h1>Congratulations</h1>
<p>
 You've successfully loaded an external html file.
 <span>
 If I'm red then You managed to get the styles in there as well
 </span>
 </p>
```

Fourth/src/app/app.component.css

```
span{
    color: red;
    border: 2px solid red;
    }
```

<u>Output:</u>



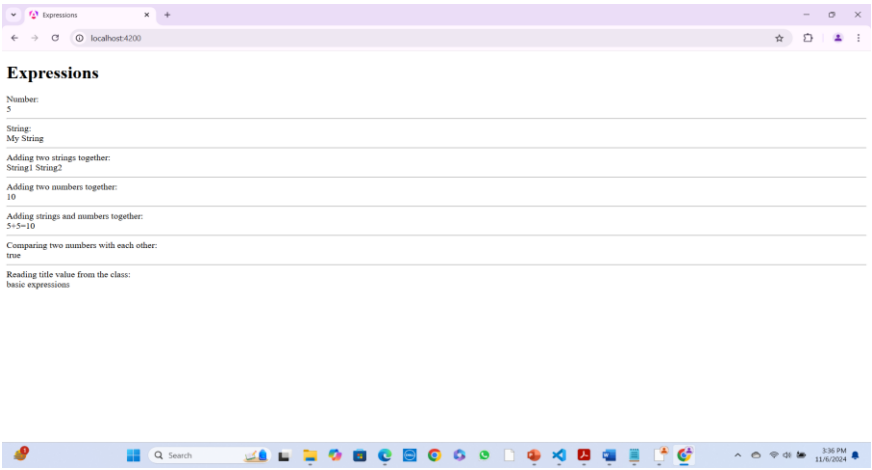
Exp.21

AIM: To demonstrate Using expressions in Angular Applications

Expressions/src/app/app.component.ts

```
//To demonstrate Using expressions in Angular Applications
import { Component } from '@angular/core';
import { RouterOutlet } from '@angular/router';

@Component({
  selector: 'app-root',
  standalone:true,
  template: `
  <h1>Expressions</h1>
  Number:<br>
  {{5}}<hr>
  String:<br>
  {{'My String'}}<hr>
  Adding two strings together:<br>
  {{'String1' + ' ' + 'String2'}}<hr>
  Adding two numbers together:<br>
  {{5+5}}<hr>
  Adding strings and numbers together:<br>
  {{5 + '+' + 5 + '='}}{{5+5}}<hr>
  Comparing two numbers with each other:<br>
  {{5===5}}<hr>
  Reading title value from the class:<br>
  {{title}}
  `,
  })
  export class AppComponent {
title='basic expressions'
}
```

output:



Exp.22

AIM: To demonstrate creating different components like header and footer in Angular.
Note: use the following command to create the application and components
>ng new component1
>component1> ng g c header
>component1> ng g c footer

Component1/src/app/app.component.ts

```
import { Component } from '@angular/core';
import { RouterOutlet } from '@angular/router';
import { HeaderComponent } from './header/header.component';
import { FooterComponent } from './footer/footer.component';
@Component({
  selector: 'app-root',
  standalone: true,
  imports: [RouterOutlet,HeaderComponent,FooterComponent],
  templateUrl: './app.component.html',
  styleUrl: './app.component.css'
})
export class AppComponent {
  title = 'component1';
}
```

Component1/src/app/app.component.html

```
<hr>
<h1> CREATING TWO COMPONENTS </h1>
<hr>
<app-header> </app-header>
<hr>
<app-footer> </app-footer>
<hr>
```

Component1/src/app/header/header.component.html

```
  <h1>Welcome to the HEADER COMPONENT</h1>
```

Component1/src/app/header/header.component.css

```
h1
{
   color: red
}
```

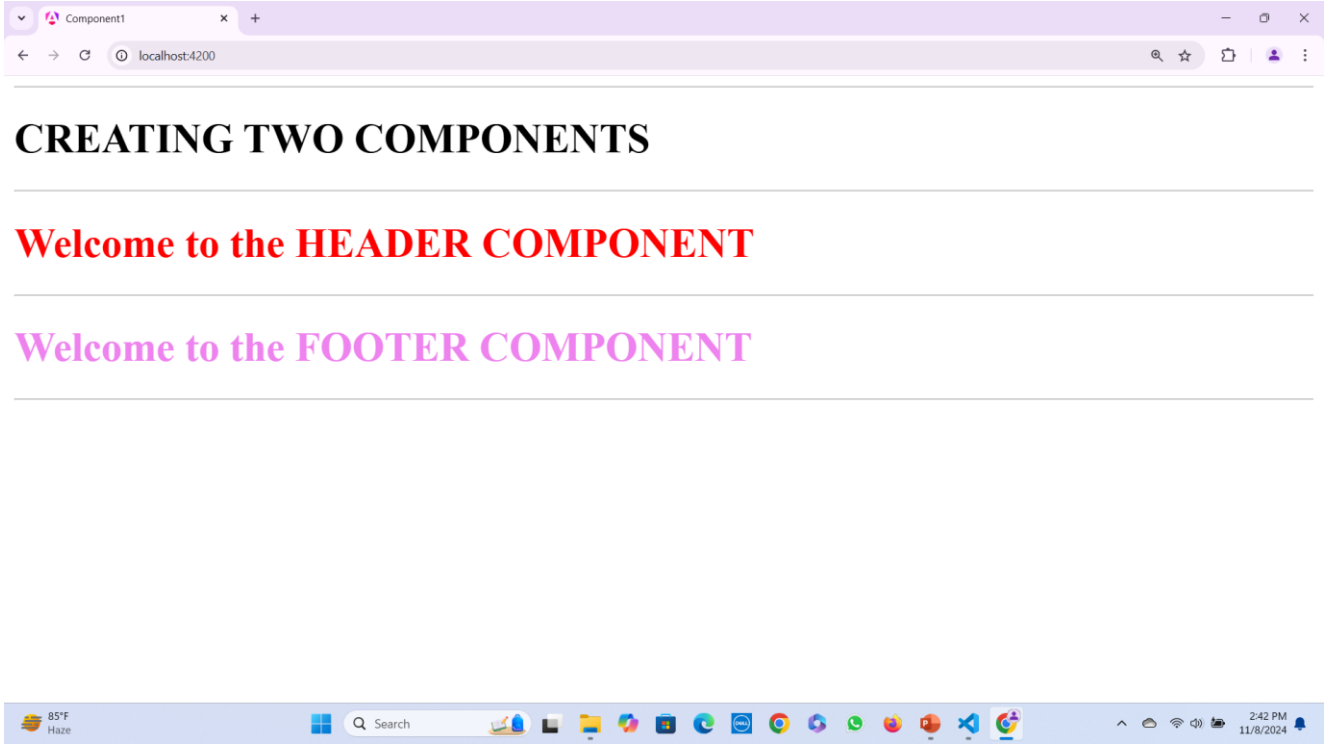
Component1/src/app/footer/footer.component.html

```
<h1>Welcome to the FOOTER COMPONENT</h1>
```

Component1/src/app/footer/footer.component.css

```
h1{
   color: violet
}
```

Output:



Exp.23

AIM: To demonstrate data binding (string interpolation, property binding & class binding) in Angular.

Databinding/src/app/app.component.ts

```
import { Component } from '@angular/core';
import { RouterOutlet } from '@angular/router';
@Component({
  selector: 'app-root',
  standalone: true,
  imports: [RouterOutlet],
  templateUrl: './app.component.html',
  styleUrl: './app.component.css’
})
export class AppComponent {
  title = 'databinding';
  isdisabled:boolean=true;
  isactive:boolean=true;

