A

PRACTICAL JOURNAL

ON

LAB ON WEB DEVELOPMENT

By

VIJAY MALAV

MC25B66

MASTER OF COMPUTER APPLICATIONS

ACADEMIC YEAR 2025-2027

DNYAAN PRASAD GLOBAL UNIVERSITY, PUNE

Dnyaan Prasad Global University
Dr. D. Y. Patil Unitech Society
School of Management & Research

Date:27/11/2025

CERTIFICATE

This is to certify that **Mr. VIJAY MALAV** has successfully completed the Practical Journal of **"WEB DEVELOPMENT"** as a partial fulfilment of his/her **Master of Computer Applications (MCA) Semester -I** under the curriculum of **Dnyaan Prasad Global University, Pune** for the academic year 2025-26.

Prof. Prasad Shaha	Mr. Amit Shrivastava	Dr. Shikha Dubey
Course Incharge	**Program Head, MCA**	**Head,Computer Application**

Dnyaan Prasad Global University
Dr. D. Y. Patil Unitech Society
School of Management & Research

Ref. No.: Date:

+	PROGRAM NAME	SIGN
1	Lab 1 – Design a College Department Website using HTML5 & CSS3	
2	Lab 2 – Create a JavaScript Registration Form with Validation	
3	Lab 3 – JavaScript DOM Product Catalog	
4	Lab 4 – Weather Dashboard using Fetch API & Async/Await	
5	Lab 5 – JavaScript OOP Task Manager with LocalStorage	
6	Lab 6 – TypeScript Student Result Management System	
7	Lab 7 – Angular Employee Directory (Components & Services)	
8	Lab 8 – Angular News Feed App using HTTP Client	
9	Lab 9 – Angular Routing – Online Course Portal	
10	Lab 10 – MCA Student Information Portal (Capstone)	
11	Q1 – Multi-section Webpage using Semantic Tags	
12	Q2 – Student Feedback Form using HTML5 Elements	
13	Q3 – CSS Selectors Demonstration	
14	Q4 – CSS Grid 3-Column Homepage	
15	Q5 – Responsive Layout using Media Queries	

16	Q6 – Banner Animation using CSS3	
17	Q7 – var/let/const Scope Demonstration	
18	Q8 – Student Records using Objects & Arrays	
19	Q9 – Array Methods: slice, map, filter, reduce	
20	Q10 – JavaScript Calculator	
21	Q11 – Block & Function Scope Example	
22	Q12 – Modify DOM Text & Style	
23	Q13 – Click Counter using Event Listener	
24	Q14 – To-Do List Application	
25	Q15 – Image Gallery with Preview Change	
26	Q16 – Login Form Validation	
27	Q17 – Promise-based API Delay Simulation	
28	Q18 – Async/Await API Simulation	
29	Q19 – Student Class with Grade Calculation	
30	Q20 – Hello TypeScript Program	
31	Q21 – Type Annotations Demo	
32	Q22 – TypeScript Student Manager (Classes & Modules)	
33	Q23 – Angular Project Setup using CLI	
34	Q24 – Angular Student Info Component	
35	Q25 – Two-Way Binding using ngModel	
36	Q26 – Reusable Angular Student Card Component	

Dnyaan Prasad Global University
Dr. D. Y. Patil Unitech Society
School of Management & Research

NAME == VIJAY MALAV
ROLL NO==MCA25B66

LAB ASSIGMENT

Lab 1-

HTML5 and CSS3 – Structured & Responsive Web Design
Case Title: Design a College Department Website (Static Layout)
Objective: To create a multi -page static website using semantic HTML5 elements and modern CSS3 design features.
Tasks:
• Develop structured pages like Home , About Us , Faculty , Courses , Contact .
• Use semantic tags (header , nav , section , article , footer).
• Apply CSS3 Selectors, Flexbox, and Grid for layout.
• Implement media queries to make the site responsive for mobile and desktop views.
• Add a CSS animation for a navigation hover effect or banner text.
Outcome: A fully responsive static website prototype for a college department. make all requriements are full filed and code gives proper also,make diff files for all html,css, and so

Program.html-

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>College Department - Home</title>
   <link rel="stylesheet" href="css/style.css">
```

```
</head>
<body>

<header>
  <h1 class="banner-text">Computer Science Department</h1>
</header>

<nav>
  <ul>
    <li><a href="program.html">Home</a></li>
    <li><a href="about.html">About Us</a></li>
    <li><a href="faculty.html">Faculty</a></li>
    <li><a href="courses.html">Courses</a></li>
    <li><a href="contact.html">Contact</a></li>
  </ul>
</nav>

<section class="hero">
  <article>
    <h2>Welcome to the Department</h2>
    <p>
      Our department aims to provide high-quality education and prepare
students
      for successful careers in technology.
    </p>
  </article>
</section>

<footer>
  <p>© 2025 Computer Science Department</p>
</footer>

</body>
</html>
```

About.html-

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
  <title>About Us - College Department</title>
  <link rel="stylesheet" href="css/style.css">
</head>
<body>

<header><h1>About Us</h1></header>

<nav>
  <ul>
    <li><a href="program.html">Home</a></li>
    <li><a href="about.html" class="active">About Us</a></li>
    <li><a href="faculty.html">Faculty</a></li>
    <li><a href="courses.html">Courses</a></li>
    <li><a href="contact.html">Contact</a></li>
  </ul>
</nav>

<section class="content">
  <article>
    <h2>Department Overview</h2>
    <p>
      The Computer Science Department offers advanced programs in computing,
      programming, and research-driven education.
    </p>
  </article>
</section>

<footer><p>© 2025 Computer Science Department</p></footer>

</body>
</html>
```

Faculty.html-

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Faculty - College Department</title>
  <link rel="stylesheet" href="css/style.css">
</head>
```

```
<body>

<header><h1>Our Faculty</h1></header>

<nav>
  <ul>
    <li><a href="index.html">Home</a></li>
    <li><a href="about.html">About Us</a></li>
    <li><a href="faculty.html" class="active">Faculty</a></li>
    <li><a href="courses.html">Courses</a></li>
    <li><a href="contact.html">Contact</a></li>
  </ul>
</nav>

<section class="grid">
  <article class="card">
    <h3>Dr. A. Sharma</h3>
    <p>Professor (AI & ML)</p>
  </article>

  <article class="card">
    <h3>Dr. R. Iyer</h3>
    <p>Professor (Data Science)</p>
  </article>

  <article class="card">
    <h3>Prof. Neha Gupta</h3>
    <p>Assistant Professor (Web Development)</p>
  </article>
</section>

<footer><p>© 2025 Computer Science Department</p></footer>

</body>
</html>
```

Courses.html-

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
  <title>Courses - College Department</title>
  <link rel="stylesheet" href="css/style.css">
</head>
<body>

<header><h1>Courses Offered</h1></header>

<nav>
  <ul>
    <li><a href="program.html">Home</a></li>
    <li><a href="about.html">About Us</a></li>
    <li><a href="faculty.html">Faculty</a></li>
    <li><a href="courses.html" class="active">Courses</a></li>
    <li><a href="contact.html">Contact</a></li>
  </ul>
</nav>

<section class="content">
  <article>
    <h2>Undergraduate & Postgraduate Programs</h2>
    <ul>
      <li>BSc Computer Science</li>
      <li>BCA</li>
      <li>MSc Computer Science</li>
      <li>PG Diploma in AI</li>
    </ul>
  </article>
</section>

<footer><p>© 2025 Computer Science Department</p></footer>

</body>
</html>
```

Contact.html-

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Contact - College Department</title>
  <link rel="stylesheet" href="css/style.css">
</head>
```

