

Sarhad College of Arts Commerece & Science Katraj Pune-46

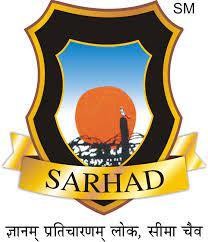
NAAC Accrediated with B+ Grade

**(FACULTY OF SCIENCE & TECHNOLOGY)**

**Subject:**

**MSc CS Sem-**

Name of the Student -……………………………………….. Roll No-……………… Exam Seat No……………………….



**SARHAD COLLEGE OF ARTS, COMMERCE AND SCIENCE KATRAJ, PUNE-46**

**(FACULTY OF SCIENCE & TECHNOLOGY)**

**Seat No :-**

CERTIFICATE

**This is informed that Miss /Mr/Mrs.**

**Roll No.**  **From**  **class has satisfactorily completed the laboratory course in**  **subject during the year**  **as per the requirement of the university of Pune.**

**Date:-**

**Teacher In-charge HOD Principal Internal Examiner External Examiner**

***INDEX***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sr**  **No** | **Title** | **DATE** | **REMARKS** | **SIGNATURE** |
| 1 | Create an HTML form that contains the Student Registration details and write a JavaScript to validate Student first and last name as it should not  contain other than alphabets and age should be between 18 to 50. |  |  |  |
| 2 | Create an HTML form that contains the Employee Registration details and  write a JavaScript to validate DOB, Joining Date, and Salary. |  |  |  |
| 3 | Create an HTML form for Login and write a JavaScript to validate email ID  using Regular Expression. |  |  |  |
| 4 | Write AngularJS by using ng-click Directive to display an alert message after  clicking the element. |  |  |  |
| 5 | Write an AngularJS script for the addition of two numbers using ng-init, ng-  model & ng-bind. And also demonstrate ng-show, ng-disabled, ng-click directives on the button component |  |  |  |
| 6 | Using AngularJS, display the 10 student details in Table format (using ng- repeat directive, use Array to store data). |  |  |  |
| 7 | Using AngularJS, create a SPA that shows Syllabus content of all subjects of MSc(CS) Sem II (use ng-view). |  |  |  |
| 8 | Using AngularJS, create a SPA to accept details such as name, mobile number, pincode, and email address and make validation. Name should contain characters only, mobile number should contain only 10 digits,  Pincode should contain only 6 digits, email ID should contain only one @ and  . symbol. |  |  |  |
| 9 | Using AngularJS, create a SPA for the Login System. |  |  |  |
| 10 | Create an HTML form using AngularJS that contains the Student Registration details and validates Student first and last name as it should not contain other than alphabets and age should be between 18 to 50, and display a  greeting message depending on the current time using ng-show (e.g., Good Morning, Good Afternoon, etc.) (Use AJAX). |  |  |  |
| 11 | Create an AngularJS Application that shows the current Date and Time of the System (Use Interval Service). |  |  |  |
| 12 | Using AngularJS, create a SPA to carry out validation for a username entered in a textbox. If the textbox is blank, alert ‘Enter username’. If the number of characters is less than three, alert ‘Username is too short’. If the value entered is appropriate, print ‘Valid username’, and password should be a  minimum of 8 characters. |  |  |  |
| 13 | Create an AngularJS Application that shows the location of the current web  page. |  |  |  |
| 14 | Create a Node.js file that will convert the output "Hello World!" into  uppercase letters. |  |  |  |
| 15 | Using Node.js, create a web page to read two file names from the user and  append contents of the first file into the second file |  |  |  |
| 16 | Create a Node.js file that opens the requested file and returns the content to  the client. If anything goes wrong, throw a 404 error. |  |  |  |
| 17 | Create a Node.js file that writes an HTML form with an upload field. |  |  |  |
| 18 | Create a Node.js file that demonstrates create database and table in MySQL |  |  |  |
| 19 | Create a Node.js file that selects all records from the "customers" table and  displays the result object on the console. |  |  |  |
| 20 | Create a Node.js file that inserts multiple records in the "student" table and  displays the result object on the console |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 21 | Create a Node.js file that selects all records from the "customers" table and  deletes the specified record. |  |  |  |
| 22 | Create a Simple Web Server using Node.js. |  |  |  |
| 23 | Using Node.js, create a User Login System |  |  |  |
| 24 | Using Node.js, create an eLearning System. |  |  |  |
| 25 | Using Node.js, create a Recipe Book |  |  |  |
| 26 | Write a Node.js script to interact with the file system and serve a web page from a file. |  |  |  |
| 27 | Write a Node.js script to build Your Own Node.js Module. Use require (‘http’) module, which is a built-in Node module that invokes the functionality of the HTTP library to create a local server. Also, use the export statement to make functions in your module available externally. Create a new text file to contain the functions in your module called “modules.js” and add this  function to return today’s date and time. |  |  |  |
| 28 | Create a JS file named main.js for an event-driven application. There should be a main loop that listens for events and then triggers a callback function when one of those events is detected. |  |  |  |
| 29 | Write a Node.js application that transfers a file as an attachment on the web and enables the browser to prompt the user to download the file using  Express.js. |  |  |  |
| 30 | Case Studies on MEAN Stack Application Development. |  |  |  |

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

**Sol:**<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Student Registration</title>

<script>

function validateForm() {

var firstName = document.getElementById("firstName").value; var lastName = document.getElementById("lastName").value; var age = document.getElementById("age").value;

var namePattern = /^[A-Za-z]+$/;

if (!namePattern.test(firstName) || !namePattern.test(lastName)) { alert("First and Last Name should contain only alphabets."); return false;

}

if (age < 18 || age > 50) {

alert("Age should be between 18 and 50."); return false;

}

return true;

}

</script>

</head>

<body>

<h2>Student Registration Form</h2>

<form onsubmit="return validateForm()">

First Name: <input type="text" id="firstName" name="firstName" required><br><br> Last Name: <input type="text" id="lastName" name="lastName" required><br><br> Age: <input type="number" id="age" name="age" required><br><br>

<input type="submit" value="Register">

</form>

</body>

</html>

### OUTPUT:

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

## Sol:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Employee Registration</title>

<script>

function validateForm() {

var dob = document.getElementById("dob").value;

var joiningDate = document.getElementById("joiningDate").value; var salary = document.getElementById("salary").value;

if (new Date(dob) > new Date()) {

alert("DOB cannot be in the future."); return false;

}

if (new Date(joiningDate) > new Date()) {

alert("Joining Date cannot be in the future."); return false;

}

if (salary <= 0) {

alert("Salary should be greater than 0."); return false;

}

return true;

}

</script>

</head>

<body>

<h2>Employee Registration Form</h2>

<form onsubmit="return validateForm()">

Date of Birth: <input type="date" id="dob" name="dob" required><br><br> Joining Date: <input type="date" id="joiningDate" name="joiningDate"

required><br><br>

Salary: <input type="number" id="salary" name="salary" required><br><br>

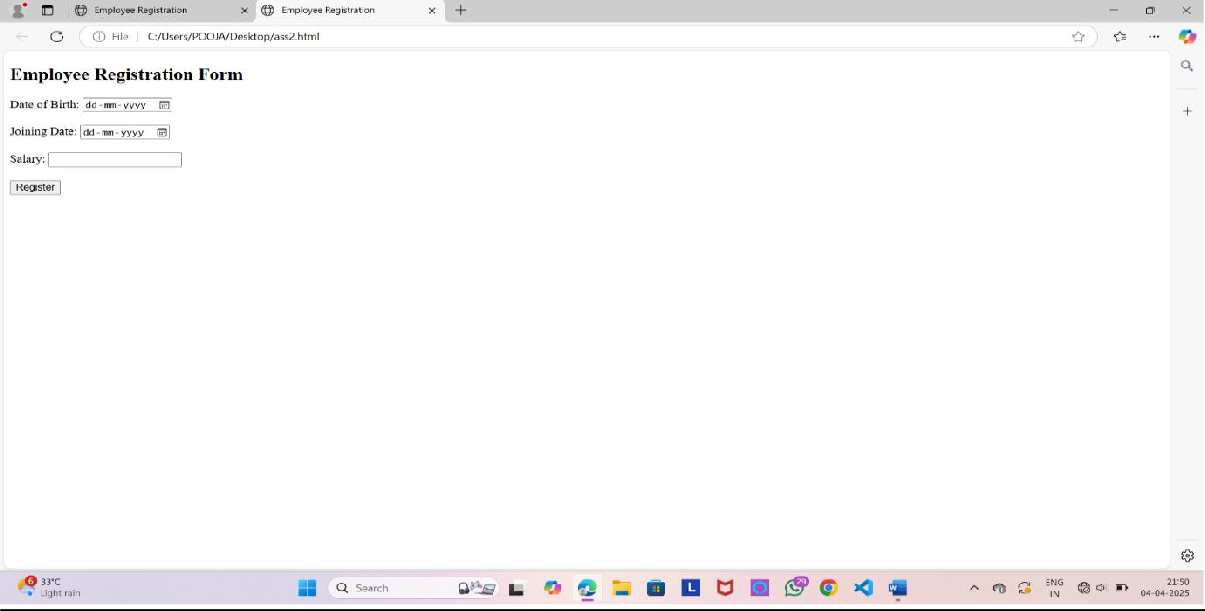
<input type="submit" value="Register">

</form>

</body>

</html>

# OUTPUT:

****

**Program 3:** Create an HTML form for Login and write a JavaScript to validate email ID using Regular Expression.

**Sol:** <!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Login</title>

<script>

function validateEmail() {

var email = document.getElementById("email").value;

var emailPattern = /^[a-zA-Z0-9.\_%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$/;

if (!emailPattern.test(email)) {

alert("Please enter a valid email address."); return false;

}

return true;

