

# PRACTICAL NO. 1

**AIM:** Installation of Visual studio code and Node.js on windows.

### 1.) Visual Studio Code

**Steps to Install Visual Studio Code on Windows :**

**Step 1:** Visit the Official Website of the Visual Studio Code using any web browser. ( [https://code.visualstudio.com/docs/?dv=win )](https://code.visualstudio.com/docs/?dv=win)

**Step 2:** Press the “Download for Windows” button on the website to start the download of the Visual Studio Code Application.

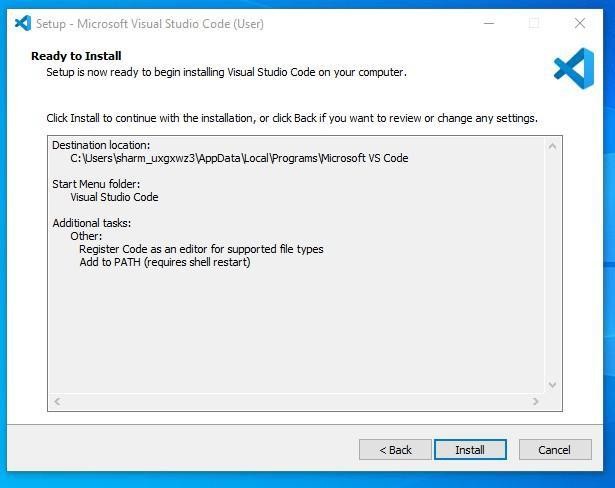
**Step 3:** When the download finishes, then the Visual Studio Code Icon appears in the downloads folder.

**Step 4:** Click on the Installer icon to start the installation process of the Visual Studio Code.

**Step 5:** After the Installer opens, it will ask you to accept the terms and conditions of the Visual Studio Code. Click on I accept the agreement and then click the Next button.

**Step 6:** Choose the location data for running the Visual Studio Code. It will then ask you to browse the location. Then click on the Next button.

**Step 7:** Then it will ask to begin the installation setup. Click on the Install button.



**Step 8:** After clicking on Install, it will take about 1 minute to install the Visual Studio Code on your device.

**Step 9:** After the Installation setup for Visual Studio Code is finished, it will show a window like this below. Tick the “Launch Visual Studio Code” checkbox and then click Next.



# 2)Node.js:

### Steps to Install Node.js on Windows :

**Step 1:** Visit the Official Website of the Node.js using any web browser**. (** [https://nodejs.org/en/download/package-manager **)**](https://nodejs.org/en/download/package-manager)

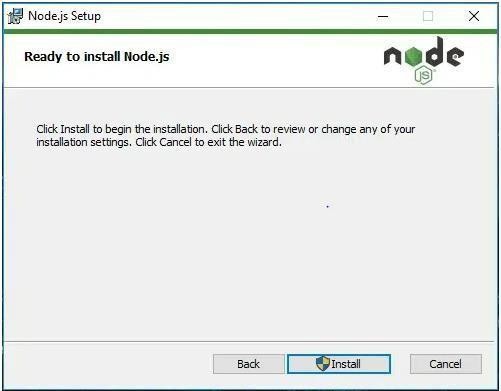
**Step 2:** Go to Prebuilt Installer and Select the Versions to download.



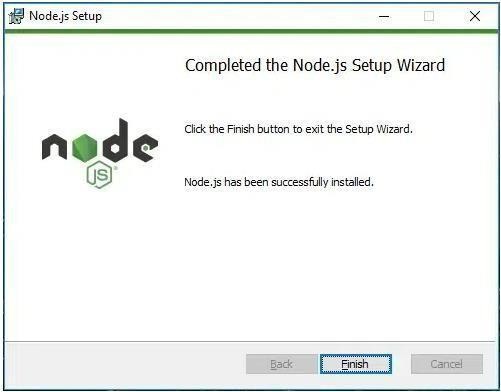
Once You download the .msi installer follow the next step.

**Step 3:** After download, Run the Node.js Installer and select “I accept the terms in the License Agreement” and select “Next”.

**Step 4:** Finish Setup and Select “Install” to complete the Node.js installation process.



Wait for “Finish” to complete the setup.



# PRACTICAL NO.2

## To Perform REPL in Node.js

console.log(5 + 5);

console.log(5 - 2);

console.log(10 / 2);

console.log(5 \* 4);

function add(a, b) { return a + b;

}

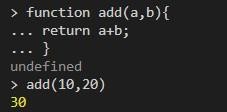
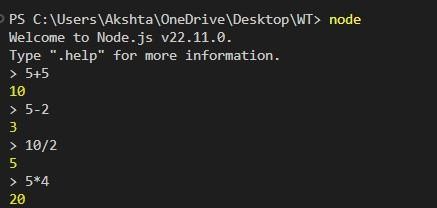
console.log(add(10, 20));

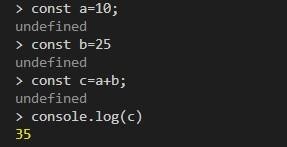
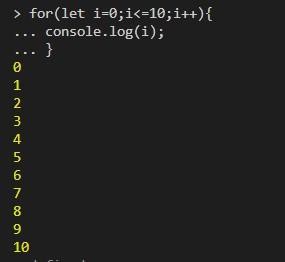
for (let i = 0; i <= 10; i++) { console.log(i);

}

const a = 10; const b = 20; const c = a + b; console.log(c);

**OUTPUT:**

****

****

# PRACTICAL NO.3

## Demonstrate the basic arithmetic operations in Node.js

function sum(a, b) { return a + b;

}

function sub(a, b) { return a - b;

}

function mul(a, b) { return a \* b;

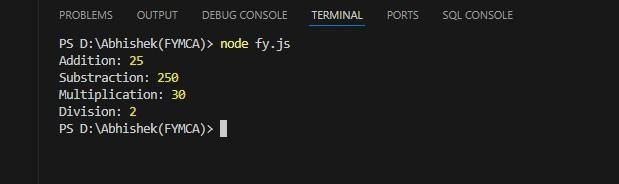
}

function div(a, b) { return a / b;

}

console.log("Addition: ", sum(5, 5)); console.log("Subtraction: ", sub(3, 2)); console.log("Multiplication", mul(4, 8)); console.log("Division: ", div(6, 2));

**OUTPUT:**

****

# PRACTICAL NO.4

## To determine whether a given number is even or odd in Node.js

function displayresult(a) { console.log(a);

}

function check(num) { let sum = num;

if (num % 2 == 0) { console.log("Number is Even")

} else {

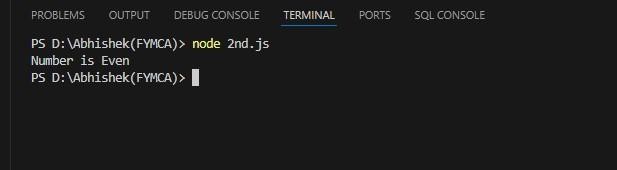
console.log("Number is odd")

}

}

check(18, displayresult)

**OUTPUT:**

****

# PRACTICAL NO.5

## To print all prime numbers up to a given number in Node.js.

function isPrime(n)

{if(n==1||n==0) return false; for(var i=2;i<n;i++){ if(n%i==0) return false;

} return true;

}

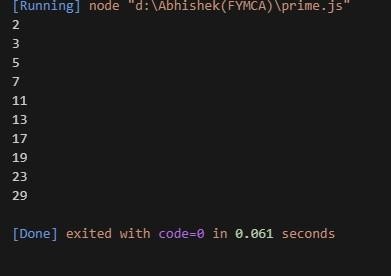
var num =30;

for(var i=1;i<=num;i++){ if(isPrime(i)){ console.log(i);

}

}

**OUTPUT:**

****

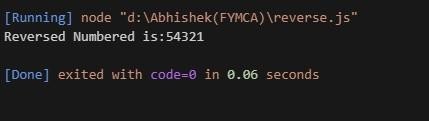
# PRACTICAL NO.6

## Create an application in NodeJS to reverse the given number and display it (Note: 5 digit number)

var number = 12345;

var reversedNumber = number.toString().split('').reverse().join(''); console.log('Reversed number is: ' + reversedNumber);

**OUTPUT:**

****

# PRACTICAL NO.7

## Create an application in Node.js to display Armstrong number 15.

function isArmstrongNumber(num) { let sum = 0;

const strNum = String(num); const len = strNum.length; for (let i = 0; i < len; i++) {

sum += Math.pow(Number(strNum[i]), len);

}

return sum === num;

}

function printFirstNArmstrongNumbers(n) { let count = 0;

let num = 1;

while (count < n) {

if (isArmstrongNumber(num)) { console.log(num);

count++;

}

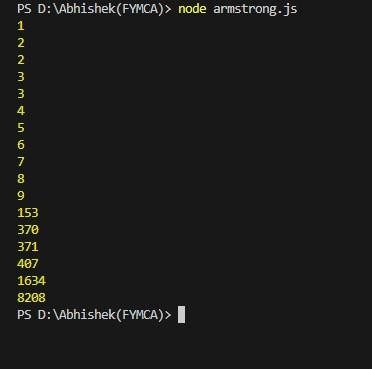
num++;