```
<body>

<header><h1>Contact Us</h1></header>

<nav>
  <ul>
    <li><a href="program.html">Home</a></li>
    <li><a href="about.html">About Us</a></li>
    <li><a href="faculty.html">Faculty</a></li>
    <li><a href="courses.html">Courses</a></li>
    <li><a href="contact.html" class="active">Contact</a></li>
  </ul>
</nav>

<section class="content">
  <article>
    <h2>Get in Touch</h2>
    <p>Email: csdept@college.edu</p>
    <p>Phone: +91 98765 43210</p>
  </article>
</section>

<footer><p>© 2025 Computer Science Department</p></footer>

</body>
</html>
```

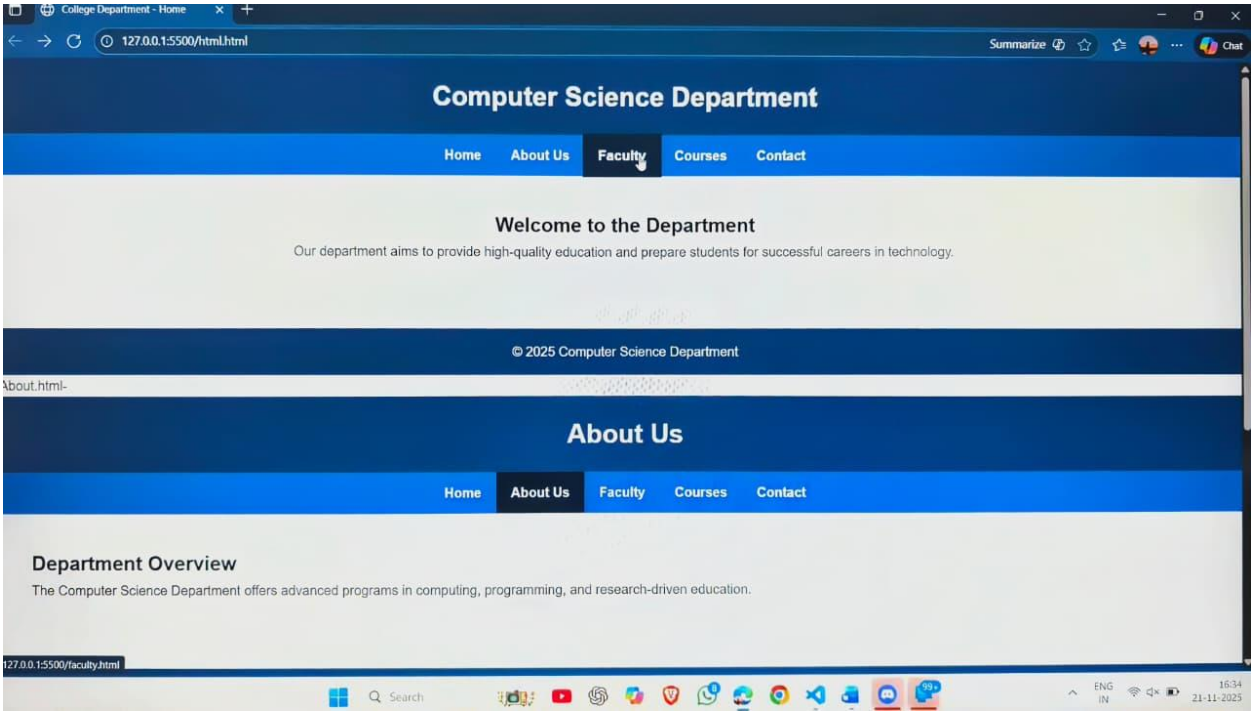
Style.css-

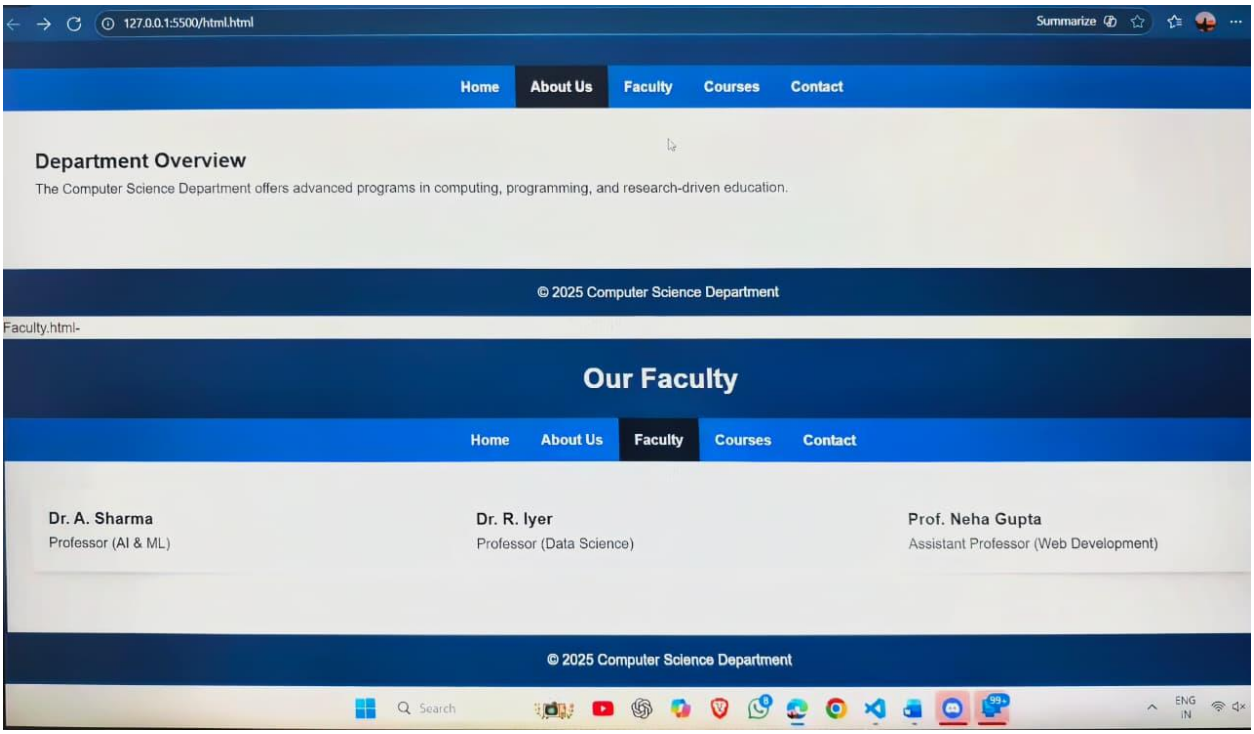
```
/* Basic Page Reset */
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}

body {
  font-family: Arial, sans-serif;
  line-height: 1.6;
  background: #f6f6f6;
}

header {
```

```
    background: #003366;
    color: white;
    padding: 20px;
    text-align: center;
}

/* Navigation */
nav ul {
    list-style: none;
    display: flex;
    justify-content: center;
    background: #0055aa;
}

nav ul li {
    margin: 0;
}

nav ul li a {
    display: block;
    padding: 12px 18px;
    color: white;
    text-decoration: none;
    font-weight: bold;
    transition: 0.3s;
}

/* Hover animation */
nav ul li a:hover {
    background: #002244;
    transform: scale(1.05);
}

/* Active page */
.active {
    background: #001a33;
}

/* Hero Section */
.hero {
    padding: 40px;
    text-align: center;
}
```

```
/* Content Section */
.content {
  padding: 40px;
}

/* Faculty Cards Grid */
.grid {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(220px, 1fr));
  gap: 20px;
  padding: 30px;
}

.card {
  background: white;
  padding: 20px;
  border-radius: 8px;
  box-shadow: 0 3px 5px rgba(0,0,0,0.2);
}