}

</script>

</head>

<body>

<h2>Login Form</h2>

<form onsubmit="return validateEmail()">

Email: <input type="email" id="email" name="email" required><br><br>

Password: <input type="password" id="password" name="password" required><br><br>

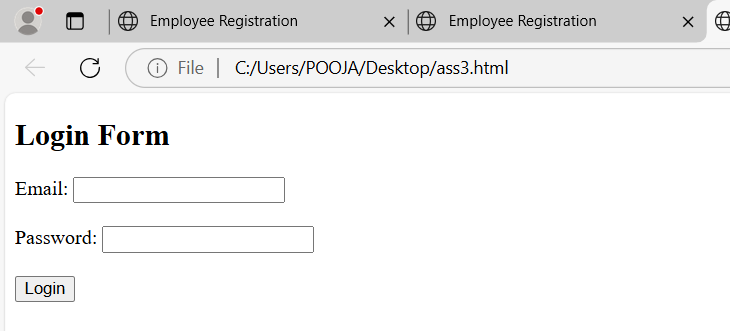
<input type="submit" value="Login">

</form>

</body>

</html>

##### OUTPUT:



**Program 4:** Write AngularJS by using ng-click Directive to display an alert message after clicking the element

## Sol:

<!DOCTYPE html>

<html lang="en" ng-app="myApp">

<head>

<meta charset="UTF-8">

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

<title>AngularJS ng-click</title>

<script

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

<script>

var app = angular.module('myApp', []); app.controller('myCtrl', function($scope) {

$scope.showAlert = function() { alert("You clicked the button!");

}; });

</script>

</head>

<body ng-controller="myCtrl">

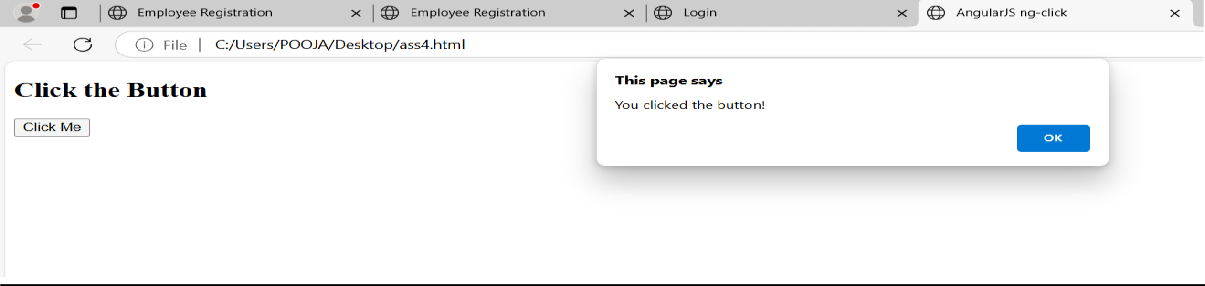
<h2>Click the Button</h2>

<button ng-click="showAlert()">Click Me</button>

</body>

</html>

# OUTPUT:

****

**Program 5:** Write an AngularJS script for the addition of two numbers using ng-init, ng-model & ng- bind. And also demonstrate ng-show, ng-disabled, ng-click directives on the button component

#### Sol:

<!DOCTYPE html>

<html lang="en" ng-app="myApp">

<head>

<meta charset="UTF-8">

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

<title>AngularJS Addition</title>

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

<script>

var app = angular.module('myApp', []); app.controller('myCtrl', function($scope) {

$scope.num1 = 0;

$scope.num2 = 0;

$scope.result = 0;

});

</script>

</head>

<body ng-controller="myCtrl">

<h2>Addition of Two Numbers</h2>

<input type="number" ng-model="num1" ng-init="num1=0" placeholder="Enter first number">

<input type="number" ng-model="num2" ng-init="num2=0" placeholder="Enter second number">

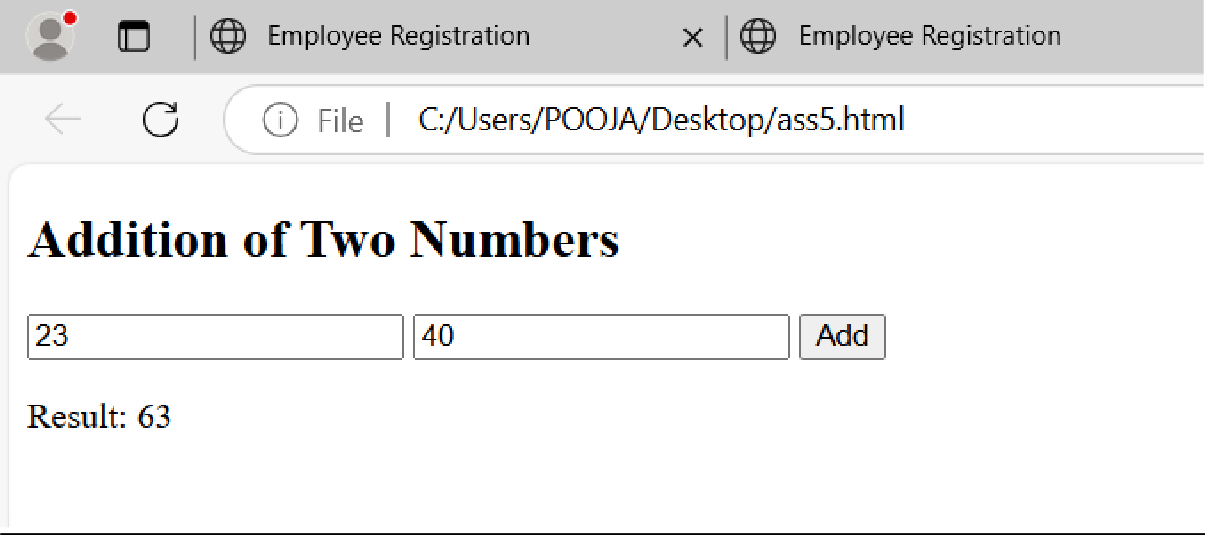
<button ng-click="result = num1 + num2" ng-disabled="num1 === 0 || num2 === 0">Add</button>

<p>Result: <span ng-bind="result"></span></p>

</body>

</html>

##### OUTPUT:

******

**Program 6:** Using AngularJS, display the 10 student details in Table format (using ng-repeat directive, use Array to store data).

## Sol:

<!DOCTYPE html>

<html lang="en" ng-app="myApp">

<head>

<meta charset="UTF-8">

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

<title>Student Details</title>

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

<script>

var app = angular.module('myApp', []); app.controller('myCtrl', function($scope) {

$scope.students = [

{name: "John", age: 20, grade: "A"},

{name: "Sarah", age: 21, grade: "B"},

{name: "Mike", age: 22, grade: "C"},

{name: "Emma", age: 23, grade: "A"},

{name: "Sophia", age: 24, grade: "B"},

{name: "James", age: 25, grade: "A"},

{name: "David", age: 26, grade: "C"},

{name: "Lucas", age: 27, grade: "B"},

{name: "Isabella", age: 28, grade: "A"},

{name: "Mia", age: 29, grade: "C"}

];

});

</script>

</head>

<body ng-controller="myCtrl">

<h2>Student Details</h2>

<table border="1">

<thead>

<tr>

<th>Name</th>

<th>Age</th>

<th>Grade</th>

</tr>

</thead>

<tbody>

<tr ng-repeat="student in students">

<td>{{student.name}}</td>

<td>{{student.age}}</td>

<td>{{student.grade}}</td>

</tr>

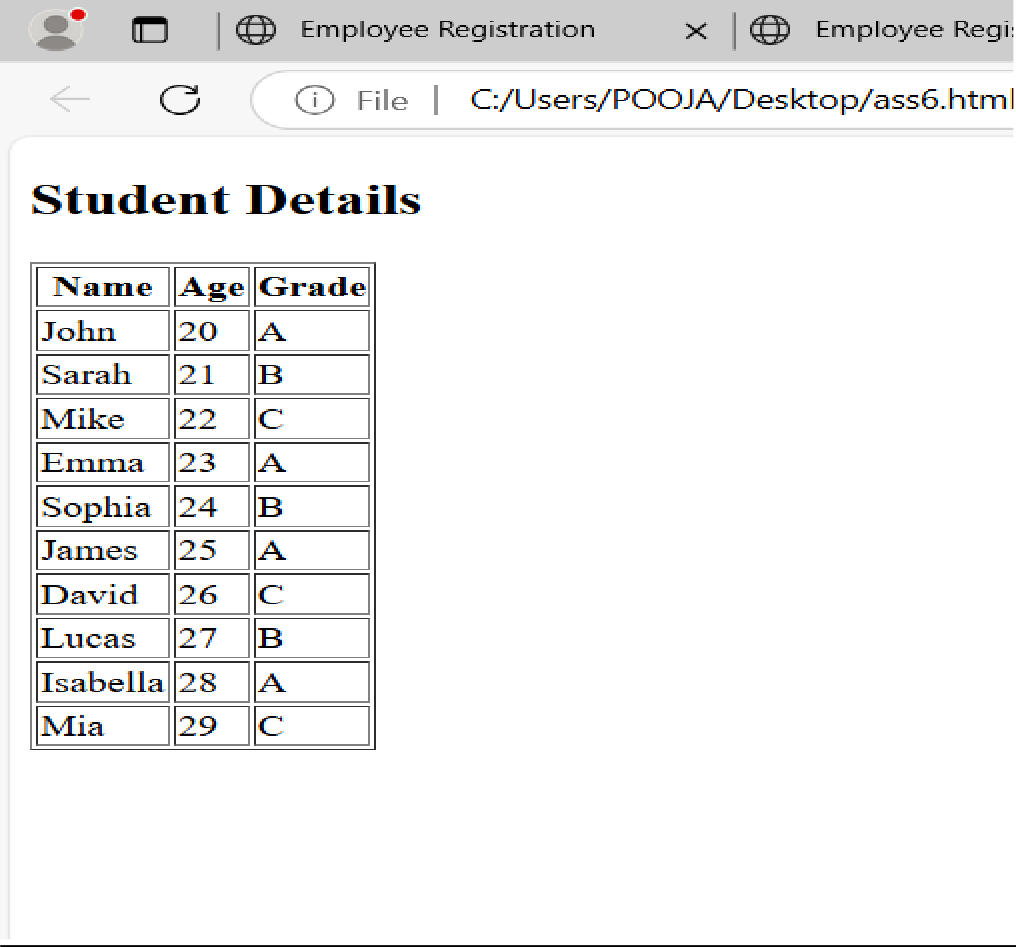
</tbody>

</table>

</body>

</html>

### OUTPUT:

******

**Program 7:** Using AngularJS, create a SPA that shows Syllabus content of all subjects of MSc(CS) Sem II (use ng-view).