}

}

printFirstNArmstrongNumbers(15);

**OUTPUT:**

****

# PRACTICAL NO.8

Roll No:04/A

## To generate the first 10 numbers in the Fibonacci sequence in Node.js.

var a=0; var b=1; var c;

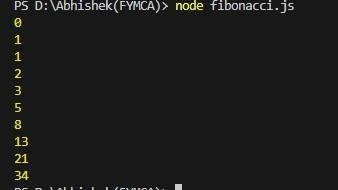
console.log(a); console.log(b); for(i=0;i<8;i++)

{ c=a+b; console.log(c); a=b;

b=c;

}

**OUTPUT:**

****

# PRACTICAL NO.9

## To demonstrate the use of set Timeout and arrow functions in Node.js.

const message = function(){ console.log("Hello NodeJS, Welcome");

}

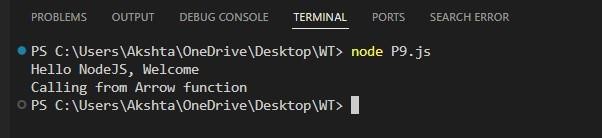
setTimeout(message,5000)

; setTimeout(()=> {

console.log("Calling from Arrow function");

},8000);

**OUTPUT:**

****

# PRACTICAL NO.10

## To demonstrate module exports in Node.js.

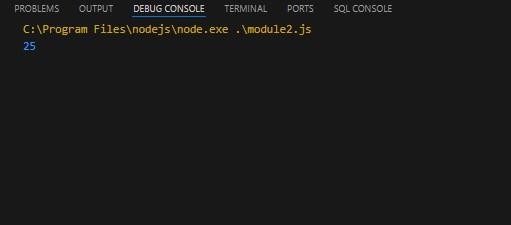
function add(a,b){ return a+b;

}

exports.add=add;

var req = require('./Node8\_1'); var res = req.add(15,9); console.log(res);

**OUTPUT:**

****

# PRACTICAL NO.11

## write an application to find area of circle, square,rectangle using module in Node.js.

function square(s){ return s\*s;

}

function rectangle(l,b){ return l\*b;

}

function circle(r){ return 3.14\*r\*r;

} exports.square=square; exports.rectangle=rectangle

; exports.circle=circle;

var req = require('./Node9\_1'); var sRES,rRes,cRes; sRES=req.square(5); rRes=req.rectangle(4,6); cRes=req.circle(4); console.log("square:",sRES); console.log("rectangle:",rRes)

; console.log("circle:",cRes);

**OUTPUT:**

****

# PRACTICAL NO.12

## Write an application to demonstrate events module in Node.js.

const EventEmitter = require('events'); const emitter = new EventEmitter();

//REGISTER

emitter.on('messageLogged',function (){ console.log('Listener called');

});

//Raise emitter.emit('messageLogged'

)

**OUTPUT:**

****

# PRACTICAL NO.13

## write an application to demonstrate function (removeListner, listnerCount) in Node.js.

const events = require("events");

const eventEmitter = new events.EventEmitter(); function listner1(){

console.log("Event received by Listner 1");

}

function listner2(){

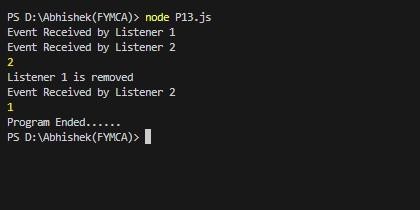
console.log("Event received by Listner2");

} eventEmitter.addListener("Write",listner1); eventEmitter.on("Write",listner2); eventEmitter.emit("Write"); console.log(eventEmitter.listenerCount("write"))

; eventEmitter.removeListener("write",listner1); console.log("Listner1 is removed"); eventEmitter.emit("write"); console.log(eventEmitter.listenerCount("write"))

; console.log("program Ended. ")

**OUTPUT:**

****

# PRACTICAL NO.14

## create an application in node.js to Return Event Emitter.

var emitter = require('events').EventEmitter; function LoopProcessor(num){

var e = new emitter(); setTimeout(function(){ for(var i=1;i<=num;i++){ e.emit('BeforeProcess ',i);

console.log('processing number: '+i); e.emit('AfterProcess ',i);

}

}, 2000)

return e;

}

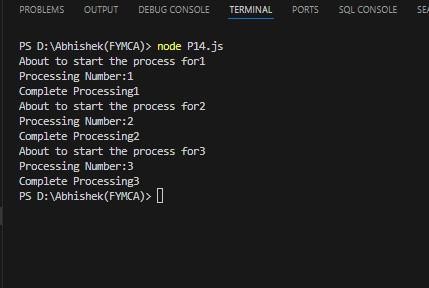
var lp = LoopProcessor(3); lp.on('BeforeProcess', function(data){ console.log('About to start the process for '+data);

});

lp.on('AfterProcess', function(data){ console.log('Completed processing '+data);

});

**OUTPUT:**

****

# PRACTICAL NO.15

## create an application in node.js to create Extend Event Emitter in Node.js.

var emitter=require('events').EventEmitter; var util = require('util');

function LoopProcessor (num) { var me = this; setTimeout(function(){

for (var i=1;i<=num;i++){ me.emit ('BeforeProcess',i); console.log('processing number: '+i); me.emit ('After Process',i);

}

}, 2000)

return this;

}

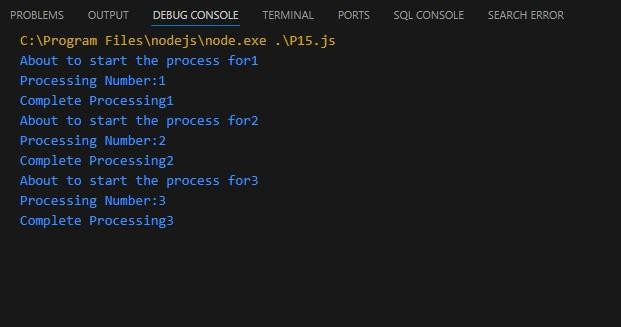
util.inherits (LoopProcessor, emitter) var lp = new LoopProcessor (3); lp.on('BeforceProcess', function(data) { console.log('About to start the process for' + data);

});

lp.on('AfterProcess', function(data) { console.log('completed processing '+ data);

});

**OUTPUT:**

****

# PRACTICAL NO.16

## Write an event emitter code to design an event called as “calculate Salary” which is used to calculate the salary of an employee by passing some arguments like Basic Salary, HRA (20% of Basic), DA(100% of Basic), TA, and deductions like Income Tax (30% of Basic) and Professional Tax of 200.

const EventEmitter = require('events');

class SalaryCalculator extends EventEmitter { calculateSalary(basic, ta) {

const hra = 0.2 \* basic; // HRA is 20% of Basic const da = basic; // DA is 100% of Basic

const incomeTax = 0.3 \* basic; // Income Tax is 30% of Basic const professionalTax = 200; // Professional Tax is 200

const salary = basic + hra + da + ta - incomeTax - professionalTax; this.emit('calculateSalary', salary);

}

}

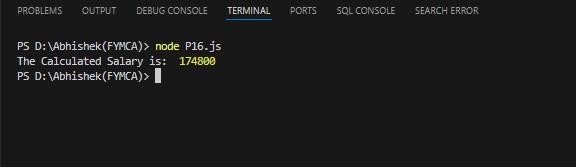
const salaryCalculator = new SalaryCalculator(); salaryCalculator.on('calculateSalary', (salary) => { console.log(`The calculated salary is: ${salary}`);

});

// Example usage:

salaryCalculator.calculateSalary(50000, 8000); // Basic Salary is 50000 and TA is 8000

**OUTPUT:**

****

# PRACTICAL NO.17

## create an application in node.js to display message after 5 second & 10 second.

const myfun = delay => {

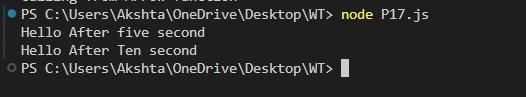
console.log('Hello After ' +delay+ ' second');

};

setTimeout(myfun,5000,'five'); setTimeout(myfun,10000,'Ten')

;

**OUTPUT:**

****

# PRACTICAL NO.18

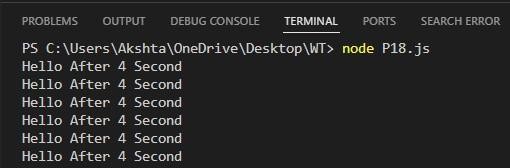
## create an application in node.js to demonstrate set interval function.

setInterval(

() => console.log('Hello After 4 Second'),4000

);

**OUTPUT:**

****

# PRACTICAL NO.19

## create an application in node.js to display factorial of a number.

function factorial(n){ let i=n;

let res=1; while(i>+1){ res = res\*i;

i--

}

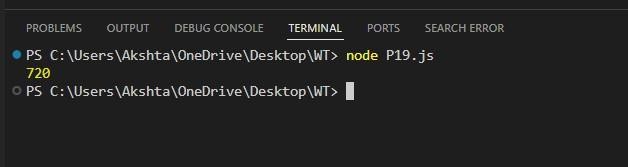
return res;