/* Banner Animation */
.banner-text {
  animation: slideIn 2s ease-out;
}

@keyframes slideIn {
  from { opacity: 0; transform: translateY(-20px); }
  to   { opacity: 1; transform: translateY(0); }
}

/* Footer */
footer {
  background: #003366;
  color: white;
  text-align: center;
  padding: 15px;
  margin-top: 40px;
}

/* Responsive Media Queries */
@media (max-width: 768px) {
  nav ul {
```

```
    flex-direction: column;
  }

  header h1 {
    font-size: 22px;
  }
}
```





Lab 2

JavaScript Basics – Interactive User Form
Case Title: Online Registration Form with Validation
Objective: To implement client -side validation and interactivity using JavaScript.

Tasks:
• Create a student registration form with input fields (name, email, phone, course, etc.).
• Validate data dynamically (email format, mandatory fields, numeric constraints).
• Use event listeners for form submission and input events.
• Display real -time validation messages using DOM manipulation.
Outcome: Functional form that validates user input without page reloadHTML CODE –

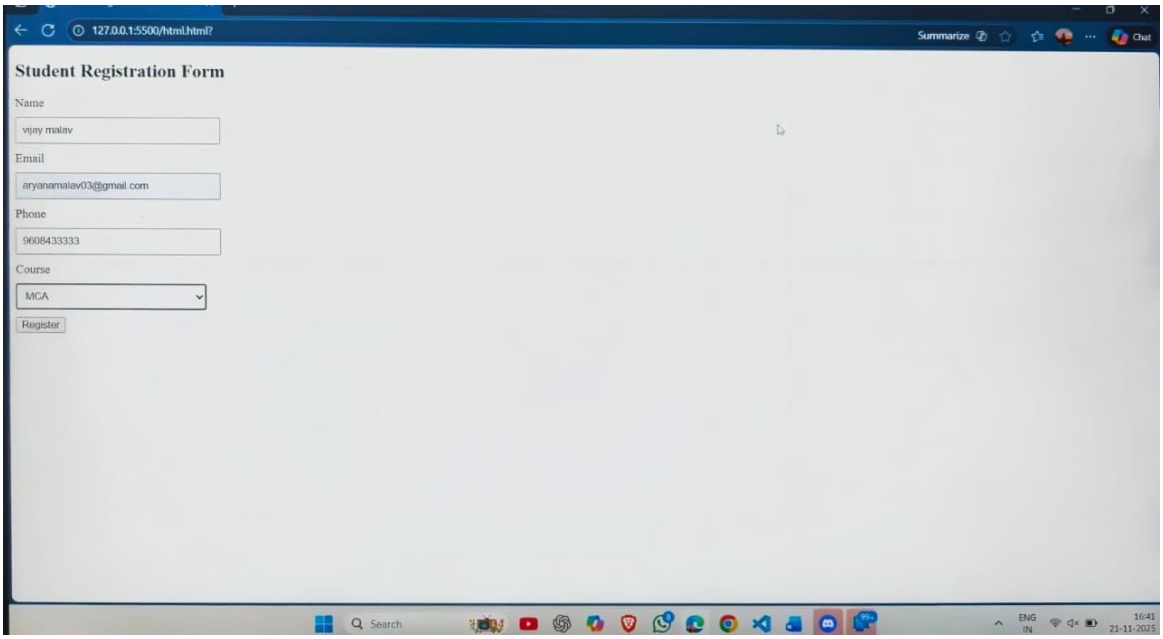
```
<!DOCTYPE html>
<html>
<head>
<title>Student Registration</title>
```

```
<style>
input, select { display:block; margin:10px 0; padding:8px; width:250px; }
span { color:red; font-size:12px; }
</style>
</head>

<body>

<h2>Student Registration Form</h2>

<form id="regForm">

  <label>Name</label>
  <input type="text" id="name">
  <span id="nameErr"></span>

  <label>Email</label>
  <input type="text" id="email">
  <span id="emailErr"></span>

  <label>Phone</label>
  <input type="text" id="phone">
  <span id="phoneErr"></span>

  <label>Course</label>
  <select id="course">
    <option value="">Select</option>
    <option>MCA</option>
    <option>BCA</option>
    <option>BCS</option>
  </select>
  <span id="courseErr"></span>

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

</form>

<script src="form.js"></script>

</body>
</html>
```

Form.js

```
let form = document.getElementById("regForm");

// real-time input validation
document.getElementById("email").addEventListener("input",
validateEmail);
document.getElementById("phone").addEventListener("input",
validatePhone);

form.addEventListener("submit", function(e) {
   e.preventDefault();  // stop page reload
   validateForm();
});

function validateForm() {
   let valid = true;

   let name = document.getElementById("name").value;
   if (name.trim() === "") {
      document.getElementById("nameErr").innerText = "Name is required";
      valid = false;
   } else document.getElementById("nameErr").innerText = "";

   validateEmail();
   validatePhone();

   let course = document.getElementById("course").value;
   if (course === "") {
      document.getElementById("courseErr").innerText = "Please select a
course";
      valid = false;
   } else document.getElementById("courseErr").innerText = "";

   if (valid) alert("Registration Successful!");
}

function validateEmail() {
   let email = document.getElementById("email").value;
   let emailErr = document.getElementById("emailErr");

   let pattern = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;

   if (email.trim() === "") emailErr.innerText = "Email is required";
   else if (!pattern.test(email)) emailErr.innerText = "Invalid email format";
```

```
    else emailErr.innerText = "";
}

function validatePhone() {
    let phone = document.getElementById("phone").value;
    let phoneErr = document.getElementById("phoneErr");

    if (phone.trim() === "") phoneErr.innerText = "Phone is required";
    else if (isNaN(phone)) phoneErr.innerText = "Numbers only";
    else if (phone.length !== 10) phoneErr.innerText = "Enter 10 digits";
    else phoneErr.innerText = "";
}
```



Lab 3

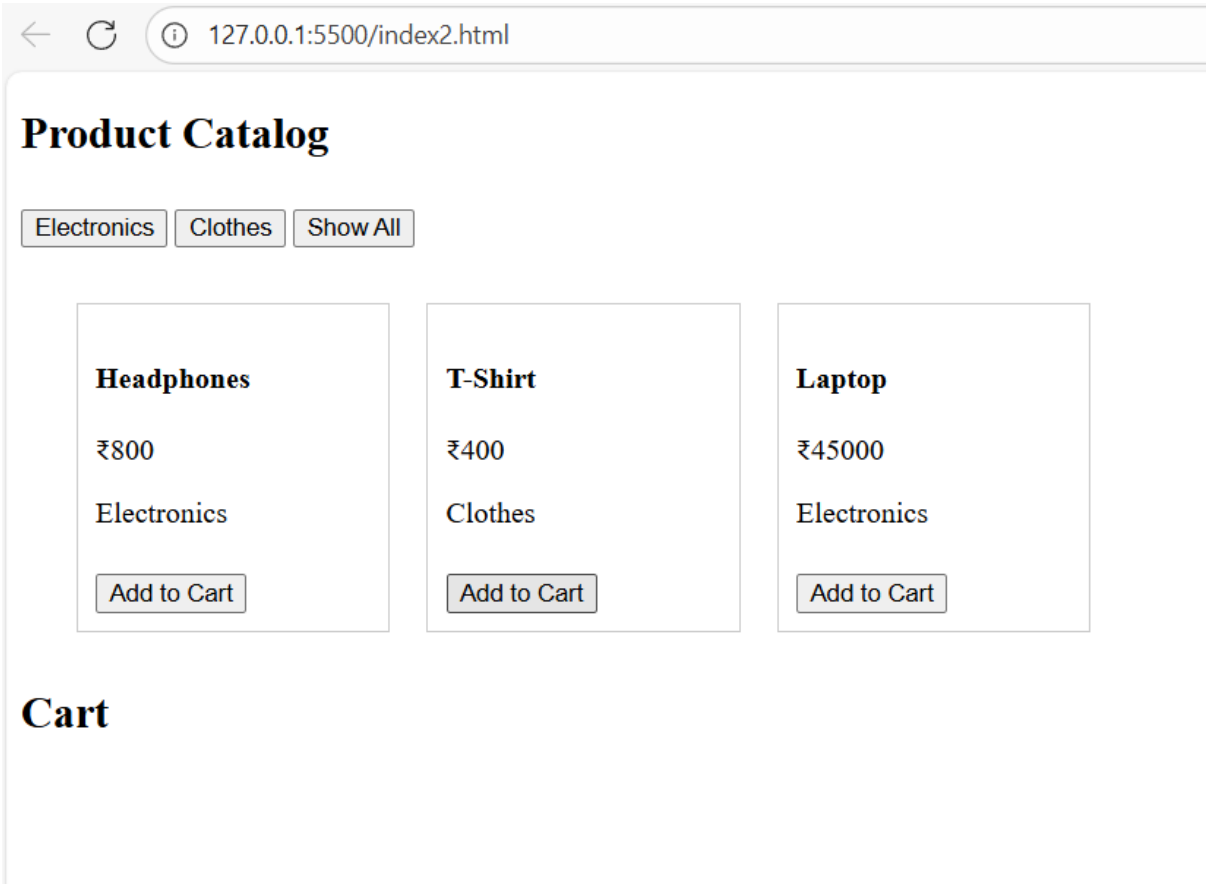
JavaScript DOM & Event Handling – Dynamic Product Catalog Case Title: E - Commerce Product Listing Page Objective: To dynamically generate and modify webpage content using the DOM API. Tasks: • Display a product list (name, image, price, category) using DOM creation methods. • Implement sorting/filtering buttons (e.g., by category or price). • Add event handling for "Add to Cart" and "View Details" buttons. • Display selected items dynamically in a cart section. Outcome: Interactive front -end product catalog simulating real -world e - commerce behavior.