}
```

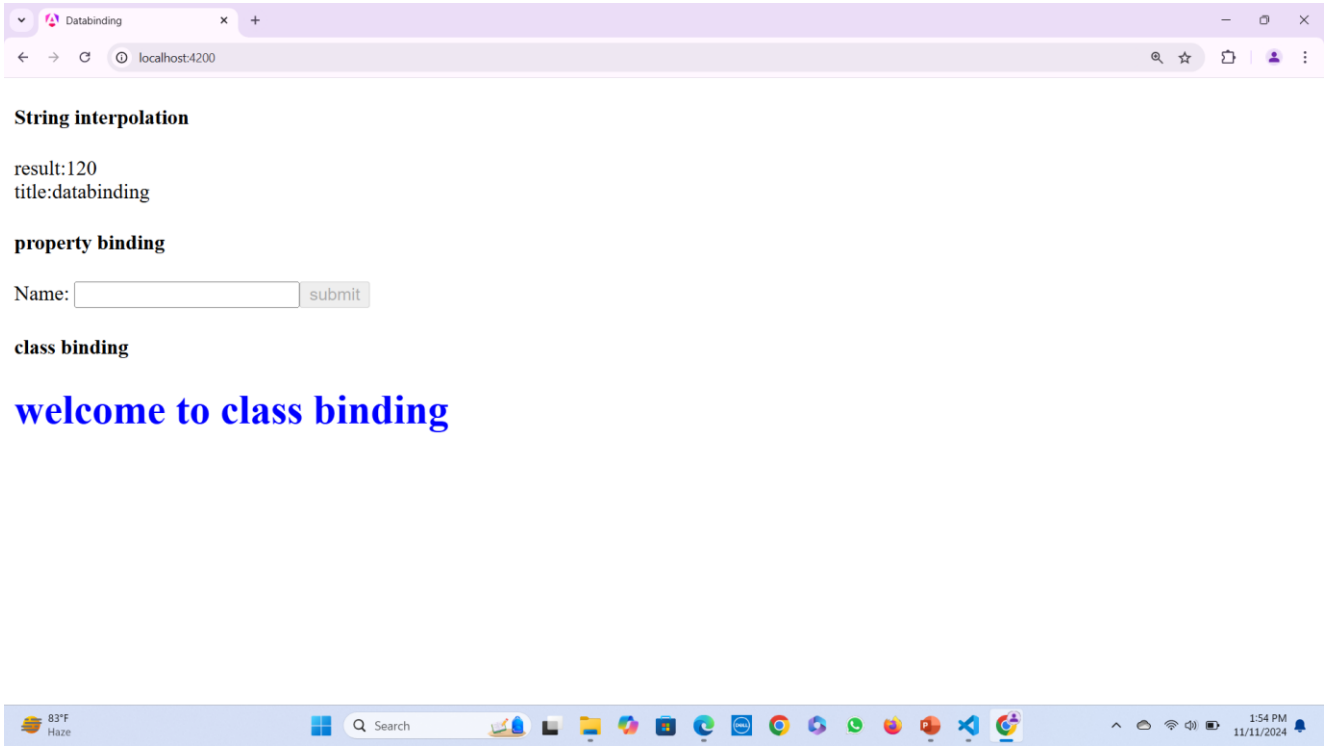
Databinding/src/app/app.component.html

```
<h4> String interpolation</h4>
result:{{100+20}} <br>
title:{{title}}

<h4> property binding</h4>
Name: <input type='text'>
<button [disabled]=isdisabled> submit</button><br>

<h4> class binding</h4>
<h1 [class]="isactive?'active':'inactive'"> welcome to class  binding</h1>
```

Databinding/src/app/app.component.css

```
.active{
    color:blue
}
.inactive{
    color:green
}
```

Output:


Databinding
localhost:4200
String interpolation
result:120
title:databinding
property binding
Name:
submit
class binding
welcome to class binding
83°F
Haze
Search
1:54 PM
11/11/2024

Exp.24

AIM: To demonstrate data binding using events (event binding) in Angular.

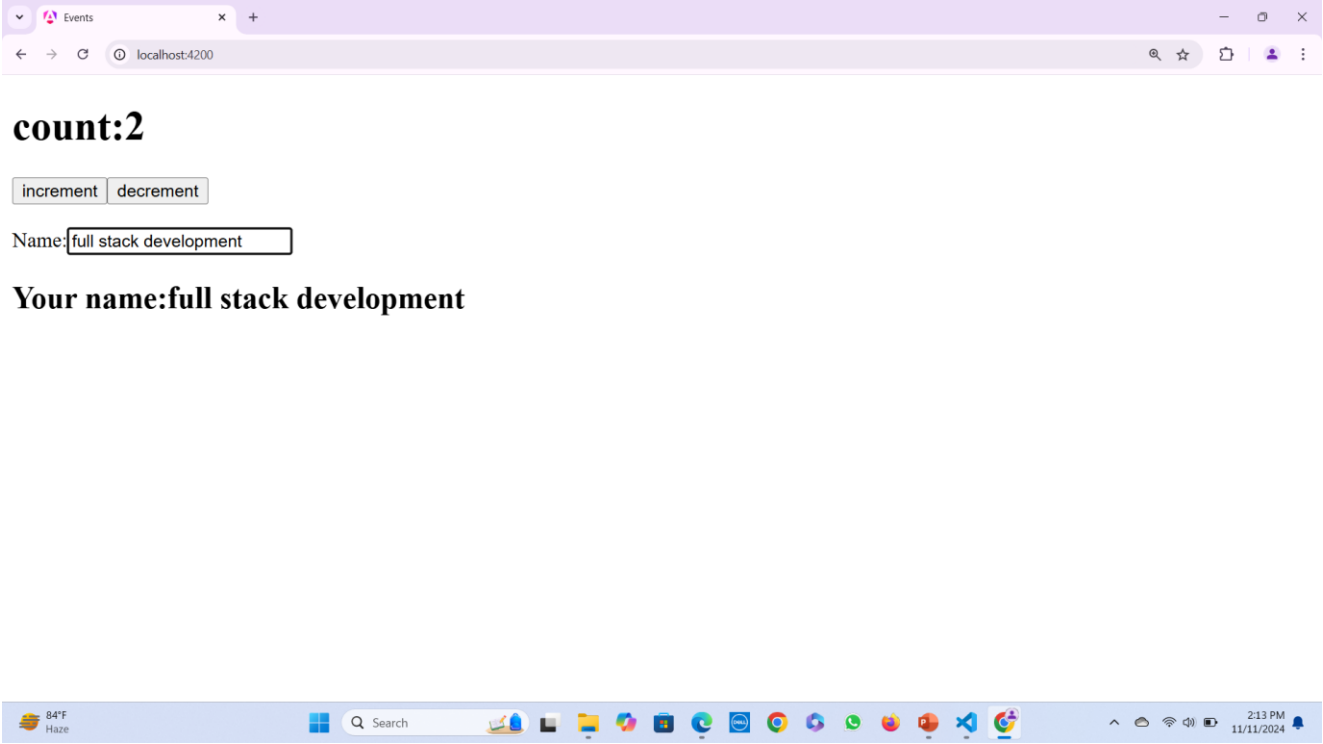
Events/src/app/app.component.ts

```
import { Component } from '@angular/core';
import { RouterOutlet } from '@angular/router';

@Component({
  selector: 'app-root',
  standalone: true,
  imports: [RouterOutlet],
  templateUrl: './app.component.html',
  styleUrl: './app.component.css'
})
export class AppComponent {
  title = 'events';
  counter:number=0;
  name:any="null"
  increment()
  {
    this.counter+=1;
  }
  decrement()
  {
    this.counter-=1;
  }

changeName(e:any)
{
  this.name=e.target.value
}
}
```

Events/src/app/app.component.html

```
<h1> count:{{counter}}</h1>
<button (click)="increment()"> increment</button>
<button (click)="decrement()"> decrement</button>
<br>
<br>

Name:<input type="text"(input)="changeName($event)">
<h2> Your name:{{name}}</h2>
```

Output:


Events
localhost:4200
count:2
increment
decrement
Name:
full stack development
Your name:full stack development
84°F
Haze
Search
2:13 PM
11/11/2024

Exp.25

AIM: To demonstrate two way data binding in Angular.

Note: use the following command to create the application
>ng new twowaybinding --no-standalone

Src/app/app.module.ts

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { FormsModule } from '@angular/forms';
@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    FormsModule
      ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

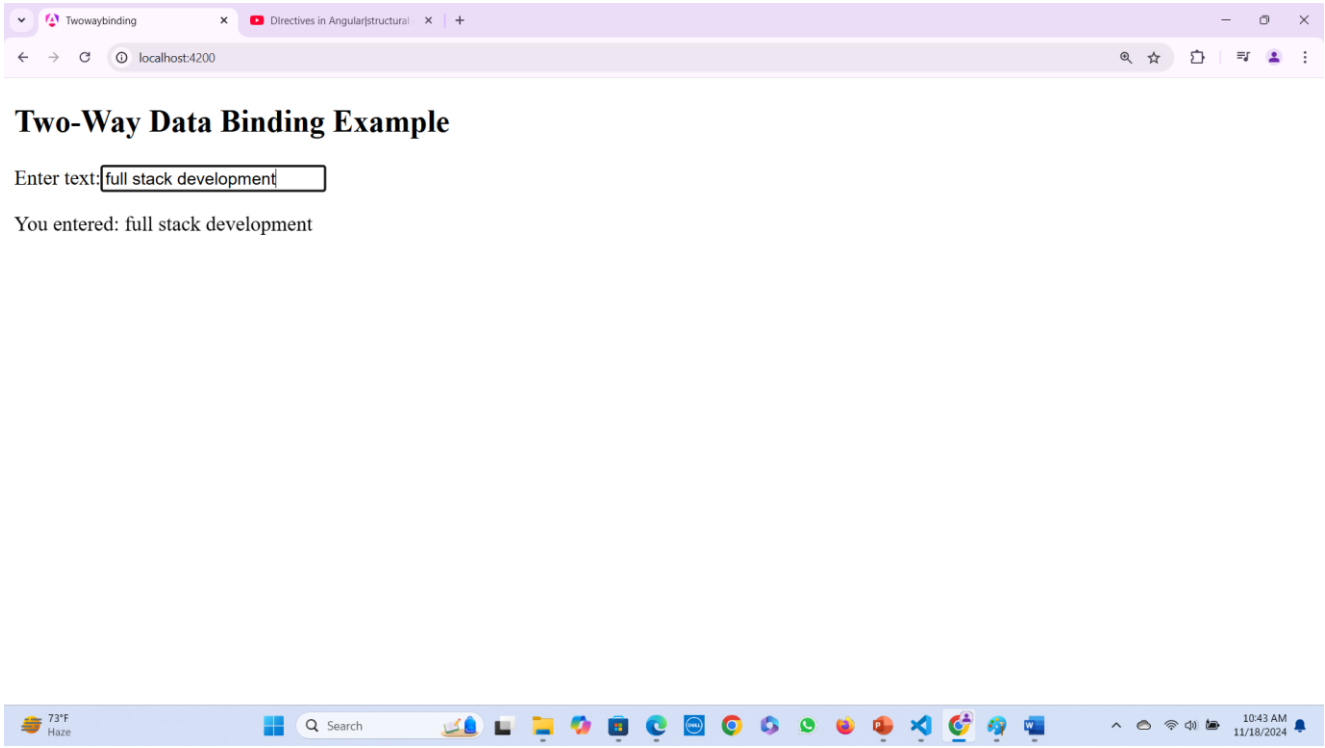
src/app/app.component.ts

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrl: './app.component.css’
})
export class AppComponent {
  title = 'twowaybinding';
  city="hyderabad"

}
```