}

const num = 6;

const result = factorial(num); console.log(result);

**OUTPUT:**



# PRACTICAL NO.20

## Write as application to create http Server and Display message in Node.js.

var http = require('http');

var server = http.createServer(function(req,res){ res.write("Hello from server");

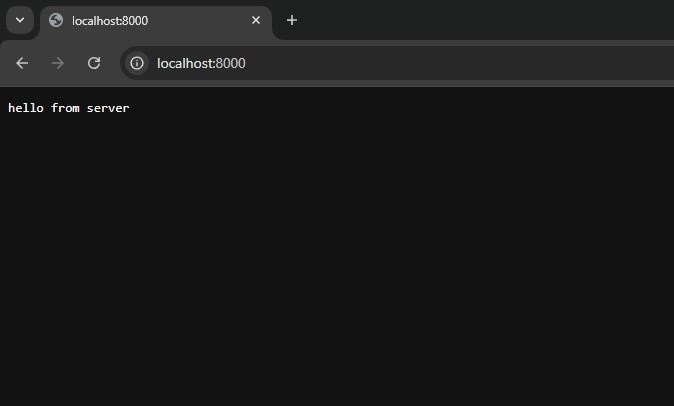
res.end();

});

server.listen(5000);

console.log('Node.js web server at port 5000 is running.. <http://localhost:5000/>')

**OUTPUT:**

****

# PRACTICAL NO.21

## Write a Node.js code to display Employee Job Registration Form saved in an HTML file in response to the client’s access request to the server.

const http = require('http'); const fs = require('fs'); http.createServer((req, res) => { fs.readFile('form.html', (err, data) => { if (data) {

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

}

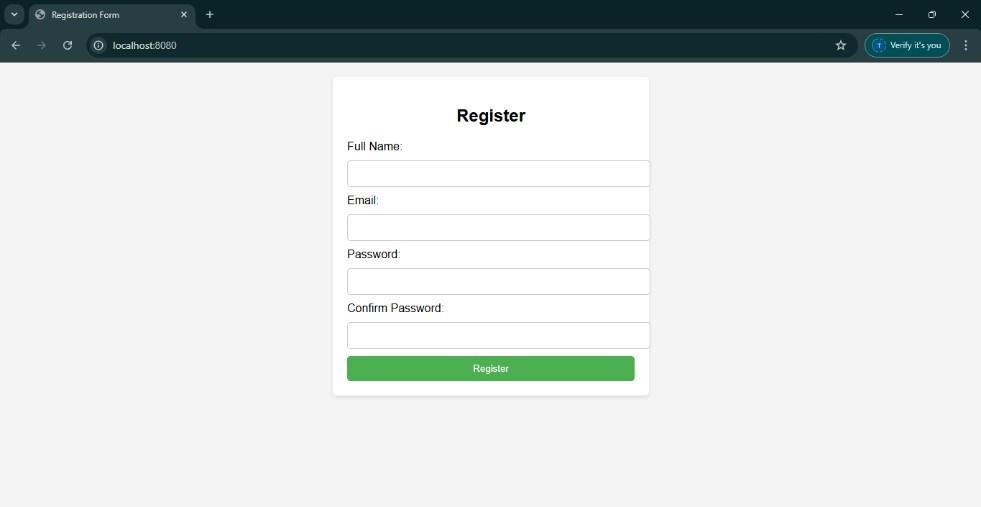
});

}).listen(8080, () => {

console.log('Server is running at [http://localhost:8080');](http://localhost:8080/)

});

**OUTPUT:**

****

# PRACTICAL NO.22

## Write as application to create Home page, Admin page and Student page using http server in Node.js.

var http = require('http');

const{ text } = require('stream/consumers');

var server =http.createServer(function(req,res){ if(req.url=='/'){

res.writeHead(200,{'content-type':'text/html'}); res.write('<html></head><body>');

res.write('<style>ul li{display: inline-block; float: right; height:

40px;} ul li a{padding: 20px; background:orange; color: white;}</style>'); res.write('<div><h1>Mr.Raj Website</h1></div><div><ul><li><a href="/admin">Contact Admin</a></li><li><a href="/student">Student</a></li><li><a href="/home">Home</a></li></ul></div></div>');

res.write('<div style="background: white; padding: 20px;"><h2>Start Page</h2><p>This is my first webpage!</p><p>Hi

everyone</p></div></body></html>'); res.end();

}

else if(req.url=='/home')

{

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

res.write('<html><head><style>body{padding-left: 43px; padding-right:43px; background- color:lightyellow;} </style></head><body><p><h1>This is

home page</h1></p><h1>Raj Pawar</h1><h3>This page is a brief insight to who I am.</h3>');

res.write('<nav style="background-color:white; text-align:center;"><ul><li><a href="/">Start Page</a></li><li><a href="/student">Student</a></li><li><a

href="/admin">Admin</a></li></ul></nav></body></html>'

); res.end();

}

else if (req.url=='/student')

{

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

res.write('<div style="display: inline-block; float: right; height:40px; padding: 20px;"><ul><li><a href="/home">Home</a></li><li><a

href="/">Start Page</a></li> <li><a href="/admin">Contact Admin</a></li></ul></div>');

res.write('<html><head><style>body{background- color:pink;}</style><title>Form</title></head><body bgcolor="White" ><h1

align="center">Student Page Form</h1>');

res.write('<form action="url" method="post"><fieldset><legend>Personal Imformation</legend>');

res.write('<lable><Strong>Student Name</strong></lable><br/><input type="text" name="Student Name" placeholder="Enter Your Name" /><br/>');

res.write('<lable><Strong>Email</strong></lable><br/><input type="email" name="eamil" placeholder="Enter Your Email Address" /></br>');

res.write('<lable><Strong>Password</strong></lable><br/>');

res.write('<input type="password" name="Password" placeholder="Enter Your Password"

/></br><lable><Strong>Gender</strong></lable><br/>');

res.write('<input type="Radio" name="Gender" value="Male" />Male <input type="Radio" name="Gender" value="FeMale" />FeMale<br/>');

res.write('<lable><Strong>Hobbies</strong></lable><br/>');

res.write('<input type="checkbox" name="Hobbies" value="Playing Sports" />Playing Sports<br/>'); res.write('<input type="checkbox" name="Hobbies" value="Listening Music" />Listening Music<br/>'); res.write('<input type="checkbox" name="Hobbies" value="Traveling"

/>Traveling<br/>'); res.write('<input type="checkbox" name="Hobbies" value="Reading Books"

/>Reading Books<br/>'); res.write('<lable><Strong>Select Your City</strong></lable><select name="City">');

res.write('<option value="Mumbai">Mumbai</option><option value="Gujrat">Gujrat</option><option value="Pune">Pune</option>');

res.write(' <option value="Thane">Thane</option></select></br><input type="submit" onclick=alert("Thanks!") name="submit"

value="Submit"/></form>'); res.end();

}

else if (req.url=='/admin')

{

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

res.write('<style>ul li{display: inline-block; float: right; height:40px;} ul li a{padding: 20px; background:orange; color: white;}</style>');

res.write('<div><ul><li><a href="/admin">Contact Admin</a></li><li><a href="/student">Student</a></li><li><a href="/home">Home</a></li></ul></div></div><br><br>'); res.write('<html><head><style>legend{text-align:center;}

body{background-color:faf89a;border: 5px solid darkred;} form{display: inline- block; float: center; padding: 20px;} ');

res.write('border-radius:4px; padding:40px 5px; max-width:100%;}</style></head>'); res.write('<legend><h1><u>Admin Login</u></h1></legend>');

res.write('<form action="#" method="POST" autocomplete="off">');

res.write('<div class="input\_field"><h3>Username</h3></div><div

class="input\_field"><input type="text" ');

res.write('name="userid" placeholder="Username" required/></div>');

res.write('<div class="input\_field"><h3>Password</h3></div><div class="input\_field"><input type="Password"');

res.write('name="pword" placeholder="Password" required/></div><p>');

res.write('<style>button{border:none; border-radius:5px; text-align:center; padding:15px 15px; background-color:lavender;<div></div></style>');

res.write('<button onclick=alert("SUCESS")>LOGIN NOW</button></form>'); res.end();

}

else{

res.end('Invalid request');