```
<!DOCTYPE html>
<html>
<head>
<title>Product Catalog</title>
<style>
#products, #cart { margin:20px; }
.card { border:1px solid #ccc; padding:10px; margin:10px; width:150px;
display:inline-block; }
button { margin-top:8px; }
</style>
</head>

<body>

<h2>Product Catalog</h2>

<button onclick="filterCategory('Electronics')">Electronics</button>
<button onclick="filterCategory('Clothes')">Clothes</button>
<button onclick="showAll()">Show All</button>

<div id="products"></div>

<h2>Cart</h2>
<div id="cart"></div>

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

App.js

```
let products = [
   { name:"Headphones", price:800, category:"Electronics" },
   { name:"T-Shirt", price:400, category:"Clothes" },
   { name:"Laptop", price:45000, category:"Electronics" }
];

let cart = [];

function displayProducts(list) {
   let container = document.getElementById("products");
   container.innerHTML = "";
```

```
  list.forEach(p => {
    let card = document.createElement("div");
    card.className = "card";

    card.innerHTML = `
      <h4>${p.name}</h4>
      <p>₹${p.price}</p>
      <p>${p.category}</p>
    `;

    let btn = document.createElement("button");
    btn.innerText = "Add to Cart";
    btn.onclick = () => addToCart(p);

    card.appendChild(btn);
    container.appendChild(card);
  });
}

function addToCart(product) {
  cart.push(product);
  showCart();
}

function showCart() {
  let c = document.getElementById("cart");
  c.innerHTML = cart.map(item => `<p>${item.name} -
₹${item.price}</p>`).join("");
}

function filterCategory(cat) {
  let filtered = products.filter(p => p.category === cat);
  displayProducts(filtered);
}

function showAll() {
  displayProducts(products);
}


displayProducts(products);
```



Lab 4-

Advanced JavaScript – Asynchronous API Integration Case Title: Live Weather Dashboard Objective: To fetch and display real -time data using fetch API, Promises, and async/await. Tasks: • Integrate any free weather API (e.g., OpenWeatherMap). • Allow user input for a city and fetch weather details dynamically. • Display temperature, humidity, and forecast icons using DOM updates. • Handle errors (invalid city or failed API call) using proper error messages. Outcome: Real -time, asynchronous web app with practical data fetching functionality.

HTML Code

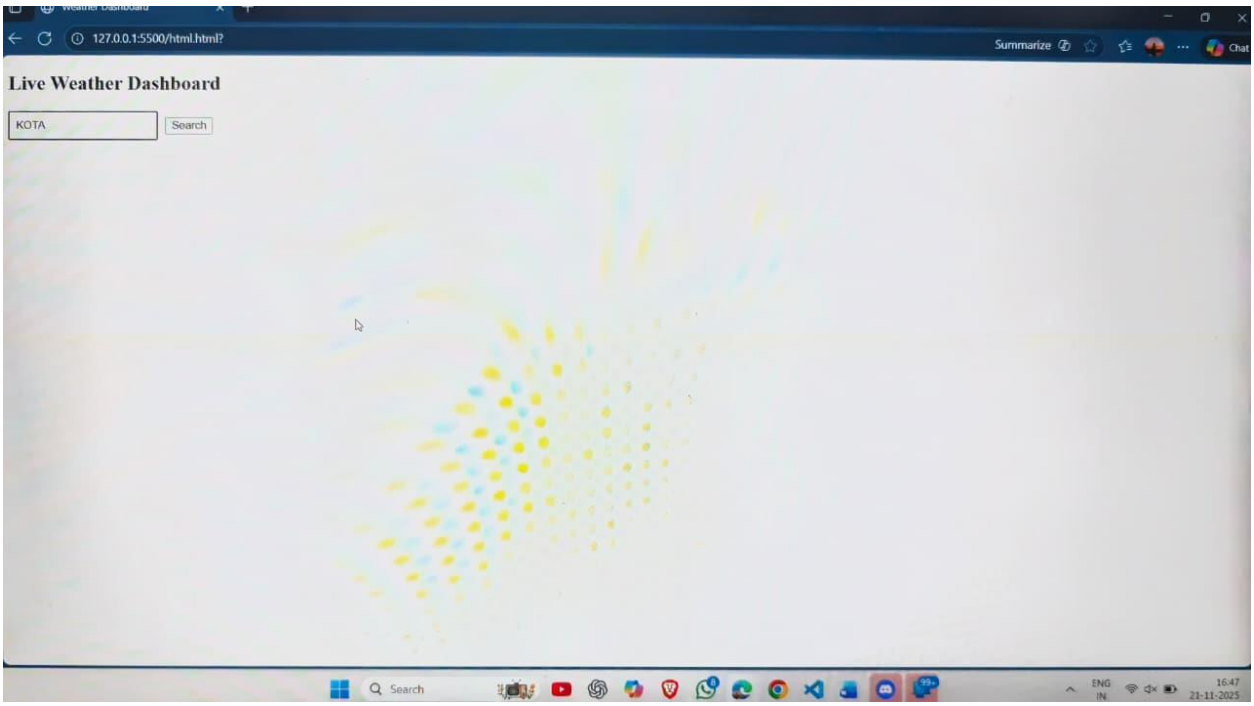
```
<!DOCTYPE html>
<html>
<head>
<title>Weather Dashboard</title>
<style>
input { padding:8px; margin-right:5px; }
#weather { margin-top:15px; font-size:18px; }
</style>
```

```
</head>

<body>

<h2>Live Weather Dashboard</h2>

<input type="text" id="city" placeholder="Enter City">
<button onclick="getWeather()">Search</button>

<div id="weather"></div>

<script src="weather.js"></script>
</body>
</html>
```