src/app/app.component.html

```
<div>
  <h2>Two-Way Data Binding Example</h2>

  <!-- Two-way binding using ngModel -->
  <label >Enter text:</label>
  <input [(ngModel)]="text"  />

  <p>You entered: {{ text }}</p>  <!-- Display the text entered by the user -->
</div>
```

Output:



Exp.26

AIM: To demonstrate directives in Angular.

Note: use the following command to create the application
>ng new directives --no-standalone

Src/app/app.module.ts

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';

import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { FormsModule } from '@angular/forms';

@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    FormsModule

  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

src/app/app.component.ts

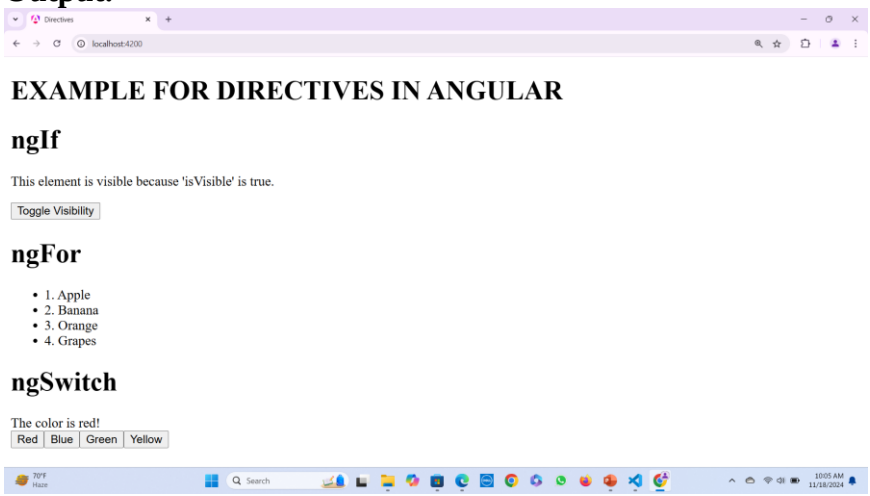
```
import { Component } from '@angular/core';
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html'
})
export class AppComponent {
  isVisible = true;  // Boolean variable to control visibility

  toggleVisibility() {
    this.isVisible = !this.isVisible;  // Toggle the visibility of the element

  }

  items = ['Apple', 'Banana', 'Orange', 'Grapes'];  // Array to iterate over

color: string = 'red';  // Default color

  changeColor(newColor: string) {
```

```
    this.color = newColor;  // Change the color based on button click

  }
}
```

src/app/app.component.html

```
<h1> EXAMPLE FOR DIRECTIVES IN ANGULAR</h1>
<h1> ngIf </h1>
<div *ngIf="isVisible">
  <p>This element is visible because 'isVisible' is true.</p>
</div>

<button (click)="toggleVisibility()">Toggle Visibility</button>

<h1> ngFor </h1>
<ul>
  <li *ngFor="let item of items; let i = index">
    {{ i + 1 }}. {{ item }}

      </li>
</ul>

<h1> ngSwitch </h1>
<div [ngSwitch]="color">
  <div *ngSwitchCase="'red'">The color is red!</div>
  <div *ngSwitchCase="'blue'">The color is blue!</div>
  <div *ngSwitchCase="'green'">The color is green!</div>
  <div *ngSwitchDefault>The color is unknown!</div>
</div>

<button (click)="changeColor('red')">Red</button>
<button (click)="changeColor('blue')">Blue</button>
<button (click)="changeColor('green')">Green</button>
<button (click)="changeColor('yellow')">Yellow</button>
```

Output:



Exp.27

AIM: To create a simple "Hello World" React application.

Prerequisites

Make sure you have the following installed:

1. **Node.js** and **npm** (Node Package Manager). You can check if these are installed by running the following commands in your terminal:

   ```
   node -v
   npm -v
   ```

If you don't have Node.js and npm installed, you can download them from nodejs.org.

Step-by-Step Guide

1. Create a New React App Using Create React App

Run the following command in your terminal:
A)npm install –g create-react-app
B)Create-react-app hello-world
(or)
A) npx create-react-app hello-world

This will create a new directory called `hello-world` and set up a boilerplate React application.

2. Navigate to Your Project Directory

Once the setup is complete, go into the `hello-world` directory:

```
cd hello-world
```

3. Open `src/App.js`

In this file, you will modify the default content to display "Hello World".

The `App.js` file should look something like this:

```
import React from 'react';
import './App.css';

function App() {
  return (
    <div className="App">
      <h1>Hello World</h1>
    </div>
  );
}
```

```
export default App;
```

4. Start the Development Server

Now that your `App.js` is set up, start the development server to view your application.

Run:

```
npm start
```

This will start the server, and you should see your "Hello World" application in your browser at `http://localhost:3000/`.

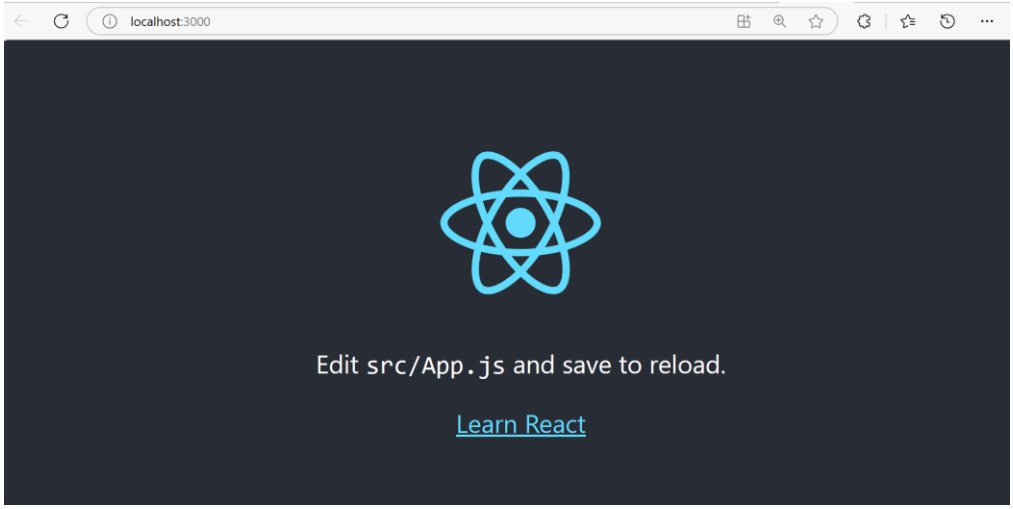
Full Project Structure

After running the above commands, your project structure should look something like this:

```
java
Copy code
hello-world/
├── node_modules/
├── public/
│   └── index.html
├── src/
│   ├── App.css
│   ├── App.js
│   ├── index.js
│   └── logo.svg
├── package.json
├── package-lock.json
└── README.md
```

- The `src/App.js` file contains the main code for your app.
- `index.js` is where React is hooked into the HTML DOM.
- `public/index.html` is the HTML template that React will render the app into.

DEFAULT OUTPUT



AFTER MODIFYING APP.JS

EXP28.

AIM: To demonstrate functional and class components in React.

NOTE:
1)Create two files in the folder '**src**' with CBC.js and FBC.js
2) use rcc and rfc to get the class and functional based syntax in the files.