}

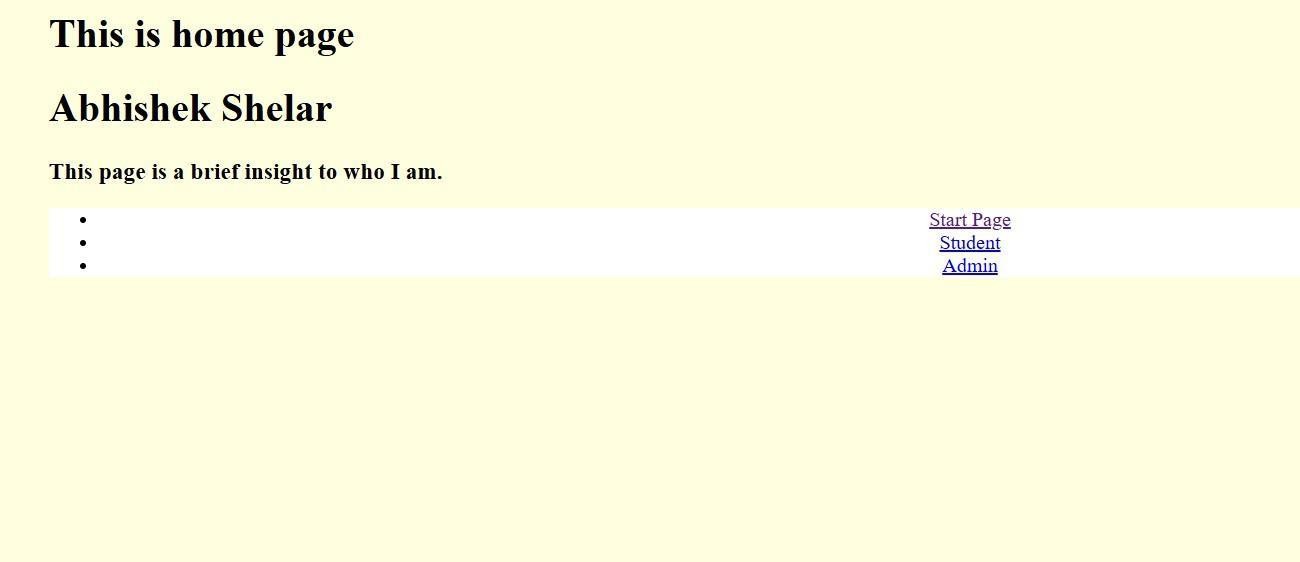
});

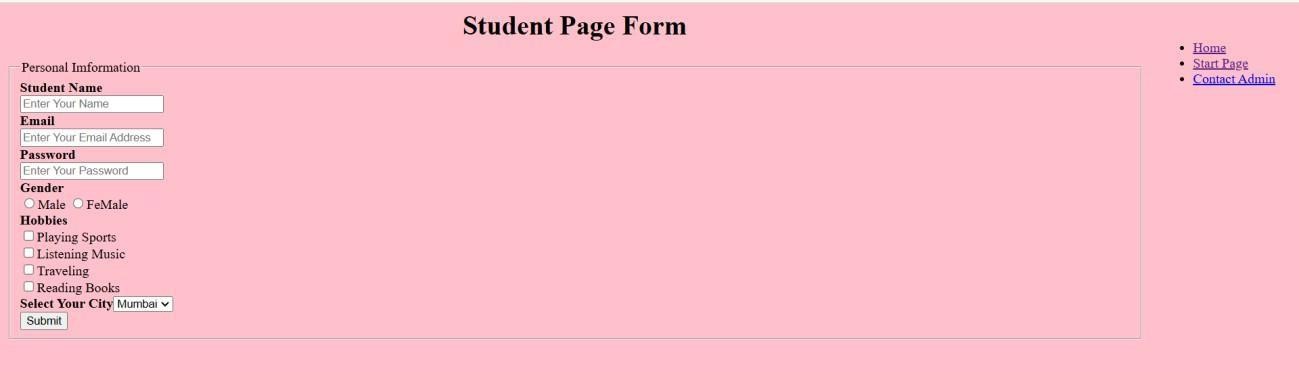
server.listen(9000);

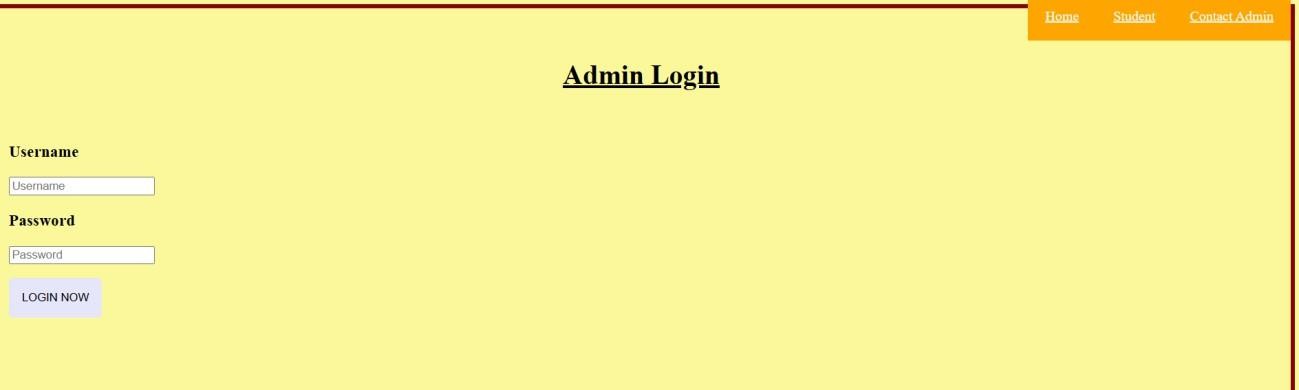
console.log('Node.js web server at port 9000 is running');

**OUTPUT:**

****



****

****

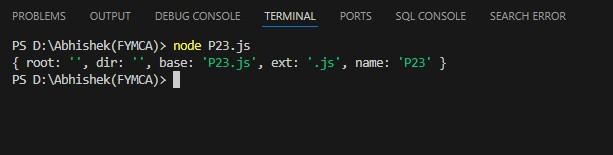
# PRACTICAL NO.23

## Write in application to display details of the current file path in Node.js.

const location = require("path");

const localobj = location.parse( filename); console.log(localobj);

**OUTPUT:**

****

# PRACTICAL NO.24

## Write an application to read file in Node.js.

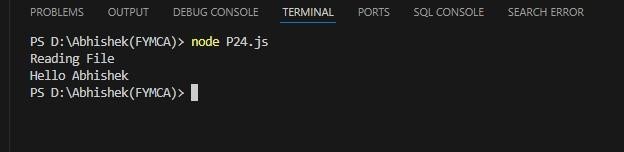
const fs = require('fs'); fs.readFile("\_txt.txt",'utf8',function(err,data)

{

console.log("Reading File"); console.log(data);

});

**OUTPUT:**

****

# PRACTICAL NO.25

## Write an application to write in file in Node.js.

const fs = require("fs");

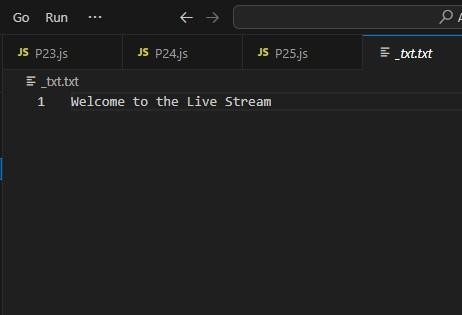
fs.writeFile("\_txt.txt",'Welcome to the live stream',function (err,data)

{

console.log("Writing File");

});

**OUTPUT:**

****

# PRACTICAL NO.26

## Write an application to add data in file in Node.js.

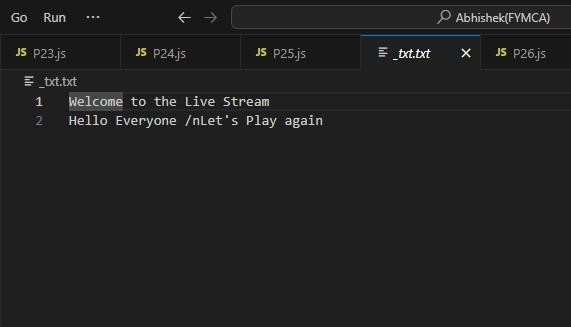
const fs = require("fs");

fs.appendFile("\_txt.txt","\nHello Everyone \nLet's play agian", function (err,data){

console.log("append file");

});

**OUTPUT:**

****

# PRACTICAL NO.27

## Write an application to delete a file in Node.js.

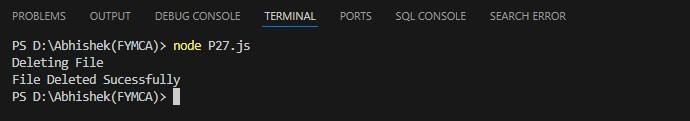
const fs = require("fs"); fs.unlink("txt2.txt",function(err,data)

{

console.log("Deleting File"); console.log("File Deleted Succesfully");

});

**OUTPUT:**

****

# PRACTICAL NO.28

## Combine Read, Write, Append, Delete file in one program in Node.js.

const fs = require("fs");

fs.writeFile("\_com.txt",'Hello world',function (err,data)

{

console.log("Writing File");

});

fs.appendFile("\_com.txt","\nHello Everyone \nGive ThumbsUp",function (err,data)

{

console.log("append file");

});

fs.readFile("\_com.txt",'utf8',function(err,data)

{

});

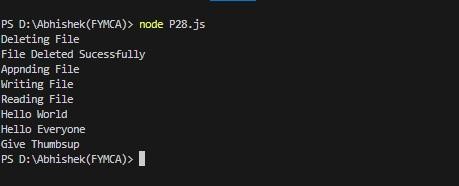
console.log("Reading File"); console.log(data);

fs.unlink("\_com.txt",function(err,data)

{

});

**OUTPUT:**

****

# PRACTICAL NO.29

## Write and application to rename a file in Node.js.

var fs = require('fs') fs.rename('\_txt.txt','Mr\_raj.txt',function(err)

{ if(err) throw err; console.log('Filed Rename')

});

**OUTPUT:**

****

# PRACTICAL NO.30

## Create an Application to create Database in Node.js.

var mysql = require('mysql')

var con = mysql.createConnection({ host:'localhost',

user:'root', password:'root'

});

con.connect(function(err)

{ if(err){ throw err;} else{

console.log("connected");}

con.query("CREATE DATABASE STUDENT1", function(err,result){ if(err) throw err;

console.log("Database Created");

});

});

**OUTPUT:**

****



# PRACTICAL NO.31

## Create an Application to create Student table with columns as id, name, address, course, contact in Node.js.

var mysql = require('mysql');

var con = mysql.createConnection({ host: 'localhost',

user: 'root', password: '12345', database: 'rahul'

});

con.connect(function(err) { if (err) throw err; console.log("connected...")