Wether.js

```
async function getWeather() {
   let city = document.getElementById("city").value;
   let box = document.getElementById("weather");

   if (city.trim() === "") {
      box.innerHTML = "Please enter a city name.";
      return;
   }

   try {

leturl
='https://api.openweathermap.org/data/2.5/weather?q=${city}&appid=abc1
23def456gh789&units=metric';

      let res = await fetch(url);

      if (!res.ok) throw new Error("City not found");

      let data = await res.json();

      let temp = data.main.temp;
      let humidity = data.main.humidity;
      let icon = data.weather[0].icon;

      box.innerHTML = `
```

```
        <p>Temperature: ${temp}°C</p>
        <p>Humidity: ${humidity}%</p>
        <img src="https://openweathermap.org/img/wn/${icon}.png">
      `;
    }
    catch (err) {
      box.innerHTML = "Error: " + err.message;
    }
}
```



Lab 5-

Advanced JavaScript – Object -Oriented Programming Case Title: Task Management System (To -Do Application) Objective: To design and implement a task manager using OOP concepts. Tasks: • Create a Task class with attributes like title, description, and status. • Add, edit, and delete tasks dynamically using DOM methods. • Implement data persistence using Local Storage . • Use inheritance to create specialized task types (e.g., WorkTask,

PersonalTask). Outcome: Reusable, object -oriented JavaScript web app for task management.

HTML CODE:

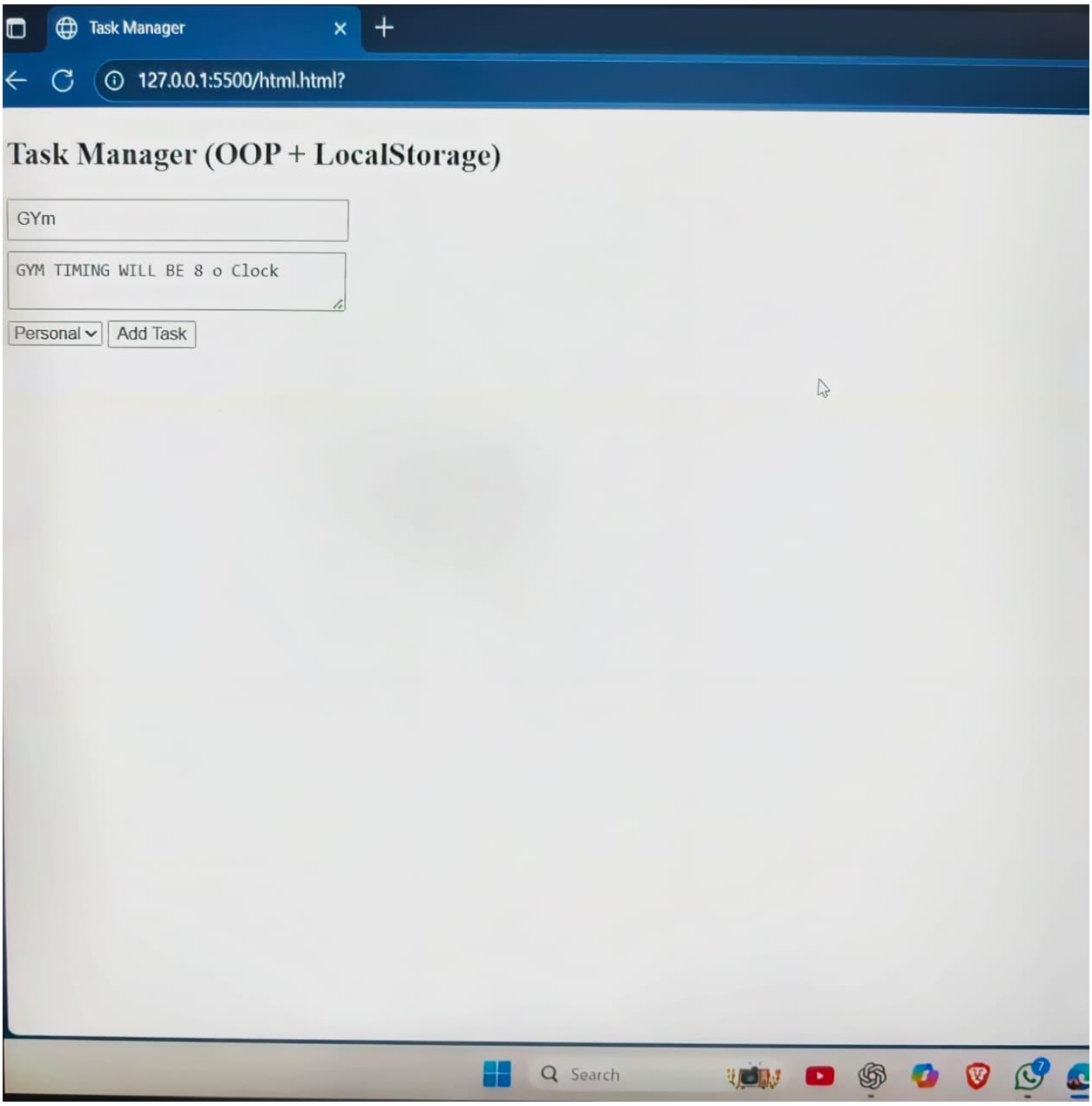
```
<!DOCTYPE html>
<html>
<head>
<title>Task Manager</title>
<style>
input, textarea { display:block; margin:8px 0; width:250px; padding:6px; }
button { margin-right:6px; }
.task { border:1px solid #ccc; padding:10px; margin:10px 0; }
</style>
</head>

<body>

<h2>Task Manager (OOP + LocalStorage)</h2>

<input type="text" id="title" placeholder="Task Title">
<textarea id="desc" placeholder="Description"></textarea>

<select id="type">
  <option value="personal">Personal</option>
  <option value="work">Work</option>
</select>

<button onclick="addTask()">Add Task</button>

<div id="taskList"></div>

<script src="task.js"></script>
</body>
</html>
```

Task.js

```
class Task {
  constructor(title, description) {
    this.title = title;
    this.description = description;
    this.status = "Pending";
  }
}
```

```
class WorkTask extends Task {}
class PersonalTask extends Task {}

let tasks = JSON.parse(localStorage.getItem("tasks")) || [];

function addTask() {
   let title = document.getElementById("title").value;
   let desc = document.getElementById("desc").value;
   let type = document.getElementById("type").value;

   if (title.trim() === "") {
      alert("Title required");
      return;
   }

   let task = (type === "work") ? new WorkTask(title, desc)
                        : new PersonalTask(title, desc);

   tasks.push(task);
   saveTasks();
   showTasks();
}

function deleteTask(index) {
   tasks.splice(index, 1);
   saveTasks();
   showTasks();
}

function editTask(index) {
   let newTitle = prompt("New title:", tasks[index].title);
   if (newTitle !== null) tasks[index].title = newTitle;

   let newDesc = prompt("New description:", tasks[index].description);
   if (newDesc !== null) tasks[index].description = newDesc;

   saveTasks();
   showTasks();
}

function toggleStatus(index) {
   tasks[index].status = tasks[index].status === "Pending" ? "Completed" :
```

```
"Pending";
   saveTasks();
   showTasks();
}

function saveTasks() {
   localStorage.setItem("tasks", JSON.stringify(tasks));
}
function showTasks() {
   let box = document.getElementById("taskList");
   box.innerHTML = "";
  tasks.forEach((t, i) => {
       let div = document.createElement("div");
       div.className = "task";

       div.innerHTML = `
          <b>${t.title}</b> (${t.status})<br>
          <small>${t.description}</small><br><br>
       `;

       let btn1 = document.createElement("button");
       btn1.innerText = "Edit";
       btn1.onclick = () => editTask(i);

       let btn2 = document.createElement("button");
       btn2.innerText = "Delete";
       btn2.onclick = () => deleteTask(i);
 let btn3 = document.createElement("button");
       btn3.innerText = "Toggle Status";
       btn3.onclick = () => toggleStatus(i);
 div.appendChild(btn1);
       div.appendChild(btn2);
       div.appendChild(btn3);
  box.appendChild(div);
   });
}
showTasks();
```



Lab 6

TypeScript Fundamentals – Type -Safe Application

Case Title: Student Result Management using TypeScript

Objective: To demonstrate use of TypeScript's type system, interfaces, and classes.