Src/CBC.js

```
import React, { Component } from 'react'

export default class CBC extends Component {
  render() {
    return (
      <div>class based component </div>
    )
  }
}
```

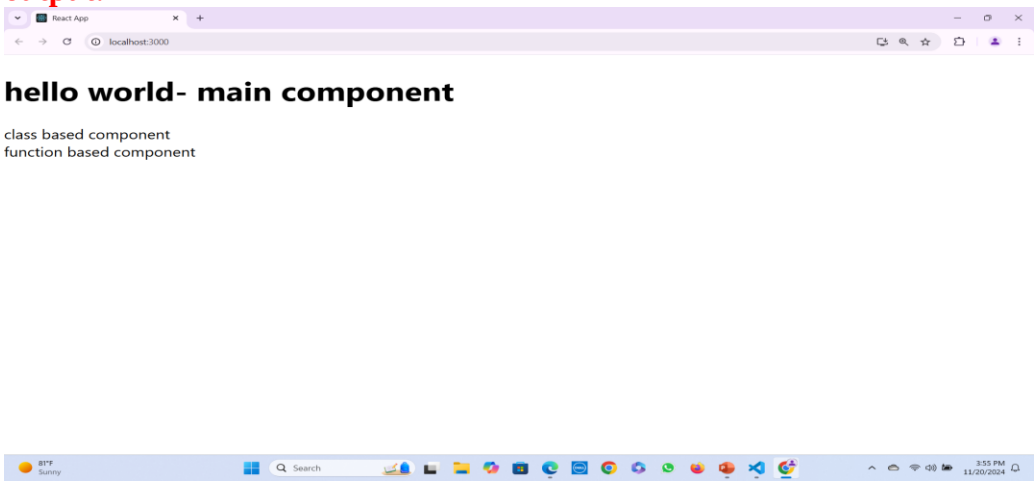
Src/FBC.js

```
import React from 'react'

export default function FBC() {
  return (
    <div> function based component</div>
  )
}
```

Src/App.js

```
import CBC from "./CBC";
import FBC from "./FBC";

function App() {
  return (
    <div>
      <h1>hello world- main component</h1>
      <CBC></CBC>
      <FBC></FBC>
    </div>

  );
}

export default App;
```

output:



EXP29.

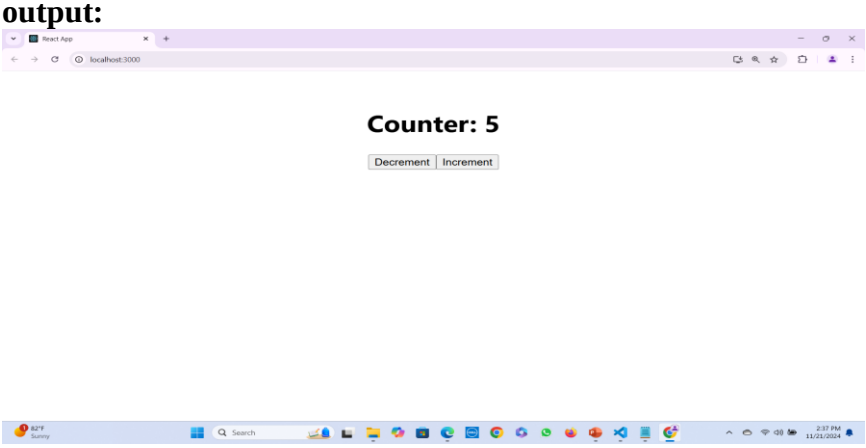
AIM: To demonstrate data (state) and data flow (props) in React.

a) **To demonstrate state in React.**

React-state/src/App.js

```
import React, { useState } from 'react';

function Counter() {
  // Declare a state variable 'count' and a function to update it 'setCount'
  const [count, setCount] = useState(0);

  // Function to handle increment
  const increment = () => {
    setCount(count + 1);

  };

  // Function to handle decrement
  const decrement = () => {
    setCount(count - 1);
  };

  return (
    <div style={{ textAlign: 'center', marginTop: '50px' }}>
      <h1>Counter: {count}</h1>

      <button onClick={decrement}>Decrement</button>
      <button onClick={increment}>Increment</button>
    </div>
  );
}

export default Counter;
```

output:



b) to demonstrate props in React

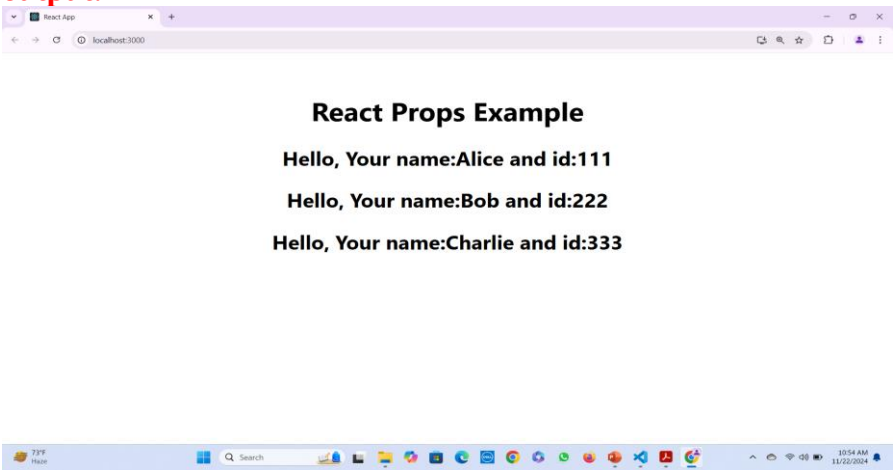
React-props/src/App.js

```
import React from 'react';
import Greeting from './Greeting';
function App() {
  return (
    <div style={{ textAlign: 'center', marginTop: '50px' }}>
      <h1>React Props Example</h1>
      {/* Passing props to the Greeting component */}
      <Greeting name="Alice" id="111" />
      <Greeting name="Bob" id="222"/>
      <Greeting name="Charlie" id="333"/>

    </div>
  );
}
export default App;
```

React-props/src/Greeting.js

```
import React from 'react';

function Greeting(props) {
  return(
      <h2>Hello, Your name:{props.name} and id:{props.id}  </h2>
  );


}

export default Greeting;
```

output:



EXP30.

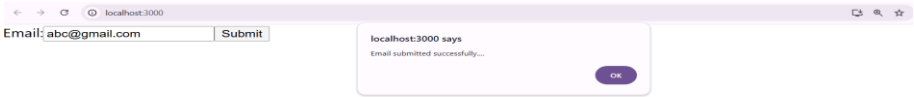
AIM: Develop a form with a field an email address and validate it using React.

Form-validate/Src/App.js

```
//email validation
import React, { useState } from 'react';
export default function ValidatedForm() {
  const [email, setEmail] = useState('');
  const [error, setError] = useState('');

  const handleChange = (e) => {
    const value = e.target.value;
    setEmail(value);

    // Simple validation
    if (!/\S+@\S+\.\S+/.test(value)) {
      setError('Invalid email address');
    } else {
      setError('');
    }
  };

  const handleSubmit = (e) => {
    e.preventDefault();
    if (!error) {
      alert('Email submitted successfully....');
    }
  };

  return (
    <form onSubmit={handleSubmit}>
      <label>
        Email:
        <input type="email" value={email} onChange={handleChange} />
      </label>
      {error && <p style={{ color: 'red' }}>{error}</p>}
```

```
      <button type="submit" disabled={!!error}>
        Submit
      </button>
    </form>
  );
}
```

Output:



Ex.no: 31

Aim: Reading form data consisting of email and password in react

Form-reading/src/App.js

```
import React, { useState } from "react";

function App() {
  const [formData, setFormData] = useState({
    username: "",
    password: "",
  });

  const handleChange = (event) => {
    const { name, value } = event.target;
    setFormData((prevState) => ({ ...prevState, [name]: value }));
  };

  const handleSubmit = (event) => {
    event.preventDefault();
    console.log(formData);
  };

  return (
    <form onSubmit={handleSubmit}>
      <label>
        Username:
        <input
          type="text"
          name="username"
          value={formData.username}
          onChange={handleChange}
        />
      </label> <br/>
      <label>
        Password:
        <input
          type="password"
          name="password"
          value={formData.password}
```

```
        onChange={handleChange}
      />
    </label> <br/>
    <button onClick={()=>
      console.log("email and password submitted...") }>
      Submit
    </button>
  </form>
 );

}
export default App;
```

output:

EXP32.
AIM: Develop a form with fields for a phone number and an email address, and validate them using React.