;

var sql = "CREATE TABLE student1 (id INT(10) PRIMARY KEY AUTO\_INCREMENT, name VARCHAR(255), address VARCHAR(255), course VARCHAR(20), contact INT(15))";

con.query(sql, function(err, result) { if (err) throw err; console.log("Table created");

});

**OUTPUT:**

****

# PRACTICAL NO.32

## Create an Application to create Student table with columns as id, name, address, course, contact in Node.js.

var mysql = require('mysql');

var con = mysql.createConnection(

{

host: 'localhost', user: 'root', password: '12345', database: 'rahul'

}

);

con.connect(function(err) { if (err) throw err; console.log("connected...")

;

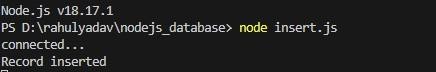
var sql2 = "INSERT INTO student1 (id, name, address, course, contact) VALUES ('1', 'yadav', 'vashi', 'MCA', '84220')";

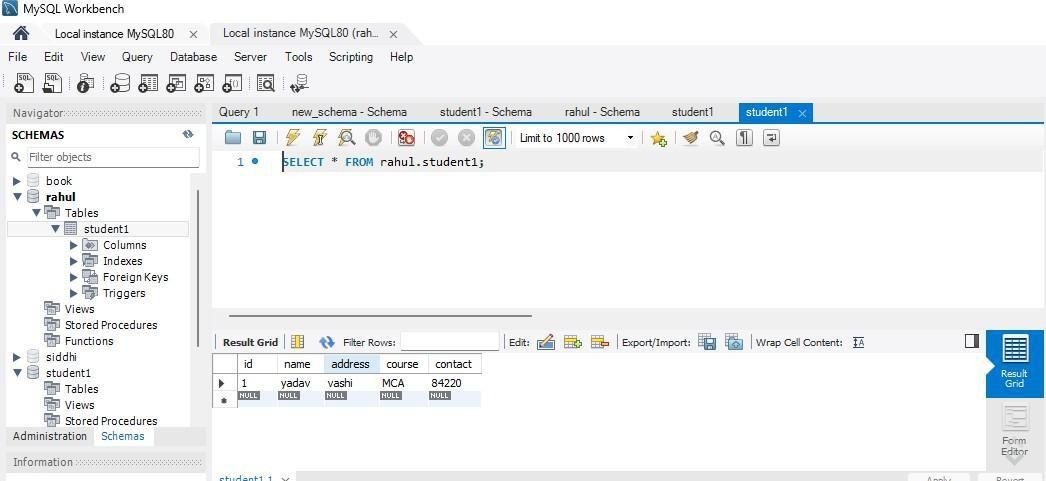
con.query(sql2, function(err, result) { if (err) throw err; console.log("Record inserted");

});

});

**OUTPUT:**

****

****

# PRACTICAL NO.33

## Create an Application to display rows into Student table in Node.js.

var mysql = require('mysql');

var con = mysql.createConnection(

{

host: 'localhost', user: 'root', password: '12345', database: 'rahul'

}

);

con.connect(function(err) { if (err) throw err; console.log("connected...")

;

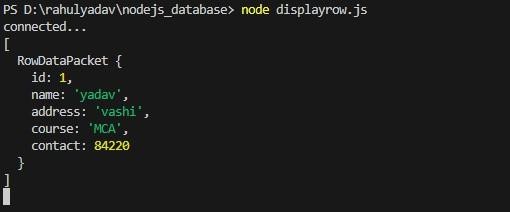
var sql2 = "SELECT \* FROM student1"; con.query(sql2, function(err, result) {

if (err) throw err; console.log(result);

});

});

**OUTPUT:**

****

# PRACTICAL NO.34

## Create an Application to Update rows in Student table in Node.js.

var mysql = require('mysql');

var con = mysql.createConnection(

{

host: 'localhost', user: 'root', password: '12345', database: 'rahul'

}

);

con.connect(function(err) { if (err) throw err; console.log("connected...")

;

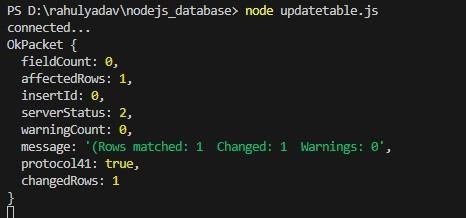
var sql2 = "UPDATE student1 SET course = 'MBA' WHERE id = '1'"; con.query(sql2, function(err, result) {

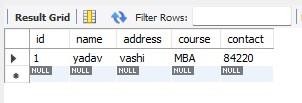
if (err) throw err; console.log(result);

});

});

**OUTPUT:**

****

****

# PRACTICAL NO.35

## Write a Node.js application to retrieve and update the record related to the entries received for the conference participation. Update the mobile number of participant whose name is “Yadav”.

var mysql = require('mysql');

var con = mysql.createConnection({ host: "localhost",

user: "root", password: "12345", database: "rahul"

});

con.connect(function(err) { if (err) throw err;

console.log("Connected successfully to server");

var sql = "SELECT \* FROM student1 WHERE name = 'yadav'"; con.query(sql, function(err, result) { if

(err) throw err; console.log("student1 found: ", result);

var newMobileNumber = '1234567890';

var updateSql = `UPDATE student1 SET contact = '${newMobileNumber}' WHERE name = 'yadav'`;

con.query(updateSql, function(err, result) { if (err) throw err;

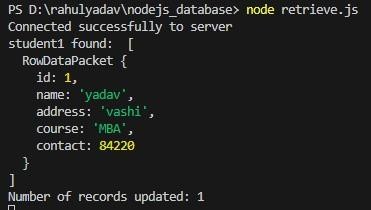
console.log("Number of records updated: " + result.affectedRows);

});

});

});

**OUTPUT:**

****

# PRACTICAL NO.36

## Create an Application to add column to Student table in Node.js.

var mysql = require('mysql');

var con = mysql.createConnection(

{

host: 'localhost', user: 'root', password: '12345', database: 'rahul'

}

);

con.connect(function(err) { if (err) throw err; console.log("connected...")

;

var sql = "ALTER TABLE student1 ADD age INT(5)"; con.query(sql, function(err, result) {

if (err) throw err;

console.log("Column inserted successfully...");

});

});

**OUTPUT:**

****

****

# PRACTICAL NO.37

## Create an Application to delete records in Student table in Node.js.

var mysql = require('mysql');

var con = mysql.createConnection(

{

host: 'localhost', user: 'root', password: '12345', database: 'rahul'

}

);

con.connect(function(err) { if (err) throw err; console.log("connected...")

;

var sql = "DELETE FROM student1 WHERE ID='1'"; con.query(sql, function(err, result) {

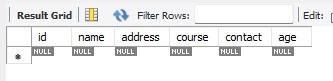
if (err) throw err;

console.log("row deleted successfuly...");

});

});





# PRACTICAL NO. 38

## AIM: To setting of React environment.

### CODE:

e

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Steps to the setup development environment** | | | | : |
| NodeJS must be installed on our computer in order to run any React application. As a result, the initial step will be to download and install NodeJS. | | | | |
| **Step 1:** | Install NodeJS first. To download and install the newest version of NodeJS befor | | | |
| installing React, go to the official NodeJS download page. After we’ve installed NodeJS on our computer, we’ll need to install React Boilerplate. | | | | |
| **Step 2:** | Configure the react environment for earlier and newer versions, and then follow anyone | | | |
| who is compatible with your node version.  Setting up React Boilerplate on Older Versions, such as Node 8.10 and npm 5.6. The boilerplate will be installed globally. Type the following line in your terminal or command prompt(cmd) in order to install the React js Boilerplate on your PC. | | | | |
| npm install -g create-react-app | | |  | |
|  | | | | |
| After running the above command in cmd and successfully installing the boilerplate, your  terminal will display some output. We’ll start building our React app after the boilerplate has  been successfully installed. The create-react-app command can be used to construct a React app.  For the Latest Versions, which include Node >=8.10 and npm >=5.6: The machine should have a version of Node >=8.10 and npm >=5.6 to use the latest features of JavaScript, which provides a pleasant development experience.  To create a new project or application, type the command given below. | | | | |
|  | |  | | |
|  | |

npx create-react-app my-app

|  |  |  |  |
| --- | --- | --- | --- |
| The command above will create the app named my-app:  The command cd my-app can be used to launch the project. Now open your browser and check the output. | | | |
| **Step 3:** | Make a React application. We’ll now use the boilerplate we loaded to develop an app. | | |
| The command below will build a new app called myapp. | | | |
| create-react-app myapp | | |  |
|  | | | |
| The command statement will create a new directory named myapp inside your current directory, which will contain all of the files needed to run a React app.  A lot of files can be found in the directory mentioned above. The index.html and index.js files will be the main focus of the introductory course. The index.html file will have a div element with the id “root” inside which everything will be rendered, and the index.js file will have all of  our React code.  Now that the development environment has been effectively set up. Last but not least, the development server must be started. | | | |
| **Step 4:** | Get the development server up and running. To start the development server, navigate to | | |
| “myapp” in your current directory and run the following command: | | | |
| npm start | |  | |
|  | | | |
| When you run the above command in cmd successfully, your compiler will prompt you to open localhost in your default browser.  To reflect the changes you’re making in your App, browse to the URL provided in the browser. When you enter the above URL into your default browser, it will display the following page. | | | |