Tasks:

- Set up a TypeScript project (tsconfig.json).
- Create interfaces for Student and Result objects.
- Write classes with methods to calculate total marks and grade.
- Compile TypeScript into JavaScript and display results in a web page.

Outcome: Type -safe application illustrating OOP and interface -based programming.

FILE 1: tsconfig.json

```
{
  "compilerOptions": {
    "target": "es6",
    "module": "commonjs",
    "outDir": "./dist",
    "strict": true
  }
}
```

index.ts (TypeScript Source Code)

```
interface Student {
    id: number;
    name: string;
    marks: number[];
}

interface Result {
    total: number;
    grade: string;
}

class ResultManager {
    calculateResult(student: Student): Result {
        const total = student.marks.reduce((sum, m) => sum + m, 0);
        let grade = "";

        if (total >= 250) {
            grade = "A";
        } else if (total >= 200) {
            grade = "B";
        } else if (total >= 150) {
            grade = "C";
        } else {
            grade = "Fail";
        }

        return { total, grade };
    }
}
```

```
const student: Student = {
   id: 1,
   name: "Vijay",
   marks: [78, 82, 50,80]
};

const rm = new ResultManager();
const result = rm.calculateResult(student);

document.getElementById("sid")!.innerText = student.id.toString();
document.getElementById("sname")!.innerText = student.name;
document.getElementById("smarks")!.innerText = student.marks.join(", ");
document.getElementById("stotal")!.innerText = result.total.toString();
document.getElementById("sgrade")!.innerText = result.grade;
```

dist/index.js (Compiled JavaScript Output)

```
"use strict";
class ResultManager {
   calculateResult(student) {
      const total = student.marks.reduce((sum, m) => sum + m, 0);
      let grade = "";
      if (total >= 250) {
         grade = "A";
      }
      else if (total >= 200) {
         grade = "B";
      }
      else if (total >= 150) {
         grade = "C";
      }
      else {
         grade = "Fail";
      }
      return { total, grade };
   }
}
const student = {
   id: 1,
   name: "vijay",
   marks: [78, 82, 50,80]
};
const rm = new ResultManager();
```

```
const result = rm.calculateResult(student);
document.getElementById("sid").innerText = student.id.toString();
document.getElementById("sname").innerText = student.name;
document.getElementById("smarks").innerText = student.marks.join(", ");
document.getElementById("stotal").innerText = result.total.toString();
document.getElementById("sgrade").innerText = result.grade;
```

index.html (Displays Result on Web Page)

```
<!DOCTYPE html>
<html>
<head>
  <title>Student Result Management</title>
</head>
<body>
  <h2>Student Result Management (TypeScript)</h2>

  <table border="1" cellpadding="8">
    <tr>
      <th>Student ID</th>
      <td id="sid"></td>
    </tr>
    <tr>
      <th>Name</th>
      <td id="sname"></td>
    </tr>
    <tr>
      <th>Marks</th>
      <td id="smarks"></td>
    </tr>
    <tr>
      <th>Total</th>
      <td id="stotal"></td>
    </tr>
    <tr>
      <th>Grade</th>
      <td id="sgrade"></td>
    </tr>
  </table>

  <script src="./dist/index.js"></script>
</body>
</html>
```

Student ID	1
Name	Vijay
Marks	78, 82, 50, 80
Total	290
Grade	A

Lab 7

Angular Basics – Component -Based Application
Case Title: Employee Directory Management System
Objective: To build a simple Angular project demonstrating components,

templates, and data binding.
Tasks:
• Set up Angular project using Angular CLI .
• Create components: EmployeeList , EmployeeDetails , and AddEmployee .
• Use two -way data binding for form inputs.
• Manage employee data using a service and display it dynamically.
Outcome: Functional Angular app following modular and component -based architecture.

File: app.module.ts

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { FormsModule } from '@angular/forms';
import { AppComponent } from './app.component';
import { EmployeeListComponent } from './employee-list/employee-list.component';
import { EmployeeDetailsComponent } from './employee-details/employee-details.component';
import { AddEmployeeComponent } from './add-employee/add-employee.component';
import { EmployeeService } from './employee.service';

@NgModule({
  declarations: [
    AppComponent,
```

```
    EmployeeListComponent,
    EmployeeDetailsComponent,
    AddEmployeeComponent
  ],
  imports: [BrowserModule, FormsModule],
  providers: [EmployeeService],
  bootstrap: [AppComponent]
})
export class AppModule {}
```

employee.service.ts

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

@Injectable({
  providedIn: 'root'
})
export class EmployeeService {
  employees = [
    { id: 1, name: 'VIJAY', role: 'Developer' },
    { id: 2, name: 'RAJ', role: 'Designer' }
  ];

  getEmployees() {
    return this.employees;
  }

  addEmployee(emp: any) {
    this.employees.push(emp);
  }

  getEmployee(id: number) {
    return this.employees.find(e => e.id === id);
  }
}
```

employee-list.component.ts

```
import { Component } from '@angular/core';
import { EmployeeService } from '../employee.service';

@Component({
  selector: 'app-employee-list',
```

```
  templateUrl: './employee-list.component.html'
})
export class EmployeeListComponent {
  employees: any[] = [];

  constructor(private empService: EmployeeService) {
    this.employees = this.empService.getEmployees();
  }
}
```

employee-list.component.html

```
<h2>Employee List</h2>
<ul>
  <li *ngFor="let e of employees">
    {{ e.id }} - {{ e.name }} - {{ e.role }}
  </li>
</ul>
```

employee-details.component.ts

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

@Component({
  selector: 'app-employee-details',
  templateUrl: './employee-details.component.html'
})
export class EmployeeDetailsComponent {
  @Input() employee: any;
}
```

employee-details.component.html

```
<div *ngIf="employee">
  <h2>Employee Details</h2>
  <p>ID: {{ employee.id }}</p>
  <p>Name: {{ employee.name }}</p>
  <p>Role: {{ employee.role }}</p>
</div>
```

add-employee.component.ts

```
import { Component } from '@angular/core';
import { EmployeeService } from '../employee.service';

@Component({
  selector: 'app-add-employee',
  templateUrl: './add-employee.component.html'
})
export class AddEmployeeComponent {
  id = 0;
  name = '';
  role = '';

  constructor(private empService: EmployeeService) {}

  add() {
    const emp = { id: this.id, name: this.name, role: this.role };
    this.empService.addEmployee(emp);
  }
}
```

add-employee.component.html

```
<h2>Add Employee</h2>

<label>ID</label>
<input type="number" [(ngModel)]="id">

<label>Name</label>
<input type="text" [(ngModel)]="name">

<label>Role</label>
<input type="text" [(ngModel)]="role">

<button (click)="add()">Add Employee</button>
```

app.component.html

```
<h1>Employee Directory Management System</h1>

<app-add-employee></app-add-employee>
```

<app-employee-list></app-employee-list>

Lab 8

Angular Services & HTTP Client Integration

Case Title: News Feed Application using Public API

Objective: To integrate Angular services and HTTP client for live data.

Tasks:

• Use HttpClientModule to fetch articles from a public API (like NewsAPI).

• Display the list of articles with title, image, and publication date.

• Implement a search feature to filter news by keyword.

• Add loading spinner and error handling for failed requests.

Outcome**:** Dynamic, service -driven Angular application with live API

File: app.module.ts

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { HttpClientModule } from '@angular/common/http';
import { FormsModule } from '@angular/forms';
import { AppComponent } from './app.component';
import { NewsService } from './news.service';

@NgModule({
  declarations: [AppComponent],
  imports: [BrowserModule, HttpClientModule, FormsModule],
  providers: [NewsService],
  bootstrap: [AppComponent]
})
export class AppModule {}
```