**Sol:**

<!DOCTYPE html>

<html lang="en" ng-app="myApp">

<head>

<meta charset="UTF-8">

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

<title>Syllabus for MSC(CS) Sem II</title>

<script

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

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular- route.js"></script>

<script>

var app = angular.module('myApp', ['ngRoute']);

app.config(function($routeProvider) {

$routeProvid

.when("/syllabus", {

template: "<h2>Syllabus for MSC(CS) Sem II</h2><ul><li>Mathematical Foundations</li><li>Operating Systems</li><li>Advanced Algorithms</li><li>Database Systems</li></ul>"

})

.otherwise({

redirectTo: '/syllabus'

});

});

</script>

</head>

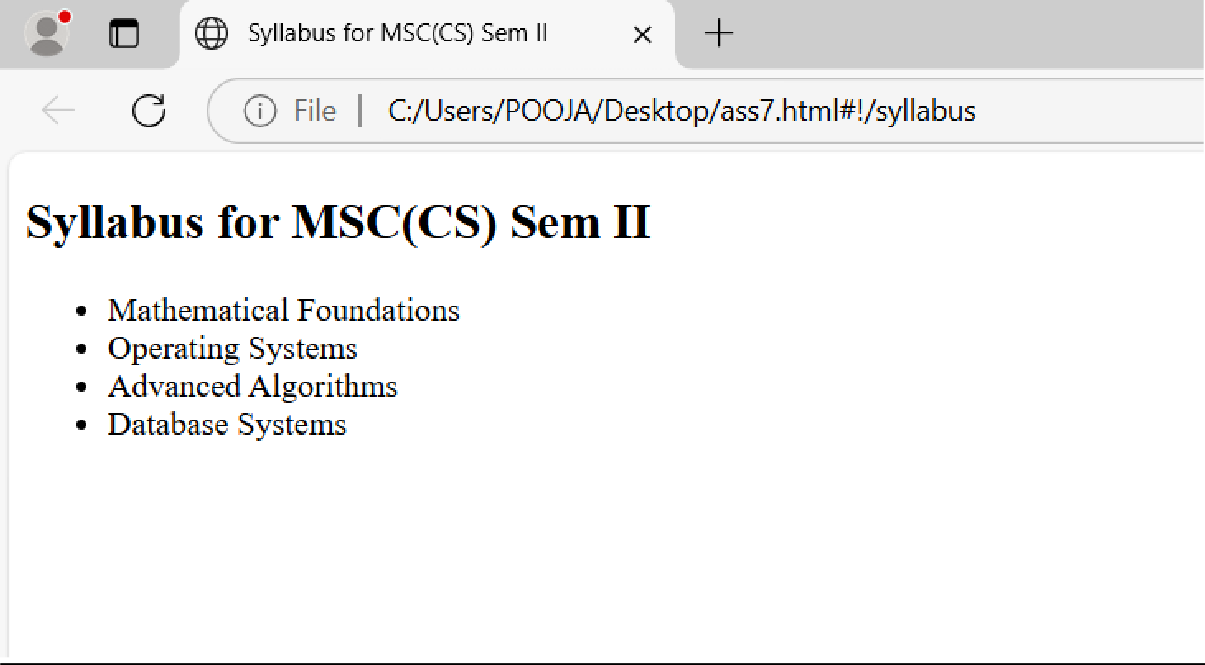
<body>

<div ng-view></div>

</body>

</html>

### OUTPUT:



**Program 8:** Using AngularJS, create a SPA to accept details such as name, mobile number, pincode, and email address and make validation. Name should contain characters only, mobile number should contain only 10 digits, Pincode should contain only 6 digits, email ID should contain only one @ and . symbol.

## Sol:

<!DOCTYPE html>

<html lang="en" ng-app="myApp">

<head>

<meta charset="UTF-8">

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

<title>User Details Form</title>

<script

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

<script>

var app = angular.module('myApp', []); app.controller('myCtrl', function($scope) {

$scope.validateForm = function() {

if (!/^[A-Za-z ]+$/.test($scope.name)) {

alert("Name should contain only characters."); return false;

}

if (!/^\d{10}$/.test($scope.mobile)) {

alert("Mobile number should be 10 digits."); return false;

}

if (!/^\d{6}$/.test($scope.pincode)) { alert("Pincode should be 6 digits."); return false;

}

if (!/\S+@\S+\.\S+/.test($scope.email)) { alert("Please enter a valid email address."); return false;

}

// If everything is valid

alert("Form submitted successfully!"); return true;

};

});

</script>

</head>

<body ng-controller="myCtrl">

<h2>User Details Form</h2>

<form ng-submit="validateForm()">

Name: <input type="text" ng-model="name" required><br><br> Mobile: <input type="text" ng-model="mobile" required><br><br>

Pincode: <input type="text" ng-model="pincode" required><br><br>

Email: <input type="email" ng-model="email" required><br><br>

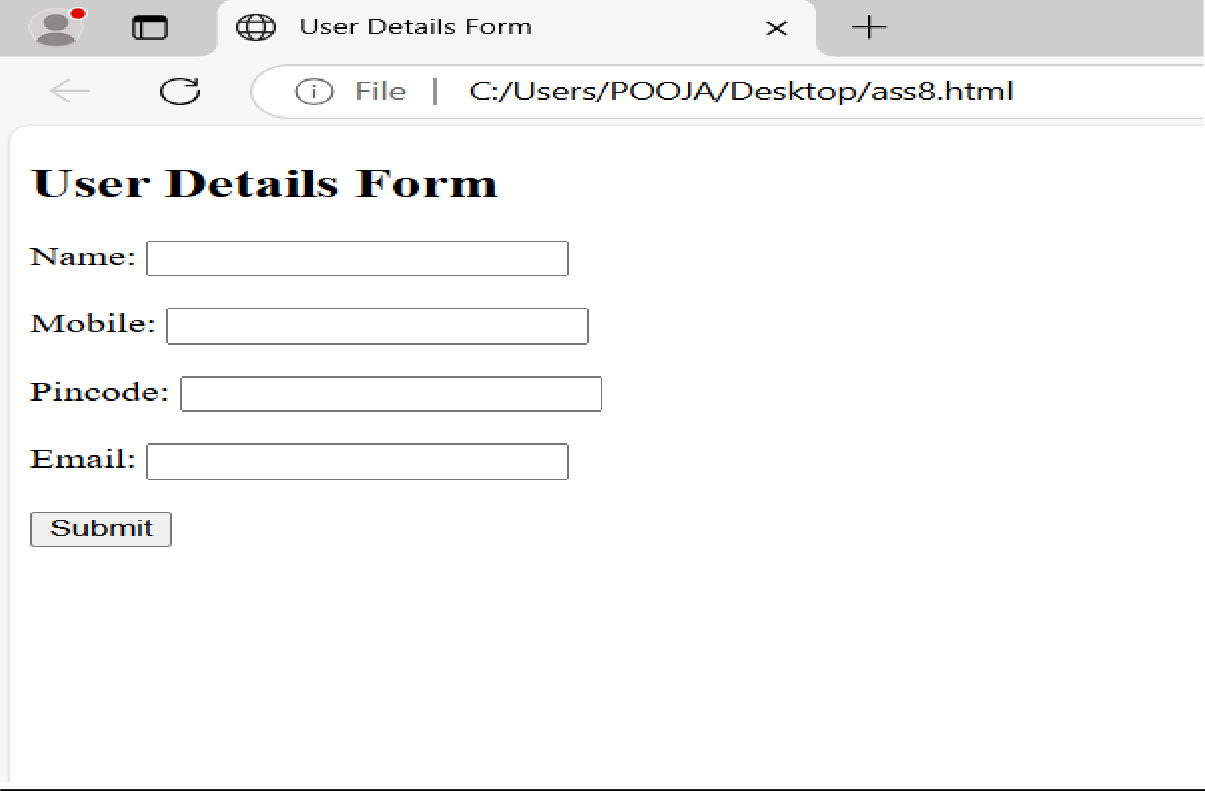
<input type="submit" value="Submit">

</form>

</body>

</html>

# OUTPUT:

****

**Program 9:** Using AngularJS, create a SPA for the Login System.

## Sol:

<!DOCTYPE html>

<html lang="en" ng-app="myApp">

<head>

<meta charset="UTF-8">

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

<title>Login System</title>

<script

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

<script>

var app = angular.module('myApp', []); app.controller('loginCtrl', function($scope) {

$scope.login = function() {

if ($scope.username === "admin" && $scope.password === "password123") {

alert("Login Successful");

} else {

alert("Invalid credentials");

}

};

});

</script>

</head>

<body ng-controller="loginCtrl">

<h2>Login</h2>

<form ng-submit="login()">

Username: <input type="text" ng-model="username" required><br><br> Password: <input type="password" ng-model="password" required><br><br>

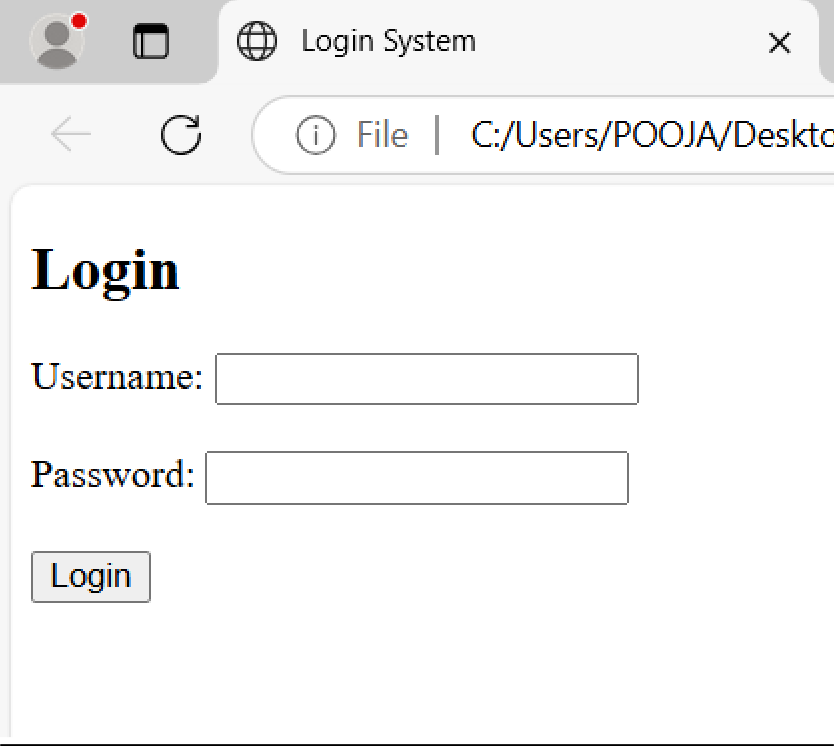
<input type="submit" value="Login">

</form>

</body>

</html>

# OUTPUT:

****

**Program 10:** Create an HTML form using AngularJS that contains the Student Registration details and validates Student first and last name as it should not contain other than alphabets and age should be between 18 to 50, and display a greeting message depending on the current time using ng- show (e.g., Good Morning, Good Afternoon, etc.) (Use AJAX).