# PRACTICAL NO.39

## Create an application in ReactJS to implement component life cycle.

import React, { useState, useEffect } from 'react'; const LifecycleComponent = () => {

const [count, setCount] = useState(0);

const [message, setMessage] = useState('Hello, World!'); useEffect(() => {

console.log('Component mounted!'); return () => {

console.log('Component will unmount!');

};

}, []);

useEffect(() => {

console.log(`Count updated to: ${count}`);

}, [count]);

const handleClick = () => { setCount(count + 1);

};

const handleMessageChange = () => { setMessage('Message has been changed!');

};

return (

<div>

<h1>React Component Lifecycle Example</h1>

<p>Message: {message}</p>

<p>Count: {count}</p>

<button onClick={handleClick}>Increment Count</button>

<button onClick={handleMessageChange}>Change Message</button>

</div>

);

};

const App = () => { return (

<div className="App">

<h2>React Component Lifecycle Demo</h2>

<LifecycleComponent />

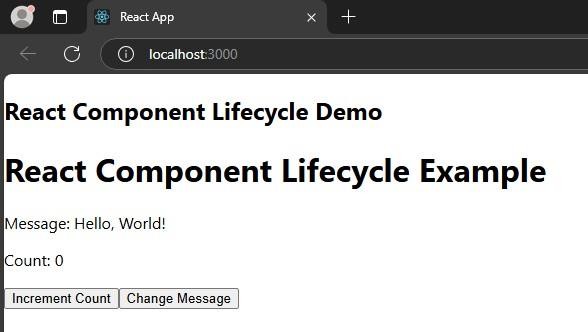
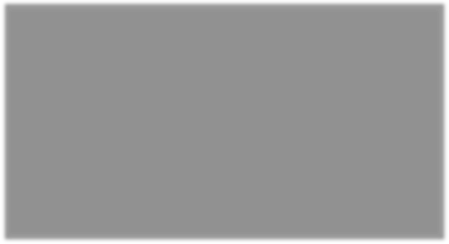
</div>

);

};

export default App;

**OUTPUT:**



# PRACTICAL NO.40

## Create an application to increment the count using Class Component in ReactJS.

import React, { Component } from 'react'; class ClassComponent extends Component { constructor(props) {

super(props); this.state = {

name: 'Class Component roll number: 56’, count: 0,

};

}

incrementCount = () => {

this.setState({ count: this.state.count + 1 });

};

render() { return (

<div>

<h2>{this.state.name}</h2>

<p>Count: {this.state.count}</p>

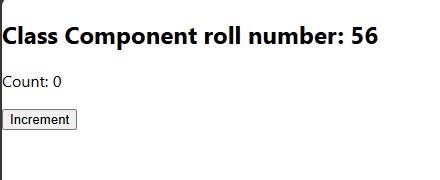
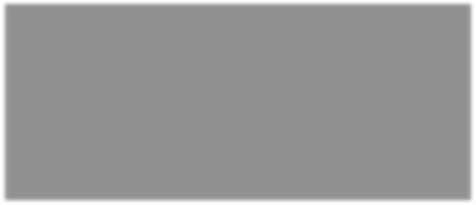
<button onClick={this.incrementCount}>Increment</button>

</div>

); }}

export default ClassComponent;

**OUTPUT:**



# PRACTICAL NO.41

## Create an application to increment the count using Functional Component in ReactJS.

import React, { useState } from 'react'; const FunctionalComponent = () => {

const [name, setName] = useState('Functional Component Roll No:56'); const [count, setCount] = useState(0);

const incrementCount = () => { setCount(count + 1);

};

return (

<div>

<h2>{name}</h2>

<p>Count: {count}</p>

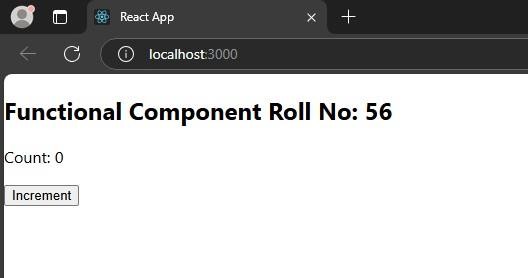
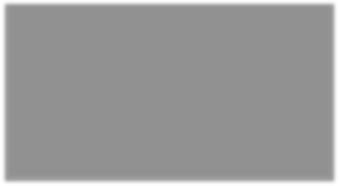
<button onClick={incrementCount}>Increment</button>

</div>

);};

export default FunctionalComponent;

**OUTPUT:**



# PRACTICAL NO.42

## Create an application to create Header, Footer and Main Content Files and import the components in to App.js File.

### Header.js:

import React from 'react'; const Header = () => { return (

<header>

<h1>Welcome to Abhishek’s React App</h1>

</header>

);

};

export default Header;

### MainContent.js:

import React from 'react';

import Header from './Header'; // Importing Header component import MainContent from './MainContent'; // Importing MainContent

component import Footer from './Footer'; // Importing Footer component function App() {

return (

<div className="App">

<Header />

<MainContent />

<Footer /

</div>

);

}

export default App;

### Footer.js:

import React from 'react'; const Footer = () => { return (

<footer>

<p>© 2024 My React App</p>

</footer>

);

};

export default Footer;

### App.js:

import React from 'react';

import Header from './Header'; // Importing Header component import MainContent from './MainContent'; // Importing MainContent

component import Footer from './Footer'; // Importing Footer component function App() {

return (

<div className="App">

<Header />

<MainContent />

<Footer />

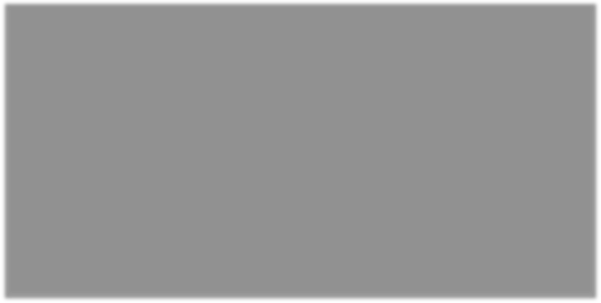
</div>

);

}

export default App;

**OUTPUT:**



# PRACTICAL NO.43

## Create an application to Display Greeting message and increment or decrement the value using State and Props.

### Greeting.js:

import React from 'react';

const Greeting = ({ name }) => { return <h2>Hello, {name}!</h2>;

};

export default Greeting;

### Counter.js:

import React, { useState } from 'react'; const Counter = () => {

const [count, setCount] = useState(0); const increment = () =>

{ setCount(count + 1);

};

const decrement = () => { setCount(count - 1);

};

return (

<div>

<h3>Counter: {count}</h3>

<button onClick={increment}>Increment</button>

<button onClick={decrement}>Decrement</button>

</div>

);

};

export default Counter;

### App.js:

import React, { useState } from 'react'; import Greeting from './Component/Greeting'; import Counter from './Component/Counter'; function App() {

const [userName, setUserName] = useState('Abhishek'); return (

<div className="App">

<Greeting name={userName} />

<Counter />

<button onClick={() => setUserName('Shelar')}>Change Name</button>

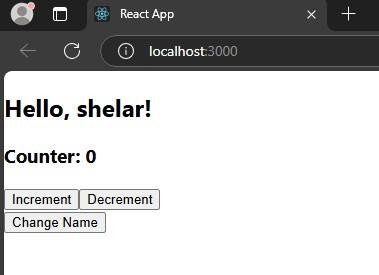
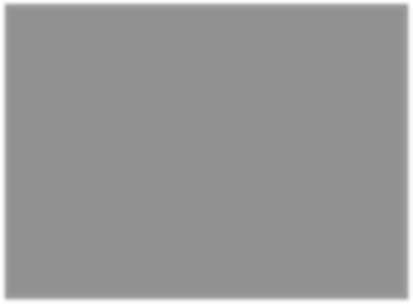
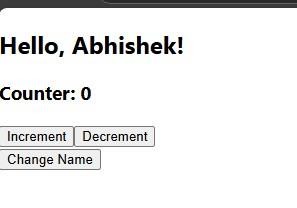
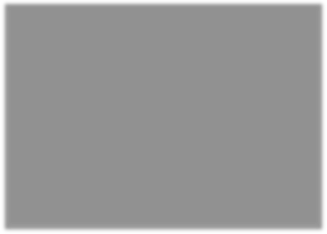
</div>

);

}

export default App;

**OUTPUT:**



# PRACTICAL NO.44

## Create an application in ReactJS to implement Event Handler (for Mouse Over Event, Button Click, Key Down Event) using DOM events.