Form-validation/src/App.js

```
import React, { useState } from 'react';

function IndianPhoneAndEmailForm() {
  const [formData, setFormData] = useState({
    email: '',
    phone: '',
  });

  const [errors, setErrors] = useState({
    email: '',
    phone: '',
  });

  const emailRegex = /^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$/;
  const indianPhoneRegex = /^[6-9]\d{9}$/;

  const handleChange = (e) => {
    const { name, value } = e.target;

    // Update form data
    setFormData((prev) => ({ ...prev, [name]: value }));
    // Validation logic
    if (name === 'email') {
      if (!emailRegex.test(value)) {
        setErrors((prev) => ({ ...prev, email: 'Invalid email format' }));
      } else {
        setErrors((prev) => ({ ...prev, email: '' }));
      }
    }
```

```
    if (name === 'phone') {
      if (!indianPhoneRegex.test(value)) {
        setErrors((prev) => ({
          ...prev,
          phone: 'Phone number must start with 6, 7, 8, or 9 and be 10 digits
long',
        }));
      } else {
        setErrors((prev) => ({ ...prev, phone: '' }));
      }
    }
  };


  const handleSubmit = (e) => {
    e.preventDefault();

    // Check for empty fields or errors
    if (!formData.email || !formData.phone || errors.email || errors.phone) {
      alert('Please correct the errors before submitting.');
      return;
    }

    // Submit data
    console.log('Form Data Submitted:', formData);
    alert('Form submitted successfully!');
  };

  return (
    <div>
      <h2>Indian Phone and Email Validation Form</h2>
      <form onSubmit={handleSubmit}>
        <div>
          <label>Email:</label>
          <input
            type="email"
            name="email"
            value={formData.email}
            onChange={handleChange}
          />
          {errors.email && <p style={{ color: 'red' }}>{errors.email}</p>}
        </div>
        <div>
          <label>Phone:</label>
          <input
            type="text"
            name="phone"
```

```
            value={formData.phone}
            onChange={handleChange}
          />
          {errors.phone && <p style={{ color: 'red' }}>{errors.phone}</p>}
        </div>
        <button type="submit" disabled={!!errors.email || !!errors.phone}>
          Submit
        </button>
      </form>
    </div>
  );
}
export default IndianPhoneAndEmailForm;
```

OUTPUT:

Sample1

Indian Phone and Email Validation Form

Email: ABC@GMAIL

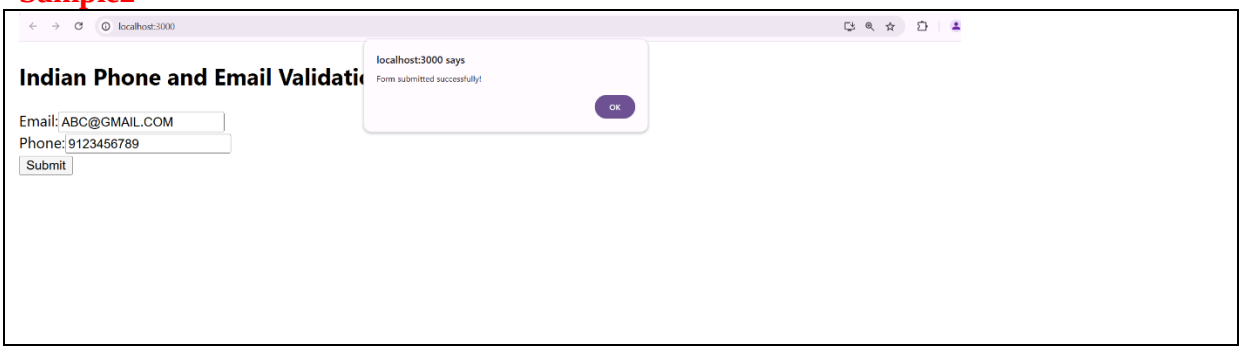
Invalid email format

Phone: 99123456

Phone number must start with 6, 7, 8, or 9 and be 10 digits long

Submit

Sample2



Ex.no: 33

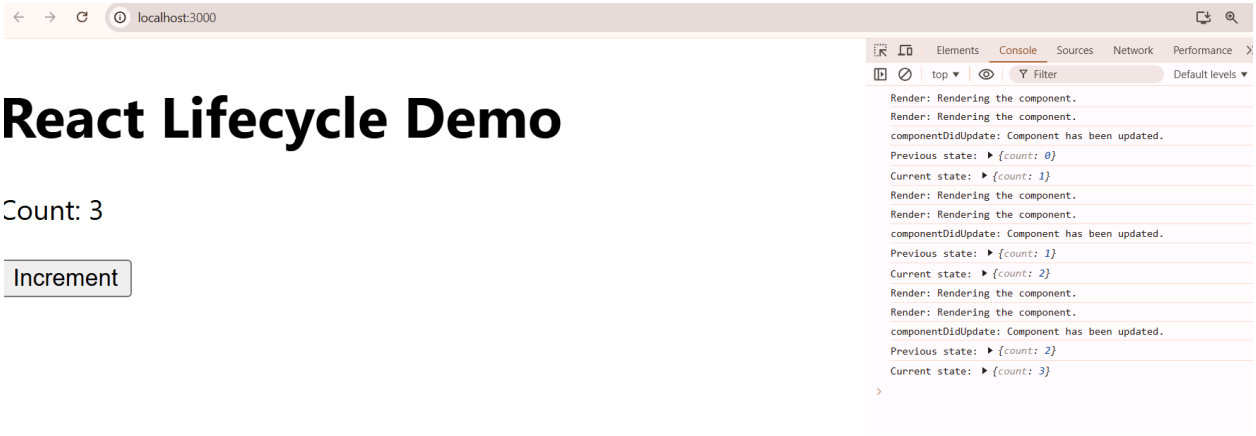
Aim: to demonstrate life cycle methods in react.

Src/App.js

```
import React from 'react';
class LifeCycleDemo extends React.Component {
  // Constructor: Initialize state or bind methods
  constructor(props) {
    super(props);
    this.state = {
      count: 0,
    };
    console.log('Constructor: Component is being created.');
  }

  // Called after the component is added to the DOM
  componentDidMount() {
    console.log('componentDidMount: Component has been added to the DOM.');
  }

  // Called when the component is updated (e.g., state or props change)
  componentDidUpdate(prevProps, prevState) {
    console.log('componentDidUpdate: Component has been updated.');
    console.log('Previous state:', prevState);
    console.log('Current state:', this.state);
  }

  // Called just before the component is removed from the DOM
  componentWillUnmount() {
    console.log('componentWillUnmount: Component is being removed.');
  }

  // Event handler to update state
  handleIncrement = () => {
    this.setState((prevState) => ({ count: prevState.count + 1 }));
```

```
  };
  render() {
    console.log('Render: Rendering the component.');
    return (
      <div>
        <h1>React Lifecycle Demo</h1>
        <p>Count: {this.state.count}</p>
        <button onClick={this.handleIncrement}>Increment</button>
      </div>
    );
  }
}export default LifeCycleDemo;
```

output:



Ex.no: 34
Aim: to demonstrate routing in react.

Src/app.js

```
import React from 'react';
import { BrowserRouter as Router, Routes, Route, Link } from 'react-router-dom';
import Home from './Home';
import About from './About';
import Contact from './Contact';

const App = () => {
  return (
    <Router>
      <div>
        <h1>React Router Demo</h1>
        {/* Navigation Bar */}
        <nav>
          <ul>
            <li><Link to="/">Home</Link></li>
            <li><Link to="/about">About</Link></li>
            <li><Link to="/contact">Contact</Link></li>
          </ul>
        </nav>
 {/* Routes for Pages */}
 <Routes>
          <Route path="/" element={<Home />} />
          <Route path="/about" element={<About />} />
          <Route path="/contact" element={<Contact />} />
  </Routes>
  </div>
    </Router>
  );
};

export default App;
```

Src/Home.js

```
import React from 'react';

const Home = () => {
  return (
    <div>
      <h2>Home Page</h2>
      <p>Welcome to the home page!</p>
    </div>
  );
};

export default Home;
```

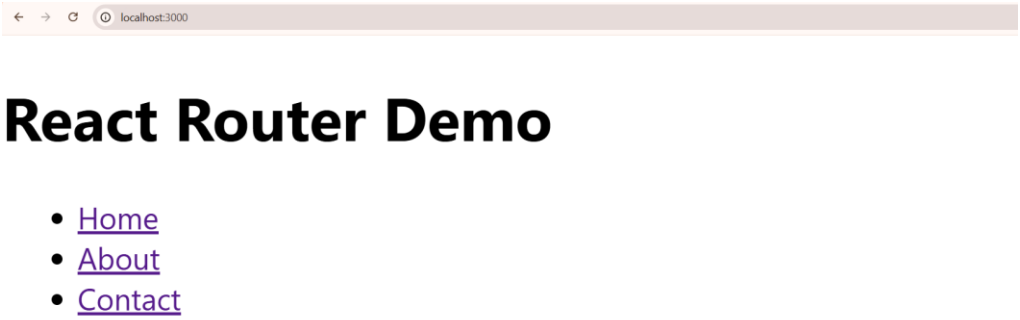
Src/About.js

```
import React from 'react';

const About = () => {
  return (
    <div>
      <h2>About Page</h2>
      <p>This page provides information about us.</p>
    </div>
  );
};

export default About;
```

Src/Contact.js

```
import React from 'react';

const Contact = () => {
  return (
    <div>
      <h2>Contact Page</h2>
      <p>Feel free to contact us at contact@example.com.</p>
    </div>
  );
};

export default Contact;
```

output :



Ex.no: 35
Aim: Write a program to create a simple calculator Application using React JS.