## Sol:

<!DOCTYPE html>

<html lang="en" ng-app="myApp">

<head>

<meta charset="UTF-8">

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

<title>Student Registration</title>

<script

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

<script>

var app = angular.module('myApp', []); app.controller('registrationCtrl', function($scope) {

$scope.validateForm = function() {

if (!/^[A-Za-z]+$/.test($scope.firstName) || !/^[A-Za-z]+$/.test($scope.lastName)) { alert("First and Last Name should contain only alphabets.");

return false; // Prevent form submission

}

if ($scope.age < 18 || $scope.age > 50) {

alert("Age should be between 18 and 50."); return false; // Prevent form submission

}

alert("Registration Successful!"); // Show success alert return true; // Allow form submission

};

var currentTime = new Date().getHours(); if (currentTime < 12) {

$scope.greeting = "Good Morning";

} else if (currentTime < 18) {

$scope.greeting = "Good Afternoon";

} else {

$scope.greeting = "Good Evening";

}

});

</script>

</head>

<body ng-controller="registrationCtrl">

<h2>Student Registration</h2>

<form ng-submit="validateForm()">

First Name: <input type="text" ng-model="firstName" required><br><br> Last Name: <input type="text" ng-model="lastName" required><br><br> Age: <input type="number" ng-model="age" required><br><br>

<input type="submit" value="Register">

</form>

<div ng-show="greeting">

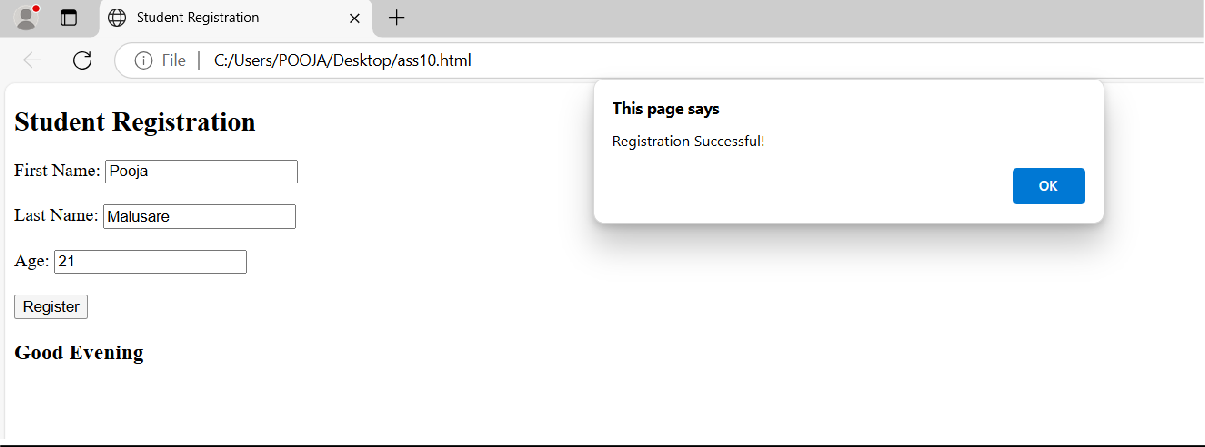
<h3>{{greeting}}</h3>

</div>

</body>

</html>

# OUTPUT:

****

**Program 11:** Create an AngularJS Application that shows the current Date and Time of the System (Use Interval Service).

## Sol:

<!DOCTYPE html>

<html lang="en" ng-app="myApp">

<head>

<meta charset="UTF-8">

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

<title>Current Date and Time</title>

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

<script>

var app = angular.module('myApp', []); app.controller('timeCtrl', function($scope, $interval) {

$scope.currentTime = new Date();

$interval(function() {

$scope.currentTime = new Date();

}, 1000); // Update every second

});

</script>

</head>

<body ng-controller="timeCtrl">

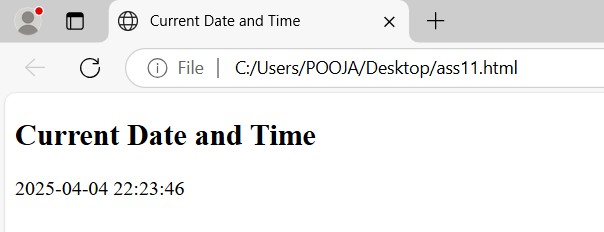
<h2>Current Date and Time</h2>

<p>{{currentTime | date: 'yyyy-MM-dd HH:mm:ss'}}</p>

</body>

</html>

### OUTPUT:

******

**Program 12:** Using AngularJS, create a SPA to carry out validation for a username entered in a textbox. If the textbox is blank, alert ‘Enter username’. If the number of characters is less than three, alert ‘Username is too short’. If the value entered is appropriate, print ‘Valid username’, and password should be a minimum of 8 characters

## Sol:

<!DOCTYPE html>

<html lang="en" ng-app="myApp">

<head>

<meta charset="UTF-8">

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

<title>Username and Password Validation</title>

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

<script>

var app = angular.module('myApp', []); app.controller('validationCtrl', function($scope) {

$scope.validate = function() { if (!$scope.username) {

alert("Enter username");

} else if ($scope.username.length < 3) { alert("Username is too short");

} else {

alert("Valid username");

}

if ($scope.password && $scope.password.length < 8) { alert("Password should be minimum 8 characters");

}

};

});

</script>

</head>

<body ng-controller="validationCtrl">

<h2>Username and Password Validation</h2>

<form ng-submit="validate()">

Username: <input type="text" ng-model="username" required><br><br> Password: <input type="password" ng-model="password" required><br><br>

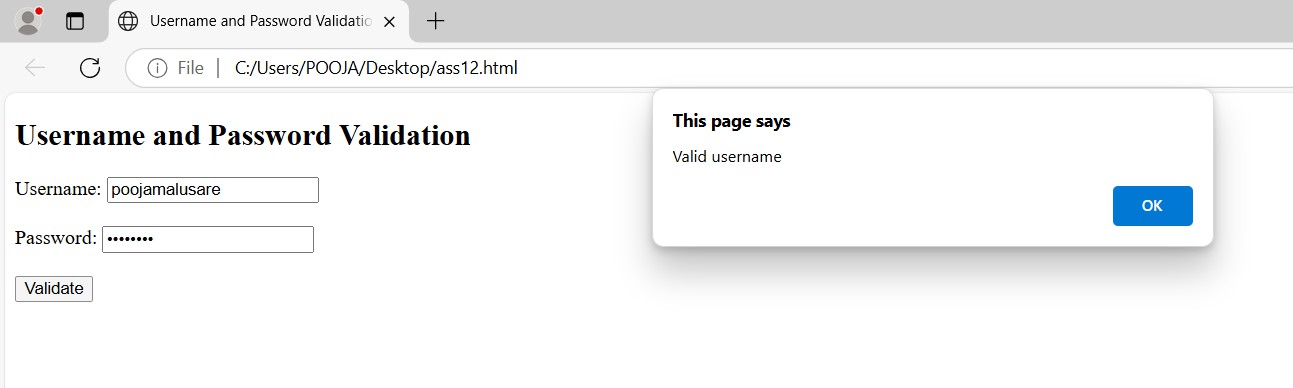
<input type="submit" value="Validate">

</form>

</body>

</html>

##### OUTPUT:

******

**Program 13:** Create an AngularJS Application that shows the location of the current web page.

## Sol:

<!DOCTYPE html>

<html lang="en" ng-app="myApp">

<head>

<meta charset="UTF-8">

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

<title>Current Web Page Location</title>

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

<script>

var app = angular.module('myApp', []); app.controller('locationCtrl', function($scope) {

$scope.location = window.location.href; // Gets the current page URL

});