### ButtonClick.js:

import React, { useState } from 'react'; const ButtonClick = () => {

const [message, setMessage] = useState('Click the button to change this message.'); const handleClick = () => {

setMessage('Button was clicked!');

};

return (

<div>

<button onClick={handleClick}>Click Me</button>

<p>{message}</p>

</div>

);

};

export default ButtonClick;

### Form.js:

import React, { useState } from 'react'; const Form = () => {

const [inputValue, setInputValue] = useState(''); const [submittedValue, setSubmittedValue] = useState(''); const handleChange = (event) => { setInputValue(event.target.value);

};

const handleSubmit = (event) => { event.preventDefault(); // Prevents the page from refreshing setSubmittedValue(inputValue);

};

return (

<div>

<form onSubmit={handleSubmit}>

<input type="text" value={inputValue} onChange={handleChange

}

placeholder="Type something"

/>

<button type="submit">Submit</button>

</form>

<p>Submitted Value: {submittedValue}</p>

</div>

);

};

export default Form;

### App.js:

import React from 'react';

import ButtonClick from './Component/ButtonClick'; import Form from './Component/Form';

function App() { return (

<div className="App">

<h1>React DOM Events Example</h1>

<ButtonClick />

<Form />

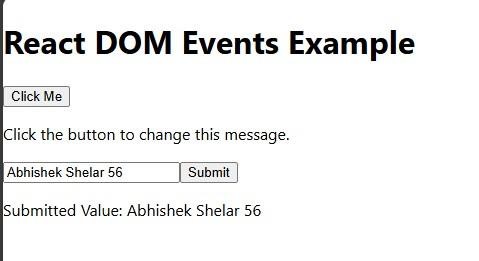
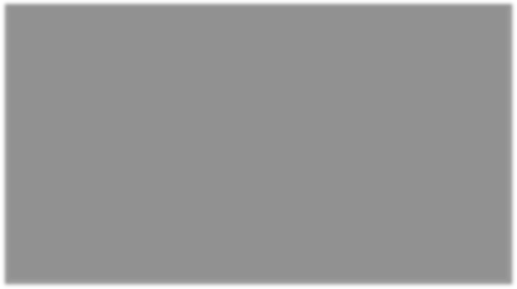
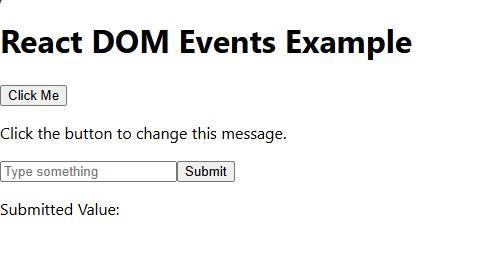
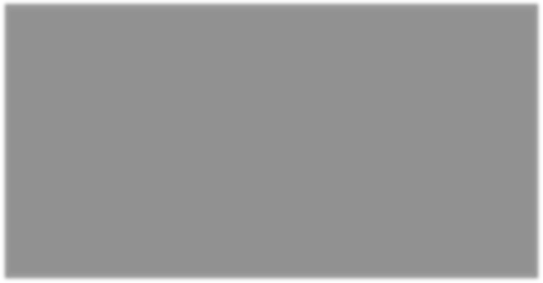
</div>

);

}

export default App;

**OUTPUT:**



# PRACTICAL NO.45

## Create an application in ReactJS form and add client and server side validation.

### App.js:

Import react, { useState } from 'react'; import axios from 'axios';

import './App.css';

function App() {

// State for form data and validation messages const [formData, setFormData] = useState({ username: '',

email: '',

password: '',

});

const [errors, setErrors] = useState({ username: '',

email: '',

password: '',

});

const [serverError, setServerError] = useState('');

// Handle input changes

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

...formData,

[name]: value,

});

};

// Client-side validation function const validateForm = () => { const newErrors = {

username: '',

email: '',

password: '',

};

// Username validation

if (!formData.username) { newErrors.username = 'Username is required';

}

// Email validation

const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/; if (!formData.email) {

newErrors.email = 'Email is required';

} else if (!emailRegex.test(formData.email)) { newErrors.email = 'Please enter a valid email';

}

// Password validation

if (!formData.password) { newErrors.password = 'Password is required';

} else if (formData.password.length < 6) {

newErrors.password = 'Password must be at least 6 characters';

}

setErrors(newErrors);

return Object.values(newErrors).every((error) => error === '');

};

// Handle form submission

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

// Client-side validation if (!validateForm()) { return;

}

// Simulate a server-side validation (e.g., via an API call) try {

const response = await axios.post('https://jsonplaceholder.typicode.com/posts', formData); if (response.status === 201) {

alert('Form submitted successfully'); setServerError('');

setFormData({ username: '',

email: '',

password: '',

});

}

} catch (error) {

setServerError('Server validation failed. Please try again.');

}

};

return (

<div className="App">

<h1>React Form with Client and Server-Side Validation</h1>

<form onSubmit={handleSubmit}>

<div>

<label>Username:</label>

<input type="text" name="username" value={formData.username

}

onChange={handleChange}

/>

{errors.username && <span className="error">{errors.username}</span>}

</div>

<div>

<label>Email:</label>

<input type="email" name="email" value={formData.email} onChange={handleChange}

/>

{errors.email && <span className="error">{errors.email}</span>}

</div>

<div>

<label>Password:</label>

<input type="password" name="password" value={formData.password

}

onChange={handleChange}

/>

{errors.password && <span className="error">{errors.password}</span>}

</div>

{serverError && <div className="server-error">{serverError}</div>}

<button type="submit">Submit</button>

</form>

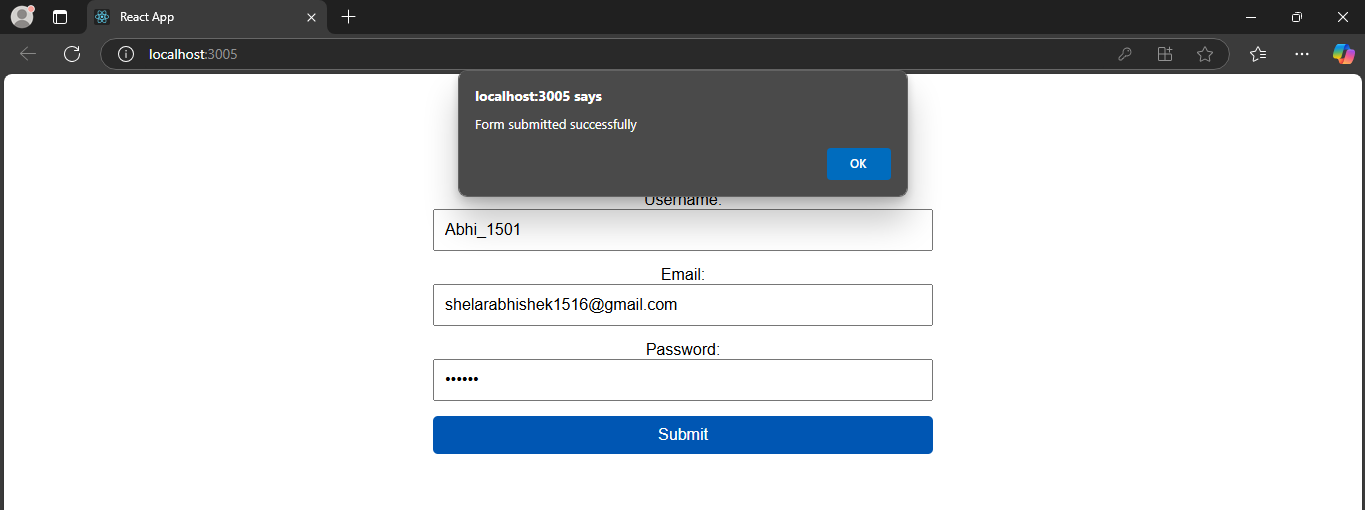
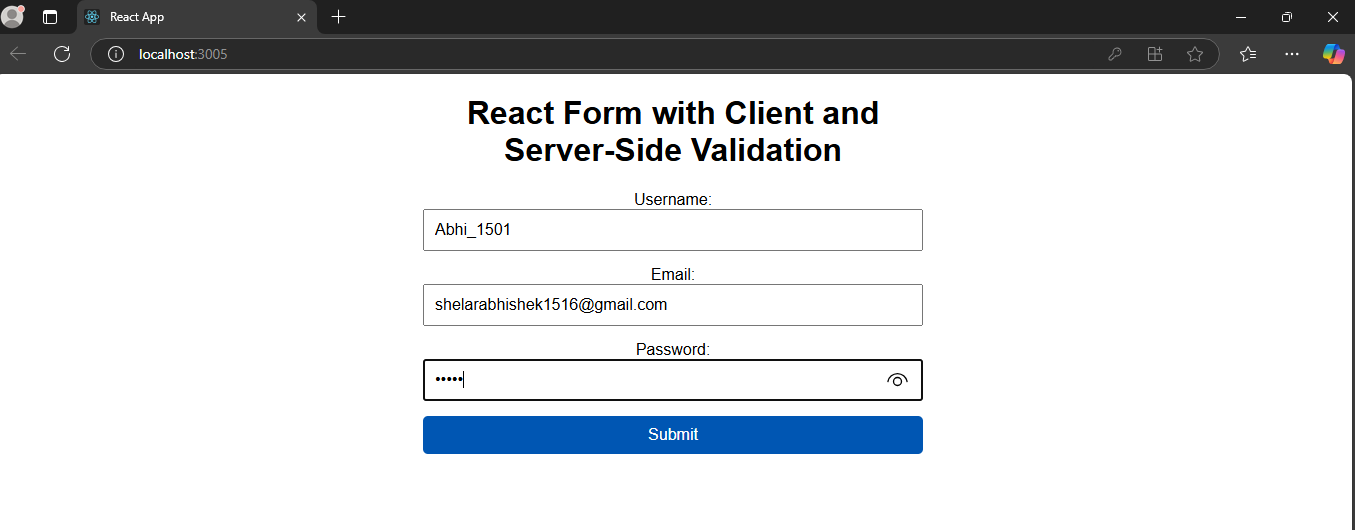
</div>