File: news.service.ts

```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';

@Injectable({
  providedIn: 'root'
})
export class NewsService {
  url = 'https://newsapi.org/v2/top-
```

```
headlines?country=in&apiKey=YOUR_API_KEY';

  constructor(private http: HttpClient) {}

  getNews() {
    return this.http.get(this.url);
  }
}


------------------------------------------------------------
File: app.component.ts
------------------------------------------------------------
import { Component } from '@angular/core';
import { NewsService } from './news.service';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html'
})
export class AppComponent {
  articles: any[] = [];
  filtered: any[] = [];
  loading = false;
  error = '';
  searchText = '';

  constructor(private news: NewsService) {
    this.loadNews();
  }

  loadNews() {
    this.loading = true;
    this.error = '';
    this.news.getNews().subscribe({
      next: (res: any) => {
        this.articles = res.articles;
        this.filtered = res.articles;
        this.loading = false;
      },
      error: () => {
        this.error = 'Failed to load news.';
        this.loading = false;
```

```
    }
   });
  }

  search() {
   this.filtered = this.articles.filter(a =>
    a.title.toLowerCase().includes(this.searchText.toLowerCase())
   );
  }
}
```

File: app.component.html

```
<h1>News Feed Application</h1>

<input type="text" placeholder="Search news..." [(ngModel)]="searchText"
(input)="search()">

<div *ngIf="loading">Loading...</div>
<div *ngIf="error">{{ error }}</div>

<div *ngIf="!loading">
  <div *ngFor="let n of filtered" style="border:1px solid #ccc; padding:10px;
margin:10px;">
   <h3>{{ n.title }}</h3>
   <img *ngIf="n.urlToImage" [src]="n.urlToImage" width="200">
   <p>{{ n.publishedAt }}</p>
  </div>
</div>
```

Lab 9

Angular Routing & SPA Development

Case Title: Online Course Portal (Single Page Application)

Objective: To demonstrate Angular routing and navigation among components.

Tasks:

- Create multiple pages: Home, Courses, Faculty, Contact.
- Use Angular Router for navigation without page reload.
- Implement route parameters to show details for each course.
- Add a 404 Not Found page for invalid routes.

Outcome: Fully functional single -page Angular application simulating a

real learning portal.

app.module.ts

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { RouterModule, Routes } from '@angular/router';
import { AppComponent } from './app.component';
import { HomeComponent } from './home/home.component';
import { CoursesComponent } from './courses/courses.component';
import { FacultyComponent } from './faculty/faculty.component';
import { ContactComponent } from './contact/contact.component';
import { CourseDetailsComponent } from './course-details/course-
details.component';
import { NotFoundComponent } from './not-found/not-found.component';

const routes: Routes = [
  { path: '', component: HomeComponent },
  { path: 'courses', component: CoursesComponent },
  { path: 'course/:id', component: CourseDetailsComponent },
  { path: 'faculty', component: FacultyComponent },
  { path: 'contact', component: ContactComponent },
  { path: '**', component: NotFoundComponent }
];

@NgModule({
  declarations: [
    AppComponent,
    HomeComponent,
    CoursesComponent,
    FacultyComponent,
    ContactComponent,
    CourseDetailsComponent,
    NotFoundComponent
  ],
  imports: [BrowserModule, RouterModule.forRoot(routes)],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule {}
```

home.component.ts

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

@Component({
  selector: 'app-home',
  templateUrl: './home.component.html'
})
export class HomeComponent {}
```

home.component.html

```
<h2>Welcome to the Online Course Portal</h2>
<p>Explore Courses, Faculty, and Contact Information.</p>
```

courses.component.ts

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

@Component({
  selector: 'app-courses',
  templateUrl: './courses.component.html'
})
export class CoursesComponent {
  courses = [
    { id: 1, title: 'Angular Basics' },
    { id: 2, title: 'Java Programming' },
    { id: 3, title: 'Python for Beginners' }
  ];
}
```

courses.component.html

```
<h2>Courses</h2>
<ul>
  <li *ngFor="let c of courses">
    <a [routerLink]="['/course', c.id]">{{ c.title }}</a>
  </li>
</ul>
```

course-details.component.ts

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

```
@Component({
  selector: 'app-course-details',
  templateUrl: './course-details.component.html'
})
export class CourseDetailsComponent {
  courseId: any;

  constructor(private route: ActivatedRoute) {
    this.courseId = this.route.snapshot.paramMap.get('id');
  }
}
```

course-details.component.html

```
<h2>Course Details</h2>
<p>Selected Course ID: {{ courseId }}</p>
```

faculty.component.ts

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

@Component({
  selector: 'app-faculty',
  templateUrl: './faculty.component.html'
})
export class FacultyComponent {
  faculty = [
    { name: 'Prasad Shaha', subject: 'Web Development' },
    { name: 'Shraddha Mam', subject: 'Networking' },
    { name: 'Swati Jadhav', subject: 'Software Testing' }
  ];
}
```

faculty.component.html

```
<h2>Faculty</h2>
<ul>
  <li *ngFor="let f of faculty">
    {{ f.name }} - {{ f.subject }}
  </li>
</ul>
```

contact.component.ts

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

@Component({
  selector: 'app-contact',
  templateUrl: './contact.component.html'
})
export class ContactComponent {}
```

contact.component.html

```
<h2>Contact Us</h2>
<p>Email: portal@example.com</p>
<p>Phone: 9876543210</p>
```

not-found.component.ts

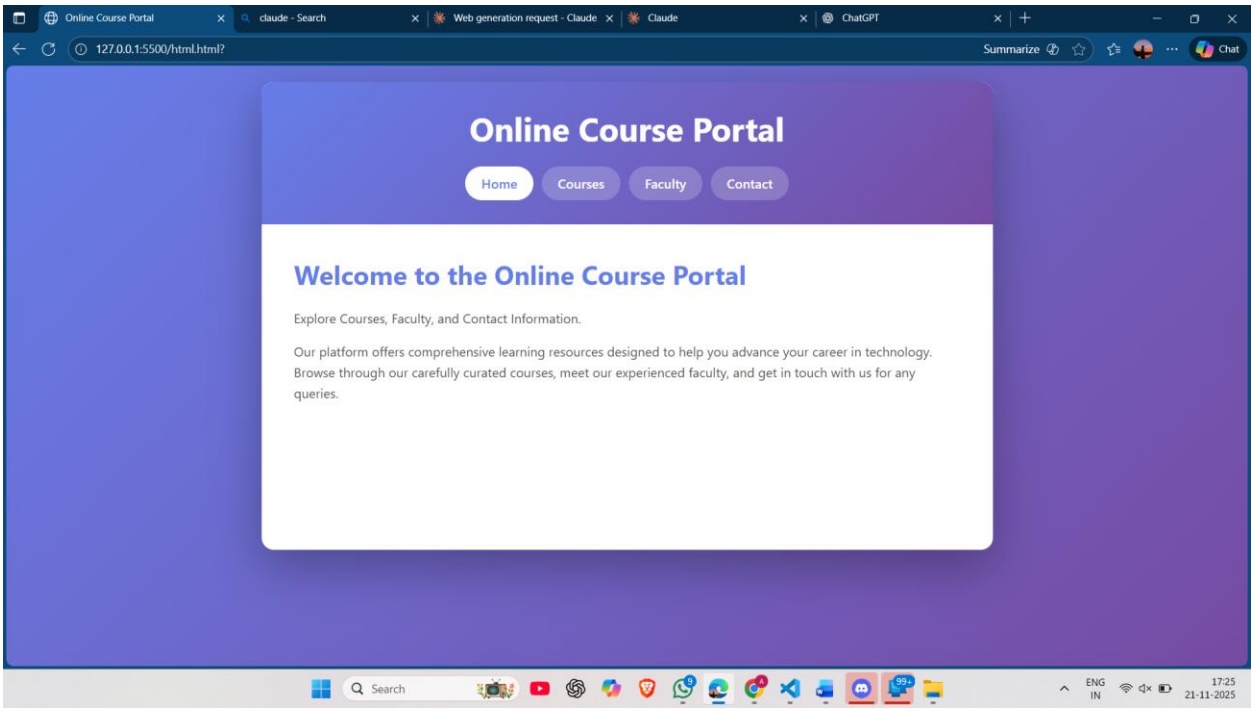
```
import { Component } from '@angular/core';