</script>

</head>

<body ng-controller="locationCtrl">

<h2>Current Web Page Location</h2>

<p>{{location}}</p>

</body>

</html>

##### OUTPUT:

**Program 14:** Create a Node.js file that will convert the output "Hello World!" into uppercase letters.

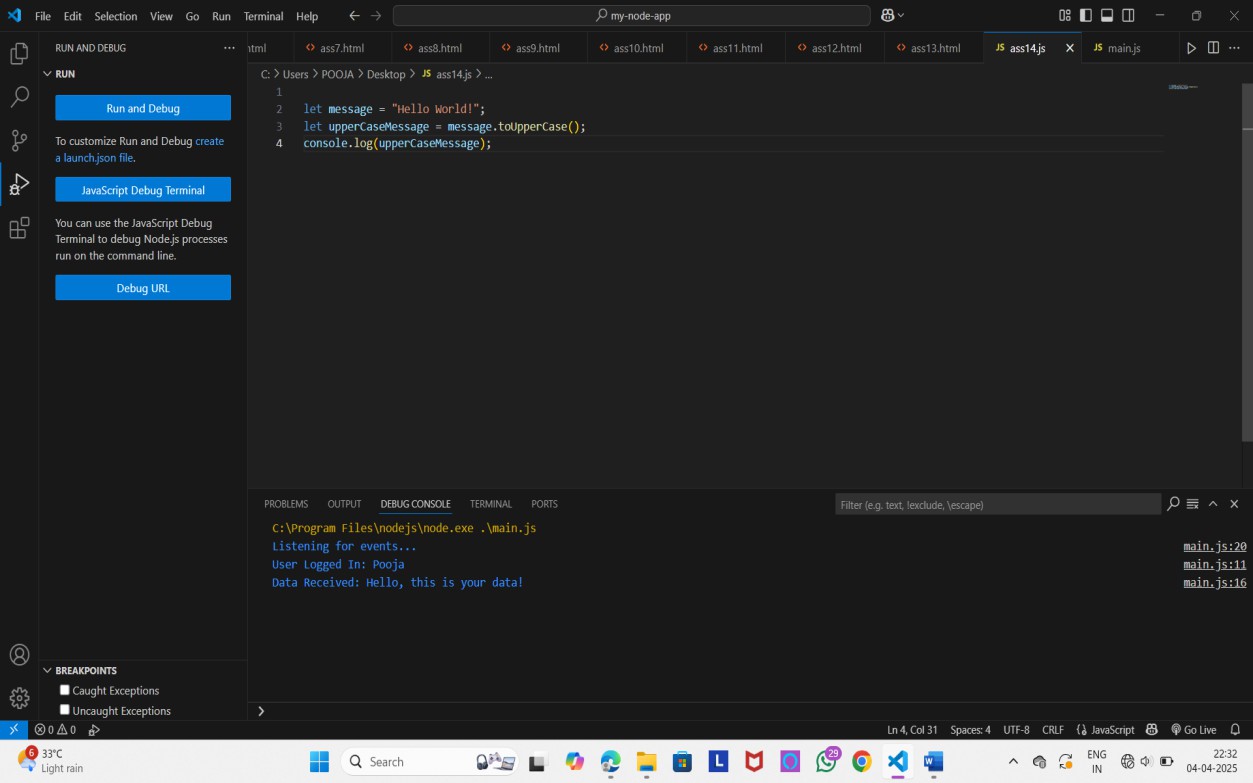
## Sol:

## const message = "Hello World!";

## const upperCaseMessage = message.toUpperCase();

## console.log(upperCaseMessage);

## output:

****

**Program 15:** Using Node.js, create a web page to read two file names from the user and append contents of the first file into the second file

**Sol:** index.html

<!DOCTYPE html>

<html>

<head>

<title>Append File Contents</title>

</head>

<body>

<h1>Append File Contents</h1>

<form action="/append" method="POST">

<label>Source File Name:</label>

<input type="text" name="sourceFile" required><br><br>

<label>Destination File Name:</label>

<input type="text" name="destFile" required><br><br>

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

</form>

</body>

</html>

## app.js

const express = require('express'); const fs = require('fs');

const path = require('path');

const bodyParser = require('body-parser');

const app = express(); const PORT = 3000;

app.use(bodyParser.urlencoded({ extended: true })); app.use(express.static('public'));

app.post('/append', (req, res) => {

const sourceFile = req.body.sourceFile; const destFile = req.body.destFile;

// Read content from source file fs.readFile(sourceFile, 'utf8', (err, data) => { if (err) {

return res.send(`Error reading source file: ${err.message}`);

}

// Append content to destination file fs.appendFile(destFile, data, (err) => { if (err) {

return res.send(`Error writing to destination file: ${err.message}`);

}

res.send(`Successfully appended content of '${sourceFile}' to '${destFile}'`);

});

});

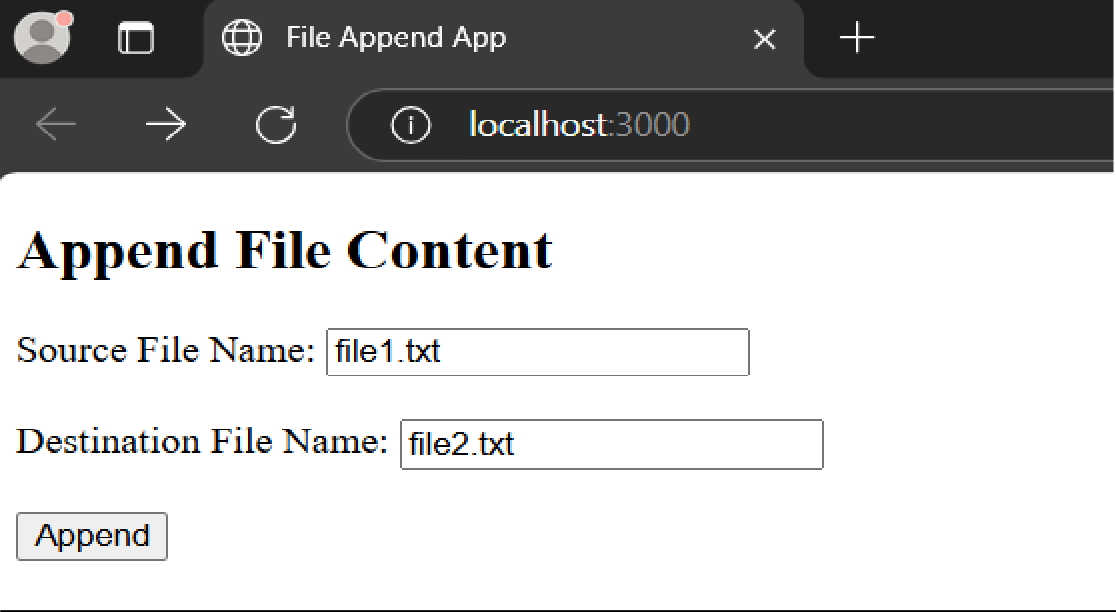
});

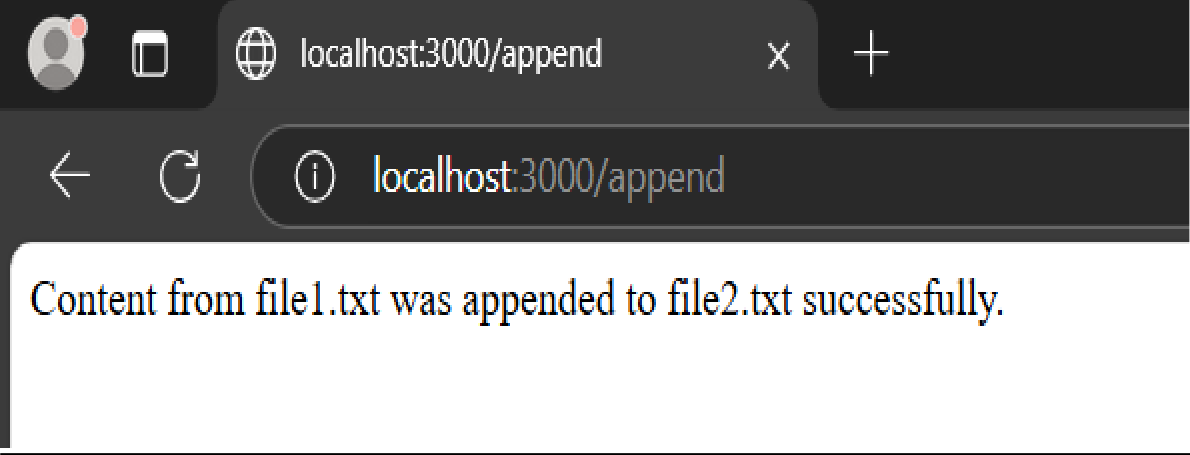
app.listen(PORT, () => {

console.log(`Server running at [http://localhost:$](http://localhost/){PORT}`);

});

### OUTPUT:



******

**Program 16:** Create a Node.js file that opens the requested file and returns the content to the client. If anything goes wrong, throw a 404 error.

**Sol:** make file named server.js const http = require('http'); const fs = require('fs');

const path = require('path');

const server = http.createServer((req, res) => {

const fileName = decodeURI(req.url).slice(1); // Remove leading "/" if (!fileName) {

res.writeHead(400, { 'Content-Type': 'text/plain' });

return res.end('Please specify a file name in the URL, e.g., /file.txt');

}

const filePath = path.join( dirname, fileName);

// Read the requested file fs.readFile(filePath, 'utf8', (err, data) => {

if (err) {

res.writeHead(404, { 'Content-Type': 'text/plain' }); return res.end('404 - File not found or cannot be read');

}

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

});});

const PORT = 3000;

server.listen(PORT, () => {

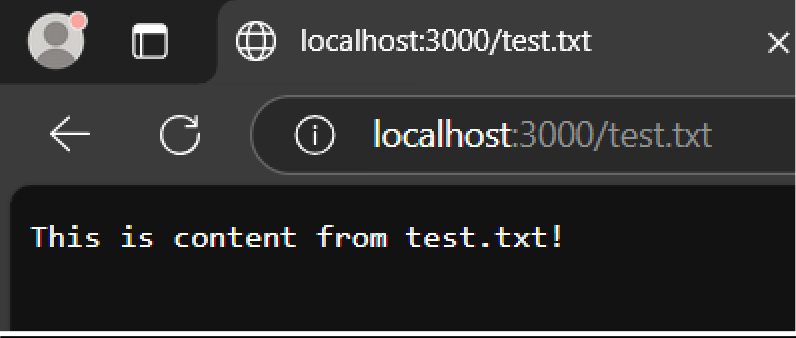
console.log(`Server running at [http://localhost:$](http://localhost/){PORT}`);

});

## Now make test.txt file:

This is content from test.txt!