Src/App.js

```
import React, { useState } from "react";
import "./App.css";

function App() {
  const [input, setInput] = useState("");

  const handleButtonClick = (value) => {
    if (value === "=") {
      try {
        setInput(eval(input).toString()); // Caution: Avoid `eval` in production apps.
      } catch (error) {
        setInput("Error");
      }
    } else if (value === "C") {
      setInput("");
    } else {
      setInput(input + value);
    }
  };

  return (
    <div className="App">
      <h1>React Calculator</h1>
      <div className="calculator">
        <div className="display">{input || "0"}</div>
        <div className="buttons">
          {["7", "8", "9", "/", "4", "5", "6", "*", "1", "2", "3", "-", "0", ".", "C", "+"].map((btn) => (
            <button key={btn} onClick={() => handleButtonClick(btn)}>
              {btn}
            </button>
          ))}
          <button className="equals" onClick={() => handleButtonClick("=")}>
```

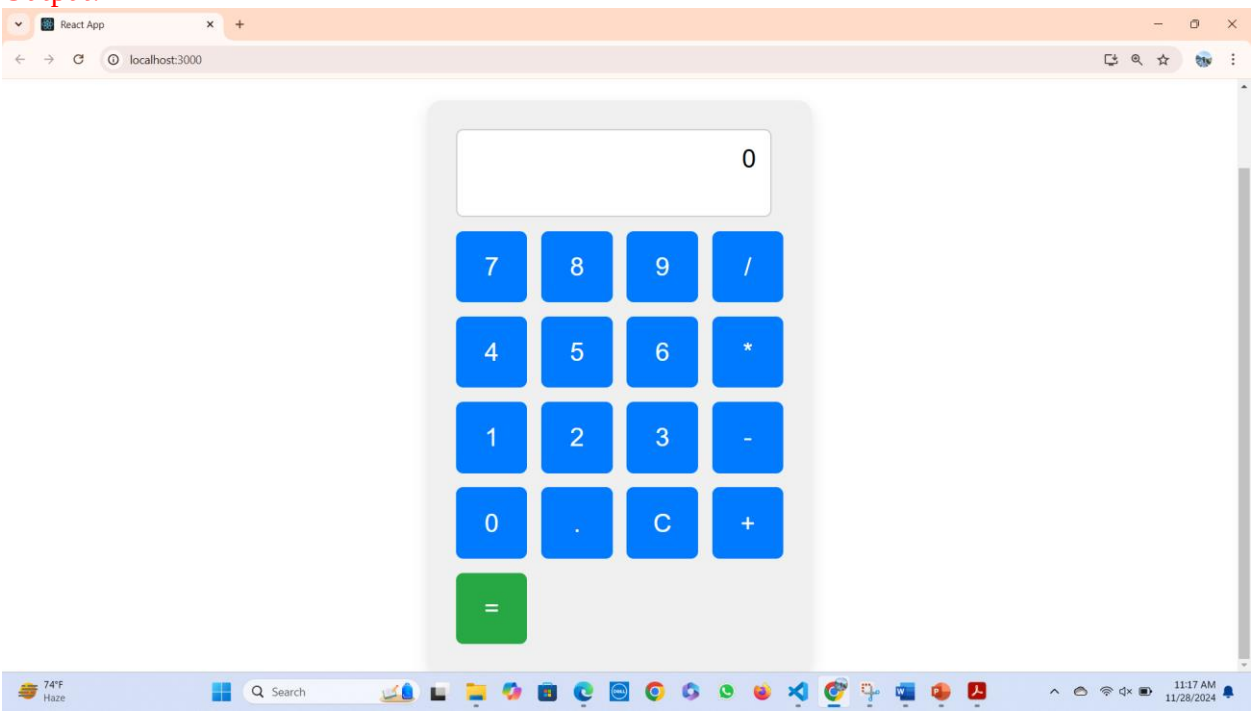
```
          =
        </button>
      </div>
    </div>
  </div>
 );
}

export default App;
```

Src/App.css

```
.App {
  text-align: center;
  font-family: Arial, sans-serif;
}

.calculator {
  display: inline-block;
  background: #f0f0f0;
  padding: 20px;
  border-radius: 10px;
  box-shadow: 0 4px 10px rgba(0, 0, 0, 0.1);
}

.display {
  width: 200px;
  height: 40px;
  margin-bottom: 10px;
  background: #fff;
  text-align: right;
  padding: 10px;
  border: 1px solid #ccc;
  border-radius: 5px;
  font-size: 18px;
  overflow-x: auto;
}

.buttons {
  display: grid;
  grid-template-columns: repeat(4, 50px);
  gap: 10px;
}

button {
  width: 50px;
```

```
  height: 50px;
  font-size: 18px;
  background: #007bff;
  color: white;
  border: none;
  border-radius: 5px;
  cursor: pointer;
  transition: background 0.3s;
}

button:hover {
  background: #0056b3;
}

.equals {
  grid-column: span 4;
  background: #28a745;
}

.equals:hover {
  background: #1e7e34;
}
```

Output:



Ex.no: 36

Aim: Write a program to create a voting application using React JS

Src/App.js

```
import React, { useState } from "react";
import "./App.css";

function App() {
  const [candidates, setCandidates] = useState([
    { name: "CSE1", votes: 0 },
    { name: "CSE2", votes: 0 },
    { name: "CSE3", votes: 0 },
  ]);

  const handleVote = (index) => {
    const updatedCandidates = [...candidates];
    updatedCandidates[index].votes += 1;
    setCandidates(updatedCandidates);
  };

  const resetVotes = () => {
    const resetCandidates = candidates.map((candidate) => ({
      ...candidate,
      votes: 0,
    }));
    setCandidates(resetCandidates);
  };

  return (
    <div className="App">
      <h1>Voting Application</h1>
      <div className="candidates">
        {candidates.map((candidate, index) => (
          <div key={index} className="candidate">
            <span>{candidate.name}</span>
```

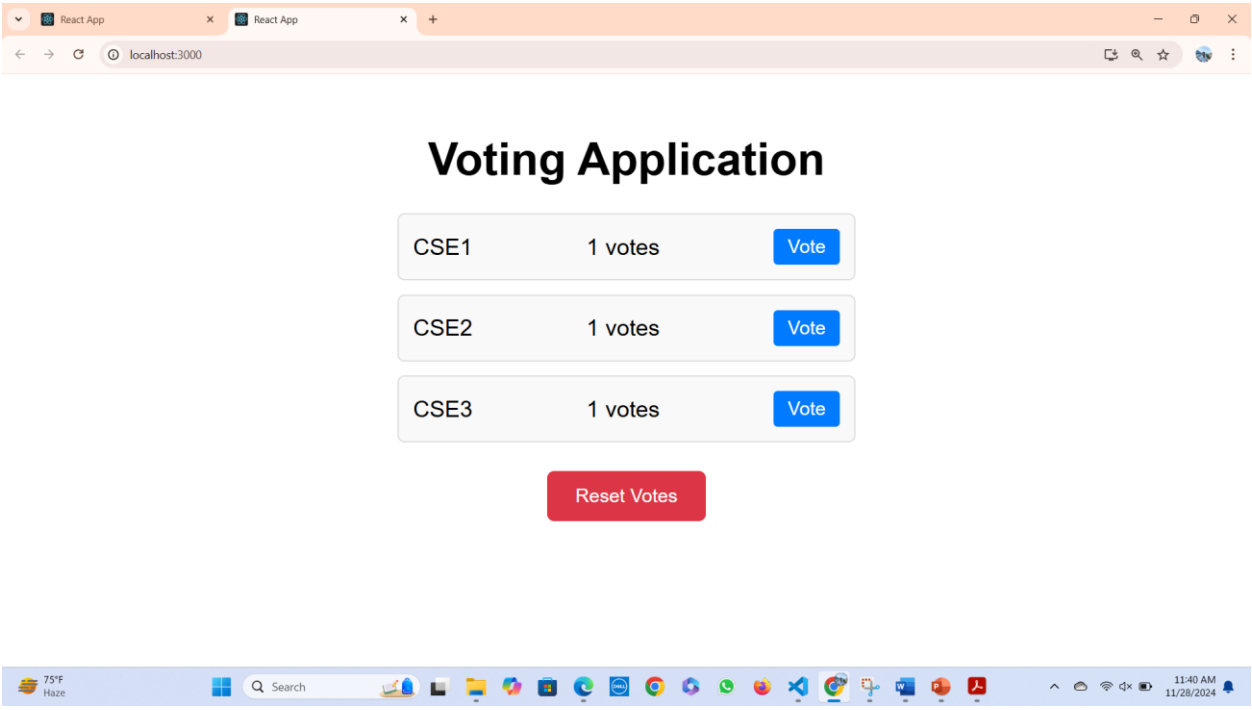
```
            <span>{candidate.votes} votes</span>
            <button onClick={() => handleVote(index)}>Vote</button>
          </div>
        ))}
      </div>
      <button className="reset" onClick={resetVotes}>
        Reset Votes
      </button>
    </div>
  );
}
export default App;
```

Src/App.css

```
.App {
  text-align: center;
  font-family: Arial, sans-serif;
  padding: 20px;
}

h1 {
  margin-bottom: 20px;
  font-size: 2em;
}

.candidates {
  display: flex;
  flex-direction: column;
  gap: 10px;
  align-items: center;
}

.candidate {
  display: flex;
  justify-content: space-between;
  align-items: center;
  width: 300px;
  padding: 10px;
  background: #f9f9f9;
  border: 1px solid #ddd;
  border-radius: 5px;
}

.candidate button {
  padding: 5px 10px;
  background: #007bff;
  color: #fff;
  border: none;
  border-radius: 3px;
  cursor: pointer;
  transition: background 0.3s;
```

```
}

.candidate button:hover {
  background: #0056b3;
}

.reset {
  margin-top: 20px;
  padding: 10px 20px;
  background: #dc3545;
  color: white;
  border: none;
  border-radius: 5px;
  cursor: pointer;
  transition: background 0.3s;
}

.reset:hover {
  background: #c82333;
}
```

Output:



Ex.no: 37

Aim: Develop a leave management system for an organization where users can apply different types of leaves such as casual leave and medical leave. They also can view the available number of days using react application.