);

}

export default App;

**OUTPUT:**

****

# PRACTICAL NO.46

## Create an application in ReactJS to show name change and increment and decrement the count using React Hooks.

### App.js:

import React, { useState, useEffect, useRef } from 'react'; import './App.css';

function App() {

// 1. useState Hook

const [count, setCount] = useState(0); const [name, setName] = useState('');

// 2. useRef Hook

const inputRef = useRef(null);

// 3. useEffect Hook (runs on component mount and updates the document title) useEffect(() => {

// This effect runs when the component mounts or when 'count' changes document.title = `Count: ${count}`;

}, [count]); // Dependency array

// Handle incrementing the count const handleIncrement = () => { setCount(count + 1);

};

// Handle decrementing the count const handleDecrement = () => { setCount(count - 1);

};

// Handle name input change

const handleNameChange = (e) => { setName(e.target.value);

};

// Focus the input field when the "Focus Input" button is clicked const focusInput = () => { inputRef.current.focus();

};

return (

<div className="App">

<h1>React Hooks Example</h1>

{/\* 4. Counter Section using useState \*/}

<div>

<h2>Counter</h2>

<p>Current Count: {count}</p>

<button onClick={handleIncrement}>Increment</button>

<button onClick={handleDecrement}>Decrement</button>

</div>

{/\* 5. Name Input Section using useState \*/}

<div>

<h2>Name Input</h2>

<input type="text" value={name}

onChange={handleNameChange} placeholder="Enter your name"

/>

<p>Your name is: {name}</p>

</div>

{/\* 6. Focus Input Section using useRef \*/}

<div>

<h2>Focus Input Field</h2>

<input type="text" ref={inputRef}

placeholder="This input can be focused"

/>

<button onClick={focusInput}>Focus Input</button>

</div>

</div>

);

}

export default App;

### App.css:

.App {

text-align: center;

font-family: Arial, sans-serif; max-width: 500px;

margin: 0 auto;

}

button {

padding: 10px 20px; margin: 10px;

font-size: 16px; cursor: pointer;

border: 2px solid #007bff; background-color: #007bff; color: white;

border-radius: 5px;

}

button:hover {

background-color: #0056b3;

}

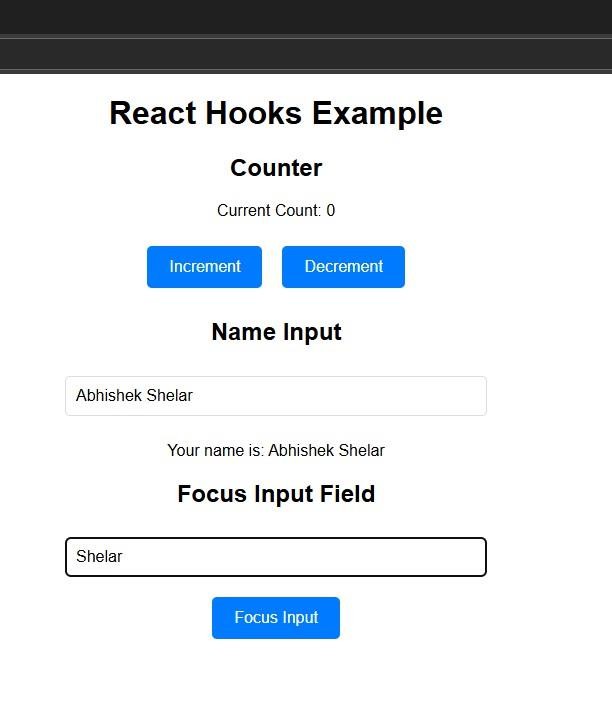
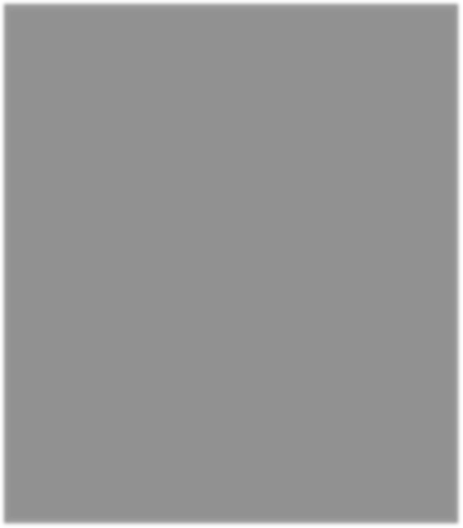
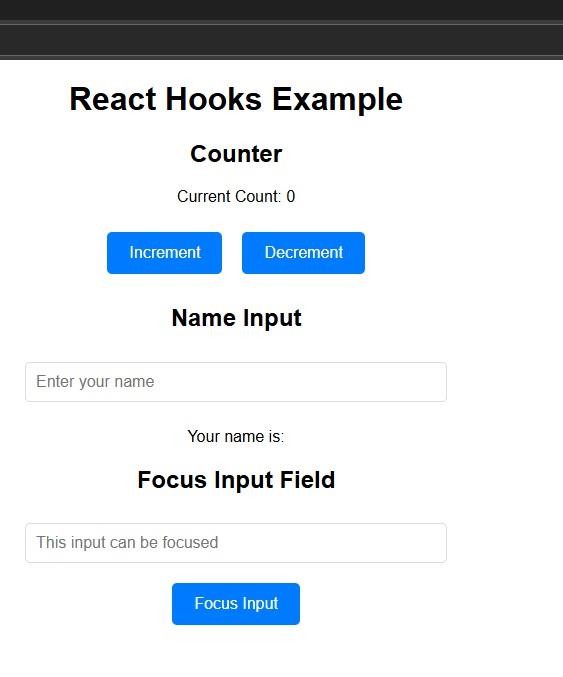
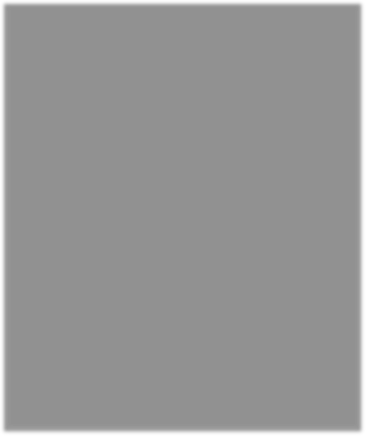
input {

padding: 10px; font-size: 16px; margin: 10px; width: 80%;

border: 1px solid #ddd; border-radius: 5px;

}

**OUTPUT:**



# PRACTICAL NO.47

## Create SPA (Single Page Application) using React Router involved setting up a basic React Project and adding Navigation between different views.

### App.js:

import React from 'react';

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

import About from './About'; import Contact from './Contact';

function App() { return (

<Router>

<div>

<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>

<Route path="/" element={<Home />} />

<Route path="/about" element={<About />} />

<Route path="/contact" element={<Contact />} />

</Routes>

</div>

</Router>

);

}

export default App;

### Home.js:

import React from 'react'; function Home() {

return (

<div>

<h2>Welcome to the Home Page</h2>

<p>This is the homepage of your SPA.</p>

</div>

);

}

export default Home;

### About.js:

import React from 'react'; function About() {

return (

<div>

<h2>About Us</h2>

<p>This page tells you more about the application.</p>

</div>

);

}

export default About;

### Contact.js:

import React from 'react'; function Contact() { return (

<div>

<h2>Contact Us</h2>

<p>You can contact us here.</p>

</div>

);

}

export default Contact;

**OUTPUT:**