### OUTPUT:



**Program 17:** Create a Node.js file that writes an HTML form with an upload field.

**Sol:** Create server.js

const express = require('express'); const multer = require('multer'); const path = require('path');

const app = express(); const PORT = 3000;

const storage = multer.diskStorage({ destination: 'uploads/',

filename: (req, file, cb) => {

cb(null, file.fieldname + '-' + Date.now() + path.extname(file.originalname));

}});

const upload = multer({ storage: storage }); app.get('/', (req, res) => {

res.send(`

<h2>Upload a File</h2>

<form action="/upload" method="POST" enctype="multipart/form-data">

<input type="file" name="file" required>

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

</form>

`);});

app.post('/upload', upload.single('file'), (req, res) => { if (!req.file) {

return res.send('No file uploaded.');

}

res.send(`File uploaded successfully: ${req.file.filename}`);

});

app.listen(PORT, () => {

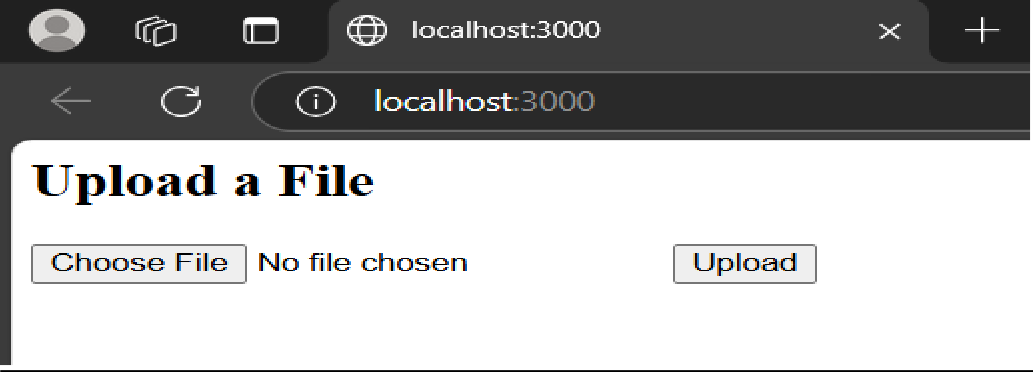
console.log(`Server running at [http://localhost:$](http://localhost/){PORT}`);

});

### Run the Server:

node server.js

### OUTPUT:

******

**Program 18:** Create a Node.js file that demonstrates create database and table in MySQL const mysql = require('mysql');

// Create connection to MySQL server (no database yet) const connection = mysql.createConnection({

host: 'localhost', user: 'root',

password: 'your\_password' // replace with your actual password

});

// Connect to MySQL server connection.connect(err => { if (err) throw err;

console.log('Connected to MySQL server.');

// Step 1: Create Database

connection.query('CREATE DATABASE IF NOT EXISTS sample\_db', (err, result) => { if (err) throw err;

console.log(' Database "sample\_db" created or already exists.');

// Switch to new database

connection.changeUser({ database: 'sample\_db' }, err => { if (err) throw err;

// Step 2: Create Table const createTableQuery = `

CREATE TABLE IF NOT EXISTS users (

id INT AUTO\_INCREMENT PRIMARY KEY, name VARCHAR(100) NOT NULL,

email VARCHAR(100) UNIQUE NOT NULL,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

)

`;

connection.query(createTableQuery, (err, result) => { if (err) throw err;

console.log(' Table "users" created or already exists.');

// Close connection connection.end();

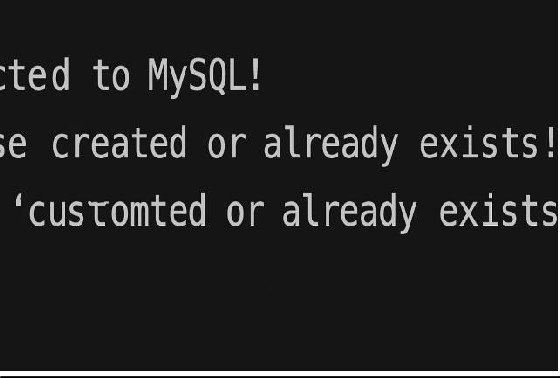
});

});

});

});

## Sol:

****

**Program 19:** Create a Node.js file that selects all records from the "customers" table and displays the result object on the console.

**Sol:** (Save this as selectCustomers.js) const mysql = require('mysql');

// Create a connection to the database

const connection = mysql.createConnection({ host: 'localhost',

user: 'root',

password: '', // default password for XAMPP

database: 'college' // change if your "customers" table is in a different DB

});

// Connect to the database connection.connect((err) => { if (err) throw err;

console.log('Connected to the database.');

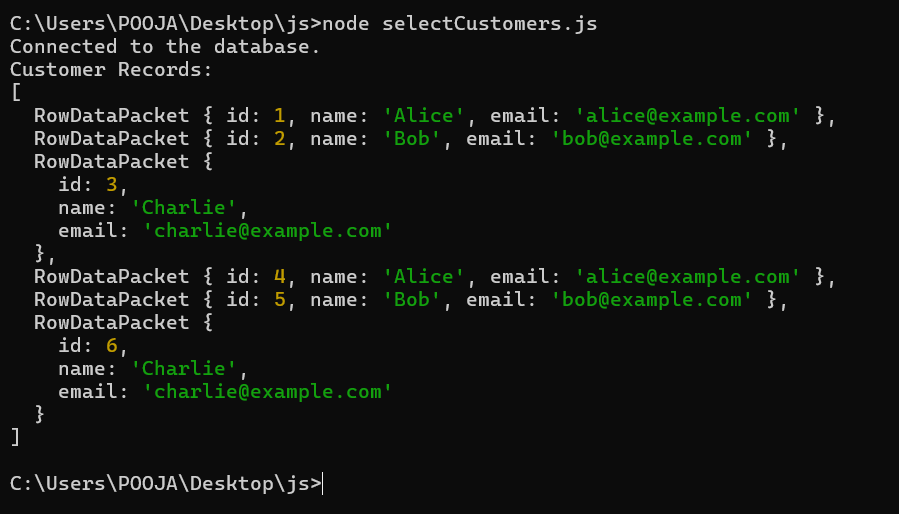
// SQL query to select all customers const sql = 'SELECT \* FROM customers';

// Execute the query connection.query(sql, (err, result) => { if (err) throw err; console.log('Customer Records:'); console.log(result);

// Close the connection connection.end();

});

});

***OUTPUT:*** (Run your Node.js file using: node selectCustomers.js)

**Program 20:** Create a Node.js file that inserts multiple records in the "student" table and displays the result object on the console

**Sol:** (Create a File with name: insertStudents.js) const mysql = require('mysql');

// Create a connection to the database

const connection = mysql.createConnection({ host: 'localhost',

user: 'root',

password: '', // no password for root in XAMPP by default database: 'college'

});

// Connect to the database connection.connect((err) => { if (err) throw err;

console.log('Connected to the database.');

// SQL query to insert multiple records

const sql = `INSERT INTO student (name, course, age) VALUES ('John Doe', 'Computer Science', 21),

('Jane Smith', 'Mathematics', 22), ('Alice Johnson', 'Physics', 20)`;

// Execute the query connection.query(sql, (err, result) => { if (err) throw err;

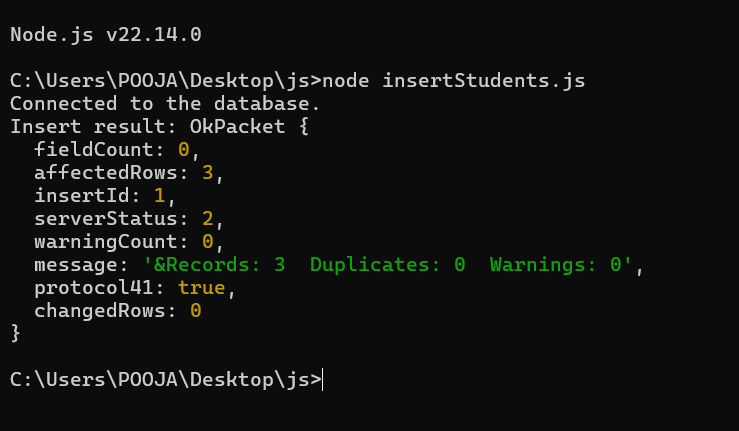
console.log('Insert result:', result);

// Close the connection connection.end();

});

});

***OUTPUT***: (From the terminal after running node insertStudents.js)



**Program 21:** Create a Node.js file that selects all records from the "customers" table and deletes the specified record

**Sol:** (Save this code as deleteCustomer.js in your project folder) const mysql = require('mysql');

// Create a connection to the database

const connection = mysql.createConnection({ host: 'localhost',

user: 'root',

password: '', // put your MySQL password if any database: 'college'

});

// Connect to MySQL connection.connect((err) => { if (err) throw err;

console.log('Connected to the database.');

// Select all records before deletion

connection.query('SELECT \* FROM customers', (err, results) => { if (err) throw err;

console.log('Customer Records Before Deletion:'); console.log(results);

// Now delete a specific customer (for example, customer with id = 2) const customerIdToDelete = 2;

connection.query('DELETE FROM customers WHERE id = ?', [customerIdToDelete], (err, deleteResult) => {

if (err) throw err;

console.log(`Deleted record with id ${customerIdToDelete}.`); console.log('Delete Result:', deleteResult);