Src/App.js

```
import React, { useState } from "react";
import "./App.css";

function App() {
  const initialLeaveBalance = {
    CasualLeave: 12,
    MedicalLeave: 10,
    EarnedLeave: 8,
  };

  const [leaveBalance, setLeaveBalance] = useState(initialLeaveBalance);
  const [leaveHistory, setLeaveHistory] = useState([]);
  const [selectedLeaveType, setSelectedLeaveType] = useState("CasualLeave");
  const [leaveDays, setLeaveDays] = useState(1);

  const handleApplyLeave = () => {
    if (leaveDays < 1 || isNaN(leaveDays)) {
      alert("Please enter a valid number of days.");
      return;
    }

    if (leaveBalance[selectedLeaveType] >= leaveDays) {
      // Update leave balance
      setLeaveBalance({
        ...leaveBalance,
        [selectedLeaveType]: leaveBalance[selectedLeaveType] - leaveDays,
      });
```

```
    // Add to leave history
    setLeaveHistory([
      ...leaveHistory,
      {
        type: selectedLeaveType,
        days: leaveDays,
        date: new Date().toLocaleDateString(),
      },
    ]);

    alert("Leave applied successfully!");
  } else {
    alert("Not enough leave balance!");
  }
};

const handleResetLeaves = () => {
  setLeaveBalance(initialLeaveBalance);
  setLeaveHistory([]);
};

return (
  <div className="App">
    <h1>Leave Management System</h1>

    <div className="leave-balance">
      <h2>Leave Balance</h2>
      <ul>
        {Object.entries(leaveBalance).map(([type, days]) => (
          <li key={type}>
            {type}: {days} days
          </li>
        ))}
      </ul>
    </div>

    <div className="apply-leave">
      <h2>Apply Leave</h2>
      <label>
        Leave Type:
        <select
          value={selectedLeaveType}
          onChange={(e) => setSelectedLeaveType(e.target.value)}
        >
          {Object.keys(leaveBalance).map((type) => (
            <option key={type} value={type}>
              {type}
            </option>
          ))}
        </select>
```

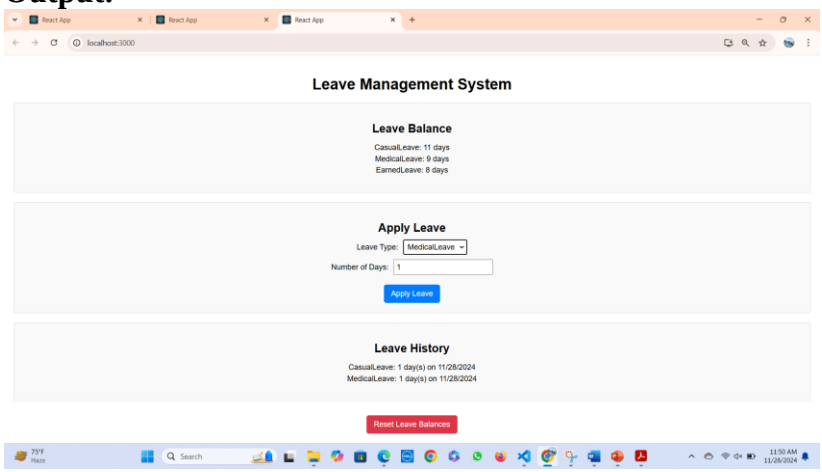
```
      </label>

      <label>
        Number of Days:
        <input
          type="number"
          min="1"
          value={leaveDays}
          onChange={(e) => setLeaveDays(parseInt(e.target.value))}
        />
      </label>

      <button onClick={handleApplyLeave}>Apply Leave</button>
    </div>

    <div className="leave-history">
      <h2>Leave History</h2>
      {leaveHistory.length > 0 ? (
        <ul>
          {leaveHistory.map((leave, index) => (
            <li key={index}>
              {leave.type}: {leave.days} day(s) on {leave.date}
            </li>
          ))}
        </ul>
      ) : (
        <p>No leave history available.</p>
      )}
    </div>

    <button className="reset-button" onClick={handleResetLeaves}>
      Reset Leave Balances
    </button>
  </div>
 );
}

export default App;
```

Src/App.css

```
.App {
  text-align: center;
  font-family: Arial, sans-serif;
  padding: 20px;
}

h1 {
  font-size: 2em;
  margin-bottom: 20px;
```

```
}
.leave-balance,
.apply-leave,
.leave-history {
  margin: 20px 0;
  padding: 20px;
  border: 1px solid #ddd;
  border-radius: 5px;
  background-color: #f9f9f9;
}
h2 {
  margin-bottom: 10px;
}
ul {
  list-style: none;
  padding: 0;
}
li {
  margin: 5px 0;
}
label {
  display: block;
  margin: 10px 0;
}
input,
select {
  margin-left: 10px;
  padding: 5px;
  font-size: 16px;
}
button {
  margin-top: 10px;
  padding: 10px 15px;
  font-size: 16px;
  background-color: #007bff;
  color: white;
  border: none;
  border-radius: 5px;
  cursor: pointer;
  transition: background 0.3s;
}
button:hover {
  background-color: #0056b3;
}
.reset-button {
  background-color: #dc3545;
}
.reset-button:hover {
  background-color: #c82333;
}
```

Output:



Ex.no: 38

AIM: Build a music store application using react components and provide routing among the web pages

Src/app.js

```
import React from 'react';
import { BrowserRouter as Router, Routes, Route } from 'react-router-dom';
import Header from './Header';
import Footer from './Footer';
import Home from './Home';
import Store from './Store';
import About from './About';
import NotFound from './NotFound';

const App = () => (
  <Router>
    <Header />
    <Routes>
      <Route path="/" element={<Home />} />
      <Route path="/store" element={<Store />} />
      <Route path="/about" element={<About />} />
      <Route path="*" element={<NotFound />} />
    </Routes>
    <Footer />
  </Router>
);

export default App;
```

Src/Header.js

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

```
const Header = () => (
  <header>
    <nav>
      <ul>
        <li><Link to="/">Home</Link></li>
        <li><Link to="/store">Store</Link></li>
        <li><Link to="/about">About</Link></li>
      </ul>
    </nav>
  </header>
);
export default Header;
```

Src/Footer.js

```
import React from 'react';

const Footer = () => (
  <footer>
    <p>&copy; 2024 Music Store</p>
  </footer>
);

export default Footer;
```

Src/Home.js

```
import React from 'react';

const Home = () => (
  <main>
    <h1>Welcome to the Music Store</h1>
    <p>Your one-stop shop for musical instruments and gear!</p>
  </main>
);

export default Home;
```

Src/Store.js

```
import React from 'react';

const Store = () => {
  const products = [
    { id: 1, name: 'Acoustic Guitar', price: '$200' },
    { id: 2, name: 'Electric Guitar', price: '$400' },
    { id: 3, name: 'Drum Set', price: '$600' },
  ];

  return (
    <main>
      <h1>Store</h1>
```

```
      <ul>
        {products.map((product) => (
          <li key={product.id}>
            {product.name} - {product.price}
          </li>
        ))}
      </ul>
    </main>
  );
};

export default Store;
```

Src/About.js

```
import React from 'react';

const About = () => (
  <main>
    <h1>About Us</h1>
    <p>We are passionate about music and dedicated to serving musicians worldwide.</p>
  </main>
);

export default About;
```

Src/NotFound.js

```
import React from 'react';

const NotFound = () => (
  <main>
    <h1>404 - Page Not Found</h1>
    <p>Sorry, the page you are looking for does not exist.</p>
  </main>
);

export default NotFound;
```

src/index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import App from './App';
import './index.css';

ReactDOM.render(
```

```
  <React.StrictMode>
    <App />
  </React.StrictMode>,
  document.getElementById('root')
);
```

src/index.css

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

header nav ul {
  display: flex;
  list-style: none;
  background-color: #333;
  padding: 0;
  margin: 0;
}

header nav ul li {
  margin: 0;
}

header nav ul li a {
  color: white;
  text-decoration: none;
  padding: 10px 20px;
  display: block;
}

header nav ul li a:hover {
  background-color: #575757;
}

footer {
  text-align: center;
  background: #333;
  color: white;
  padding: 10px 0;
  position: fixed;
  bottom: 0;
  width: 100%;
}

main {
  padding: 20px;
}
```

Output:

Ex.no: 38

AIM: Create a react application for an online store which consists of registration, login, product information pages and implement routing to navigate through these pages.

Structure:

```
src/
├── components/
│   ├── Header.js
│   ├── Footer.js
│   ├── Home.js
│   ├── Login.js
│   ├── Register.js
│   ├── Product.js
│   ├── NotFound.js
├── App.js
├── index.js
```

components/ Header.js

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

const Header = () => (
  <header>
    <nav>
      <ul>
        <li><Link to="/">Home</Link></li>
        <li><Link to="/login">Login</Link></li>
        <li><Link to="/register">Register</Link></li>
        <li><Link to="/product">Product</Link></li>
      </ul>
    </nav>
  </header>
);
```

```
export default Header;
```

components/ Footer.js

```
import React from 'react';

const Footer = () => (
  <footer>
    <p>&copy; 2024 Online Store</p>
  </footer>
);
export default Footer;
```

components/ Home.js

```
import React from 'react';

const Home = () => (
  <main>
    <h1>Welcome to the Online Store</h1>
    <p>Shop the best products here!</p>
  </main>
);

export default Home;
```

components/ Login.js

```
import React, { useState } from 'react';

const Login = () => {
  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');

  const handleSubmit = (e) => {
    e.preventDefault();
    alert(`Logged in with email: ${email}`);
  };
  return (
    <main>
      <h1>Login</h1>
      <form onSubmit={handleSubmit}>
        <label>
          Email:
          <input type="email" value={email} onChange={(e) => setEmail(e.target.value)}
required />
        </label>
        <br />
        <label>
          Password:
```

```
        <input type="password" value={password} onChange={(e) =>
setPassword(e.target.value)} required />
      </label>
      <br />
      <button type="submit">Login</button>
    </form>
  </main>
 );
};
export default Login;
```

components/ Register.js

```
import React, { useState } from 'react';
const Register = () => {
 const [formData, setFormData] = useState({
  name: '',
  email: '',
  password: '',
 });
 const handleChange = (e) => {
  setFormData({ ...formData, [e.target.name]: e.target.value });
 };

 const handleSubmit = (e) => {
  e.preventDefault();
  alert(`Registered: ${formData.name} (${formData.email})`);
 };
 return (
  <main>
   <h1>Register</h1>
   <form onSubmit={handleSubmit}>
    <label>
     Name:
     <input name="name" type="text" value={formData.name}
onChange={handleChange} required />
    </label>
    <br />
    <label>
     Email:
     <input name="email" type="email" value={formData.email}
onChange={handleChange} required />
    </label>
    <br />
    <label>
     Password:
     <input name="password" type="password" value={formData.password}
onChange={handleChange} required />
    </label>
    <br />
```

```
        <button type="submit">Register</button>
      </form>
    </main>
  );
};
export default Register;
```

components/ Product.js

```
import React from 'react';

const Product = () => {
  const products = [
    { id: 1, name: 'Laptop', price: '$1000', description: 'High-performance laptop.' },
    { id: 2, name: 'Headphones', price: '$200', description: 'Noise-cancelling headphones.' },
    { id: 3, name: 'Smartphone', price: '$800', description: 'Latest-gen smartphone.' },
  ];

  return (
    <main>
      <h1>Products</h1>
      <ul>
        {products.map((product) => (
          <li key={product.id}>
            <h2>{product.name}</h2>
            <p>{product.description}</p>
            <p>Price: {product.price}</p>
          </li>
        ))}
      </ul>
    </main>
  );
};

export default Product;
```

components/ NotFound.js

```
import React from 'react';

const NotFound = () => (
  <main>
    <h1>404 - Page Not Found</h1>
    <p>Sorry, the page you are looking for does not exist.</p>
  </main>
);
```

```
export default NotFound;
```

App.js

```
import React from 'react';
import { BrowserRouter as Router, Routes, Route } from 'react-router-dom';
import Header from './components/Header';
import Footer from './components/Footer';
import Home from './components/Home';
import Login from './components/Login';
import Register from './components/Register';
import Product from './components/Product';
import NotFound from './components/NotFound';

const App = () => (
  <Router>
    <Header />
    <Routes>
      <Route path="/" element={<Home />} />
      <Route path="/login" element={<Login />} />
      <Route path="/register" element={<Register />} />
      <Route path="/product" element={<Product />} />
      <Route path="*" element={<NotFound />} />
    </Routes>
    <Footer />
  </Router>
);

export default App;
```

Index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import App from './App';
import './index.css';

ReactDOM.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
  document.getElementById('root')
);
```

Output:

Ex.no: 39

AIM:

Create a food delivery website where users can order food from a particular restaurant listed in the website for handling http requests and responses using NodeJS.

Server.js

```
const http = require('http');
const url = require('url');

// Sample menu data
const menu = [
  { id: 1, name: 'Pizza', price: 10 },
  { id: 2, name: 'Burger', price: 5 },
  { id: 3, name: 'Pasta', price: 8 },
];

// Handle HTTP requests
const requestListener = (req, res) => {
  const { method, url: reqUrl } = req;

  // Set content type to JSON
  res.setHeader('Content-Type', 'application/json');

  // Parse the URL to handle routing
  const parsedUrl = url.parse(reqUrl, true);
  const pathname = parsedUrl.pathname;

  if (method === 'GET') {
    if (pathname === '/menu') {
      // Send the menu items
      res.statusCode = 200;
      res.end(JSON.stringify(menu));
```

```
    } else if (pathname.startsWith('/order')) {
      // Extract the order ID from the URL
      const orderId = pathname.split('/')[2];
      const orderItem = menu.find(item => item.id === parseInt(orderId));

      if (orderItem) {
        res.statusCode = 200;
        res.end(JSON.stringify({ message: `Order placed for ${orderItem.name}` }));
      } else {
        res.statusCode = 404;
        res.end(JSON.stringify({ error: 'Item not found' }));
      }
    } else {
      res.statusCode = 404;
      res.end(JSON.stringify({ error: 'Route not found' }));
    }
  } else {
    res.statusCode = 405; // Method Not Allowed
    res.end(JSON.stringify({ error: 'Method not allowed' }));
  }
};

// Create server
const server = http.createServer(requestListener);

// Start server on port 3000
server.listen(3000, () => {
  console.log('Server running on http://localhost:3000');
});
```

Output:

localhost:3000/menu


```
[
    {
        "id": 1,
        "name": "Pizza",
        "price": 10
    },
    {
        "id": 2,
        "name": "Burger",
        "price": 5
    },
    {
        "id": 3,
        "name": "Pasta",
        "price": 8
    }
]
```

localhost:3000/order/1

```
{
    "message": "Order placed for Pizza"
}
```

Ex.no: 40

AIM: Create a Node JS application for user login system.

Server.js

```
const http = require("http");
const url = require("url");
const querystring = require("querystring");

// Mock user data (in-memory storage)
const users = {
    "testuser": "password123", // username: password
};

// HTML pages for responses
const loginPage = `
<!DOCTYPE html>
<html>
<head>
    <title>Login</title>
</head>
<body>
    <h1>Login</h1>
    <form method="POST" action="/login">
        <label for="username">Username:</label><br>
        <input type="text" id="username" name="username" required><br><br>
        <label for="password">Password:</label><br>
        <input type="password" id="password" name="password" required><br><br>
        <button type="submit">Login</button>
    </form>
</body>
</html>
`;
```

```
const successPage = `
<!DOCTYPE html>
<html>
<head>
    <title>Success</title>
</head>
<body>
    <h1>Login Successful!</h1>
    <a href="/">Go back</a>
</body>
</html>
`;

const failurePage = `
<!DOCTYPE html>
<html>
<head>
    <title>Failure</title>
</head>
<body>
    <h1>Login Failed</h1>
    <p>Invalid username or password.</p>
    <a href="/">Try Again</a>
</body>
</html>
`;

// Create HTTP server
const server = http.createServer((req, res) => {
    const parsedUrl = url.parse(req.url);
    const method = req.method;

    if (parsedUrl.pathname === "/" && method === "GET") {
        // Serve the login page
        res.writeHead(200, { "Content-Type": "text/html" });
        res.end(loginPage);

    } else if (parsedUrl.pathname === "/login" && method === "POST") {
        // Handle login form submission
        let body = "";

        // Collect POST data
        req.on("data", (chunk) => {
            body += chunk.toString();
        });

        req.on("end", () => {
            const { username, password } = querystring.parse(body);

            if (users[username] && users[username] === password) {
```

```
            // Login successful
            res.writeHead(200, { "Content-Type": "text/html" });
            res.end(successPage);
        } else {
            // Login failed
            res.writeHead(401, { "Content-Type": "text/html" });
            res.end(failurePage);
        }
    });

  } else {
    // Handle 404
    res.writeHead(404, { "Content-Type": "text/html" });
    res.end("<h1>404 Not Found</h1>");
  }
});

// Start the server
const PORT = 3000;
server.listen(PORT, () => {
  console.log(`Server running at http://localhost:${PORT}`);
});
```

Output:

localhost:3000

Login

Username:

acdfg

Password:

•••••••

Login

localhost:3000/login

Login Failed

Invalid username or password.

Try Again