// Show remaining records

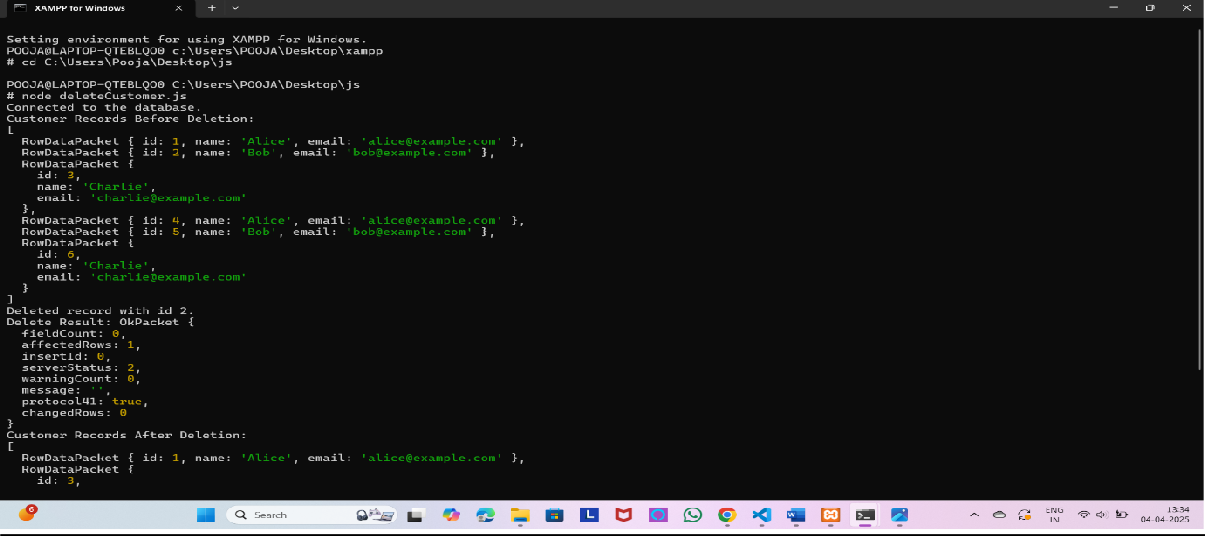
connection.query('SELECT \* FROM customers', (err, finalResults) => { if (err) throw err;

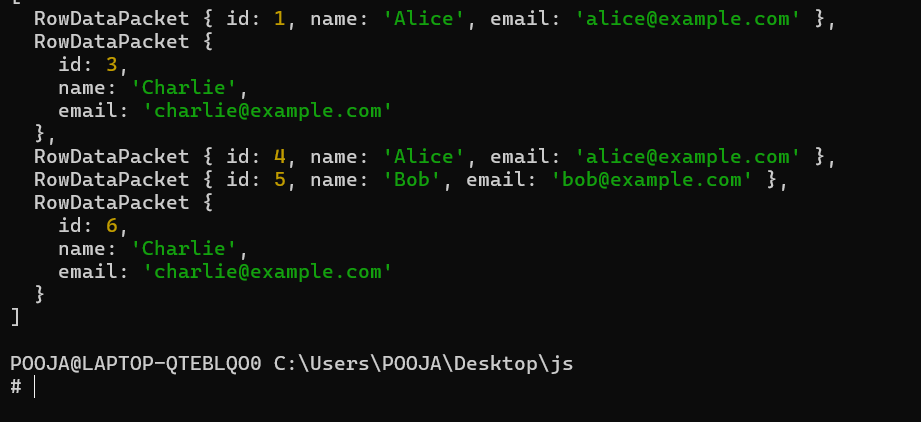
console.log('Customer Records After Deletion:'); console.log(finalResults);

connection.end();

}); }); });});

# OUTPUT:





**Program 22:** Create a Simple Web Server using Node.js.

**Sol:** (Create a file — server.js) const http = require('http');

// Create the server

const server = http.createServer((req, res) => { res.statusCode = 200; // OK

res.setHeader('Content-Type', 'text/plain');

res.end('Hello, this is your simple Node.js web server!\n');

});

// Choose a port const PORT = 3000;

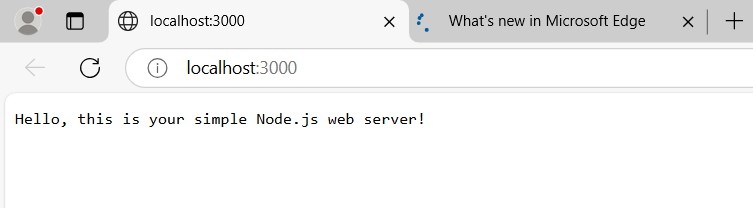
// Start the server

server.listen(PORT, () => {

console.log(`Server is running at [http://localhost:$](http://localhost/){PORT}/`);

});

### OUTPUT:

******

**Program 23:** Using Node.js, create a User Login System

**Sol:** Create from terminal (quick) run this command in command terminal : notepad index.js

const express = require('express');

const bodyParser = require('body-parser'); const bcrypt = require('bcryptjs');

const fs = require('fs'); const app = express(); const PORT = 3000;

app.use(bodyParser.json());

const USERS\_FILE = './users.json'; let users = [];

if (fs.existsSync(USERS\_FILE)) {

users = JSON.parse(fs.readFileSync(USERS\_FILE, 'utf8'));

}

function saveUsers() {

fs.writeFileSync(USERS\_FILE, JSON.stringify(users, null, 2));

}

app.post('/register', async (req, res) => { const { username, password } = req.body;

if (users.find(u => u.username === username)) {

return res.status(400).json({ message: 'User already exists.' });

}

const hashedPassword = await bcrypt.hash(password, 10); users.push({ username, password: hashedPassword });

saveUsers();

res.status(201).json({ message: 'User registered successfully!' });

});

app.post('/login', async (req, res) => {

const { username, password } = req.body;

const user = users.find(u => u.username === username); if (!user) {

return res.status(400).json({ message: 'Invalid username or password.' });

}

const isMatch = await bcrypt.compare(password, user.password); if (!isMatch) {

return res.status(400).json({ message: 'Invalid username or password.' });

}

res.json({ message: 'Login successful!' });

});

app.listen(PORT, () => {

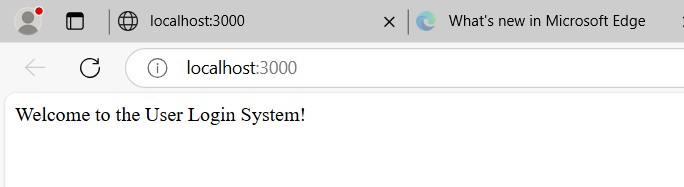
console.log(`Server running at [http://localhost:$](http://localhost/){PORT}`);

});

Now, create one more file in the same way: notepad users.json [ ]

### OUTPUT:

Run this command :node index.js



**Program 24:** Using Node.js, create an eLearning System.

#### Sol:

const express = require('express'); const app = express();

// Define a route for "/" app.get('/', (req, res) => {

res.send('<h1>Welcome to the eLearning System!</h1>');

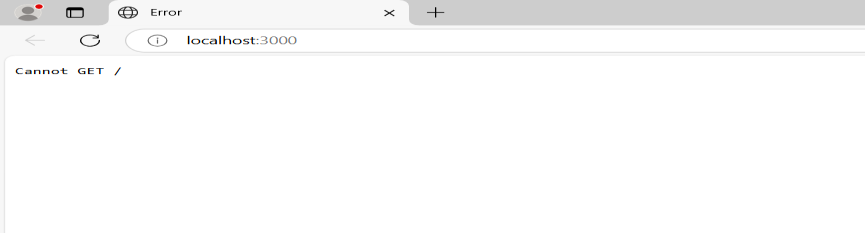
});

// Start the server const PORT = 3000; app.listen(PORT, () => {

console.log(`Server running at [http://localhost:$](http://localhost/){PORT}`);

});

### OUTPUT:

******

**Program 25:** Using Node.js, create a Recipe Book

**Sol**: Server.js file

const express = require('express');

const bodyParser = require('body-parser'); const cors = require('cors');

const { v4: uuidv4 } = require('uuid'); const app = express();

app.use(bodyParser.json()); app.use(cors());

const PORT = process.env.PORT || 3000; let recipes = [];

app.post('/recipes', (req, res) => {

const { name, ingredients, instructions } = req.body;

const newRecipe = { id: uuidv4(), name, ingredients, instructions }; recipes.push(newRecipe);

res.status(201).json(newRecipe);

});

app.get('/recipes', (req, res) => res.json(recipes));

app.listen(PORT, () => console.log(`Server running on [http://localhost:$](http://localhost/){PORT}`));

(Make form.html file)

<!DOCTYPE html>

<html>

<head>

<title>Add Recipe</title>

</head>

<body>

<h2>Add a Recipe</h2>

<form id="recipeForm">

<input type="text" id="name" placeholder="Recipe Name" required><br>

<textarea id="ingredients" placeholder='Ingredients (comma separated)' required></textarea><br>

<textarea id="instructions" placeholder='Instructions' required></textarea><br>

<button type="submit">Add Recipe</button>

</form>

<script>

document.getElementById('recipeForm').addEventListener('submit', async (e) => { e.preventDefault();

const name = document.getElementById('name').value;

const ingredients = document.getElementById('ingredients').value.split(','); const instructions = document.getElementById('instructions').value;

const res = await fetch('[http://localhost:3000/recipes',](http://localhost:3000/recipes%27) { method: 'POST',

headers: {

'Content-Type': 'application/json'

},

body: JSON.stringify({ name, ingredients, instructions })

});

const data = await res.json(); alert('Recipe Added: ' + data.name);

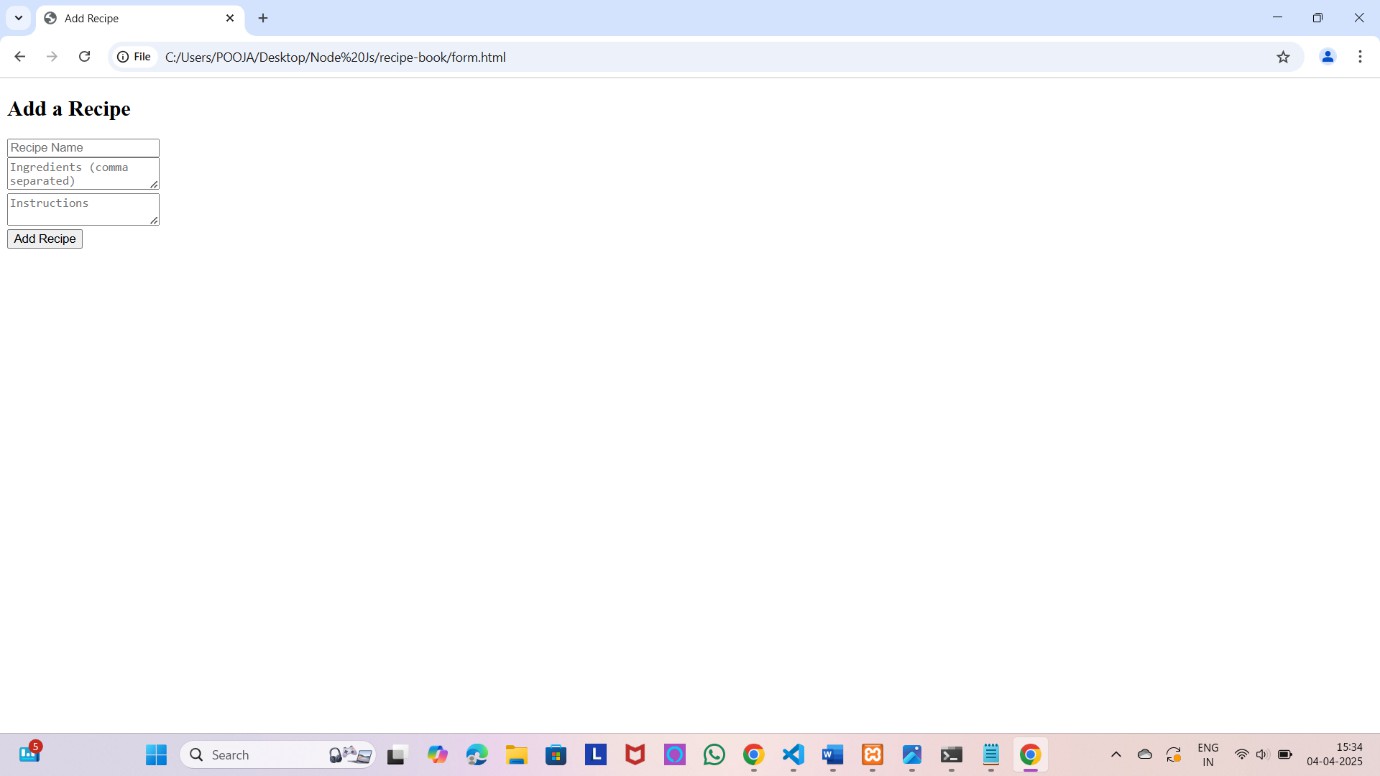
});

</script>

</body>

</html>

### OUTPUT:

******

**Program 26:** Write a Node.js script to interact with the file system and serve a web page from a file

**Sol:** 1. Create server.js

const http = require('http'); const fs = require('fs'); const path = require('path');

// Define the HTML file path

const filePath = path.join( dirname, 'index.html');

// Create the server

const server = http.createServer((req, res) => {

// Read the HTML file fs.readFile(filePath, (err, data) => {

if (err) {

res.writeHead(500, { 'Content-Type': 'text/plain' }); res.end('Internal Server Error');

} else {

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

});

}

});

// Start the server const PORT = 3000;

server.listen(PORT, () => {

console.log(`Server running at [http://localhost:$](http://localhost/){PORT}`);

});

2. Create index.html (Your Web Page)

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Simple Node.js Server</title>

<style>

body { font-family: Arial, sans-serif; text-align: center; padding: 20px; } h1 { color: #3498db; }

</style>

</head>

<body>

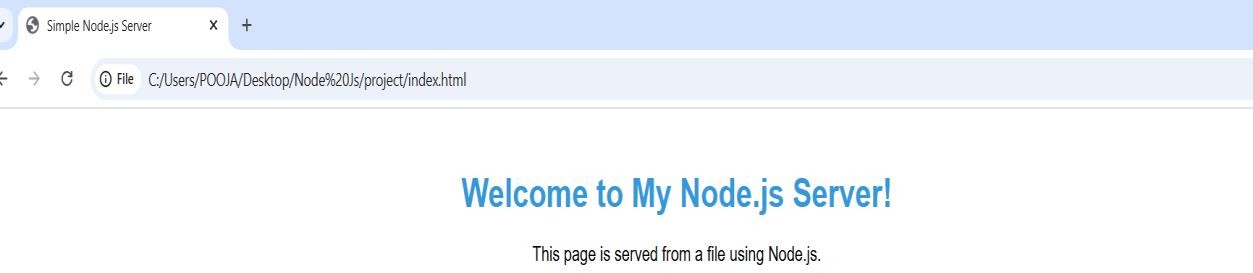
<h1>Welcome to My Node.js Server!</h1>

<p>This page is served from a file using Node.js.</p>

</body>

</html>

### OUTPUT:



**Program 27:** Write a Node.js script to build Your Own Node.js Module. Use require (‘http’) module, which is a built-in Node module that invokes the functionality of the HTTP library to create a local server. Also, use the export statement to make functions in your module available externally. Create a new text file to contain the functions in your module called “modules.js” and add this function to return today’s date and time.

**Sol:** Create a new file named: modules.js

// modules.js

// Function to get current date and time exports.getCurrentDateTime = function () {

return new Date().toLocaleString();

};

Create server.js (Main Server File)

const http = require('http'); // Import built-in HTTP module

const myModule = require('./modules'); // Import custom module

// Create an HTTP server

const server = http.createServer((req, res) => { res.writeHead(200, { 'Content-Type': 'text/plain' });

res.write(`Current Date and Time: ${myModule.getCurrentDateTime()}`); res.end();

});

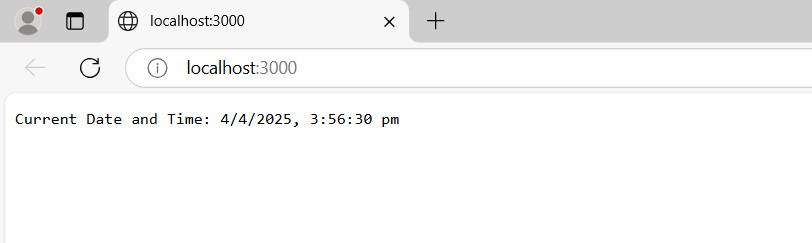
// Start the server const PORT = 3000;

server.listen(PORT, () => {

console.log(`Server running at [http://localhost:$](http://localhost/){PORT}`);

});

***OUTPUT:***

******

**Program 28:** Create a JS file named main.js for an event-driven application. There should be a main loop that listens for events and then triggers a callback function when one of those events is detected.

**Sol:** Create a JavaScript File (main.js)

// main.js

const EventEmitter = require('events'); // Import the events module

class MyEmitter extends EventEmitter {} // Create a class that extends EventEmitter

const myEmitter = new MyEmitter(); // Create an instance of MyEmitter myEmitter.on('userLoggedIn', (username) => {

console.log(`User Logged In: ${username}`);

});

myEmitter.on('dataReceived', (data) => { console.log(`Data Received: ${data}`);

});

// Simulating Events in the Main Loop console.log("Listening for events...");

// Simulate a user logging in after 2 seconds setTimeout(() => {

myEmitter.emit('userLoggedIn', 'Pooja');

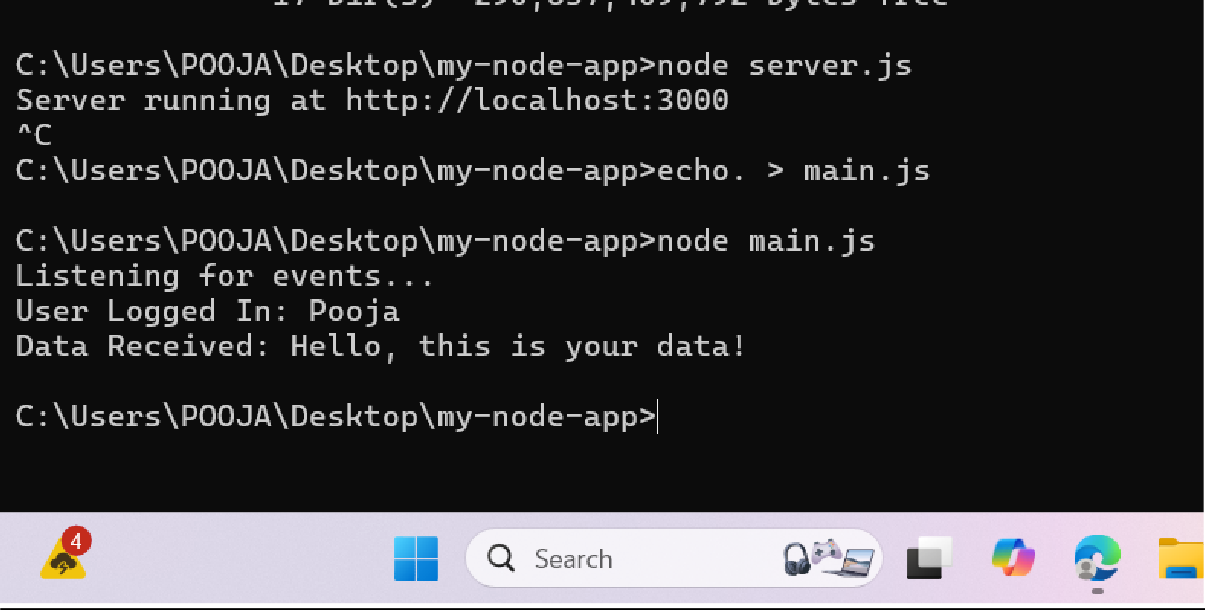
}, 2000);

// Simulate receiving data after 4 seconds setTimeout(() => {

myEmitter.emit('dataReceived', 'Hello, this is your data!');

}, 4000);

# OUTPUT:



**Program 29:** Write a Node.js application that transfers a file as an attachment on the web and enables the browser to prompt the user to download the file using Express.js.

**Sol:** Create server.js file

const express = require('express'); const path = require('path');

const app = express(); const PORT = 3000;

// Home route to guide users app.get('/', (req, res) => {

res.send('Welcome! Click <a href="/download">here</a> to download the file.');

});

// Route to serve the file for download app.get('/download', (req, res) => {

const filePath = path.join( dirname, 'sample.txt'); // Ensure sample.txt exists res.download(filePath, 'downloaded-file.txt', (err) => {

if (err) {

console.error('Error downloading file:', err); res.status(500).send('File not found');

});

}

});

// Start the server app.listen(PORT, () => {

console.log(`Server is running at [http://localhost:$](http://localhost/){PORT}`);

});

Create a sample.txt file

***OUTPUT:***