@Component({
  selector: 'app-notfound',
  templateUrl: './not-found.component.html'
})
export class NotFoundComponent {}
```

not-found.component.html

```
<h2>404 - Page Not Found</h2>
<p>The page you are looking for does not exist.</p>
```

app.component.html

```
<h1>Online Course Portal</h1>
<nav>
  <a routerLink="">Home</a> |
  <a routerLink="courses">Courses</a> |
  <a routerLink="faculty">Faculty</a> |
  <a routerLink="contact">Contact</a>
</nav>
<hr>
<router-outlet></router-outlet>
```



Lab 10-

Capstone Project (Integration of All Technologies)
Case Title: MCA Student Information Portal
Objective: To integrate HTML5, CSS3, JavaScript, TypeScript, and Angular into a unified web project.
Tasks:
• Create a responsive front -end with structured layout.
• Manage student data (CRUD operations) using Angular components and services.
• Implement form validation, data persistence, and navigation.
• Add animations and asynchronous data loading.
Outcome: Mini full -stack front -end project demonstrating complete MCA -level proficiency.

```
import { NgModule, Component, Injectable } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { RouterModule, Routes } from '@angular/router';
import { FormsModule } from '@angular/forms';

@Injectable({ providedIn: 'root' })
export class AuthService {
  currentUser: any = null;

  login(email: string, password: string) {
    if (password.length >= 8) {
      this.currentUser = { email };
```

```
    return true;
   }
   return false;
  }

  isLogged() {
   return this.currentUser !== null;
  }

  logout() {
   this.currentUser = null;
  }
}

// ---------- STUDENT SERVICE ----------
@Injectable({ providedIn: 'root' })
export class StudentService {
  students: any[] = [];

  addStudent(data: any) {
   data.id = Date.now();
   this.students.push(data);
  }

  getStudents() {
   return this.students;
  }

  deleteStudent(id: number) {
   this.students = this.students.filter(s => s.id !== id);
  }
}


@Component({
  selector: 'app-root',
  template: `
  <h1>MCA Student Portal</h1>
  <nav>
   <a routerLink="/">Home</a> |
   <a routerLink="/login">Login</a> |
   <a routerLink="/students">Students</a> |
   <a routerLink="/add-student">Add Student</a>
```

```
  </nav>
  <router-outlet></router-outlet>
  `
})
export class AppComponent {}


@Component({
  selector: 'app-home',
  template: `
    <h2>Welcome to MCA Portal</h2>
    <p>Manage Students · Courses · Records</p>
  `
})
export class HomeComponent {}


@Component({
  selector: 'app-login',
  template: `
  <h2>Login</h2>
  <form (ngSubmit)="login()">
    <input placeholder="Email" [(ngModel)]="email" name="email">
    <input placeholder="Password" type="password" [(ngModel)]="password"
name="password">
    <button>Login</button>
  </form>
  `
})
export class LoginComponent {
  email = ''; password = '';

  constructor(private auth: AuthService, private router: RouterModule) {}

  login() {
    if (this.auth.login(this.email, this.password)) {
      window.location.href = '/students';
    }
  }
}


@Component({
```

```
  selector: 'app-student-list',
  template: `
  <h2>Students</h2>
  <a routerLink="/add-student">Add Student</a>

  <ul>
    <li *ngFor="let s of list">
      {{ s.name }} - {{ s.course }}
      <button (click)="del(s.id)">Delete</button>
    </li>
  </ul>
  `
})
export class StudentListComponent {
  list: any[] = [];

  constructor(private api: StudentService) {
    this.list = this.api.getStudents();
  }

  del(id: number) {
    this.api.deleteStudent(id);
    this.list = this.api.getStudents();
  }
}



@Component({
  selector: 'app-student-form',
  template: `
  <h2>Add Student</h2>
  <form (ngSubmit)="save()">
    <input placeholder="Name" [(ngModel)]="name" name="name">
    <input placeholder="Course" [(ngModel)]="course" name="course">
    <button>Save</button>
  </form>
  `
})
export class StudentFormComponent {
  name = ''; course = '';

  constructor(private api: StudentService) {}
```

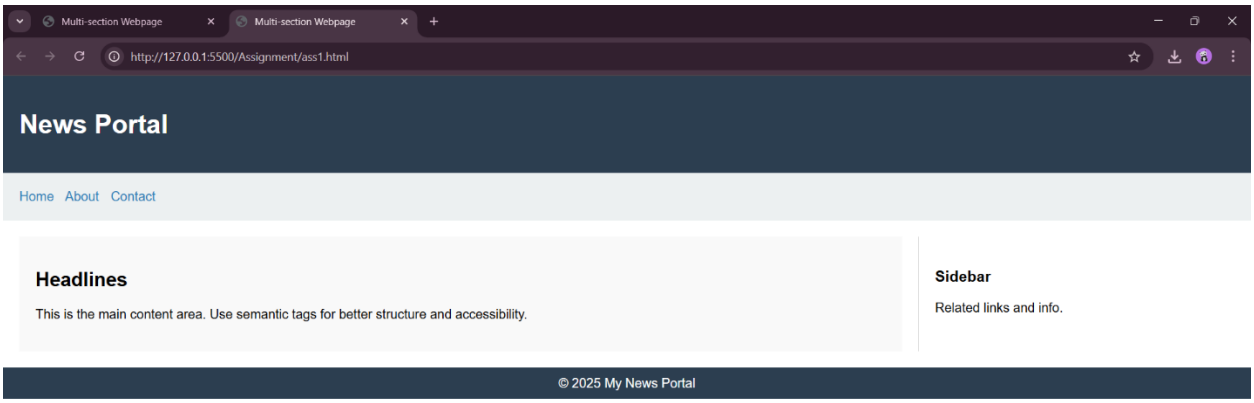
```
  save() {
    this.api.addStudent({ name: this.name, course: this.course });
    window.location.href = '/students';
  }
}


const routes: Routes = [
  { path: '', component: HomeComponent },
  { path: 'login', component: LoginComponent },
  { path: 'students', component: StudentListComponent },
  { path: 'add-student', component: StudentFormComponent }
];


@NgModule({
  declarations: [
    AppComponent,
    HomeComponent,
    LoginComponent,
    StudentListComponent,
    StudentFormComponent
  ],
  imports: [
    BrowserModule,
    FormsModule,
    RouterModule.forRoot(routes)
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule {}
```

EXTRA COMMON PROGRAMS

Q1. Develop a multi-section webpage using <header>,

<nav>, <article>, <aside>, and <footer> tags

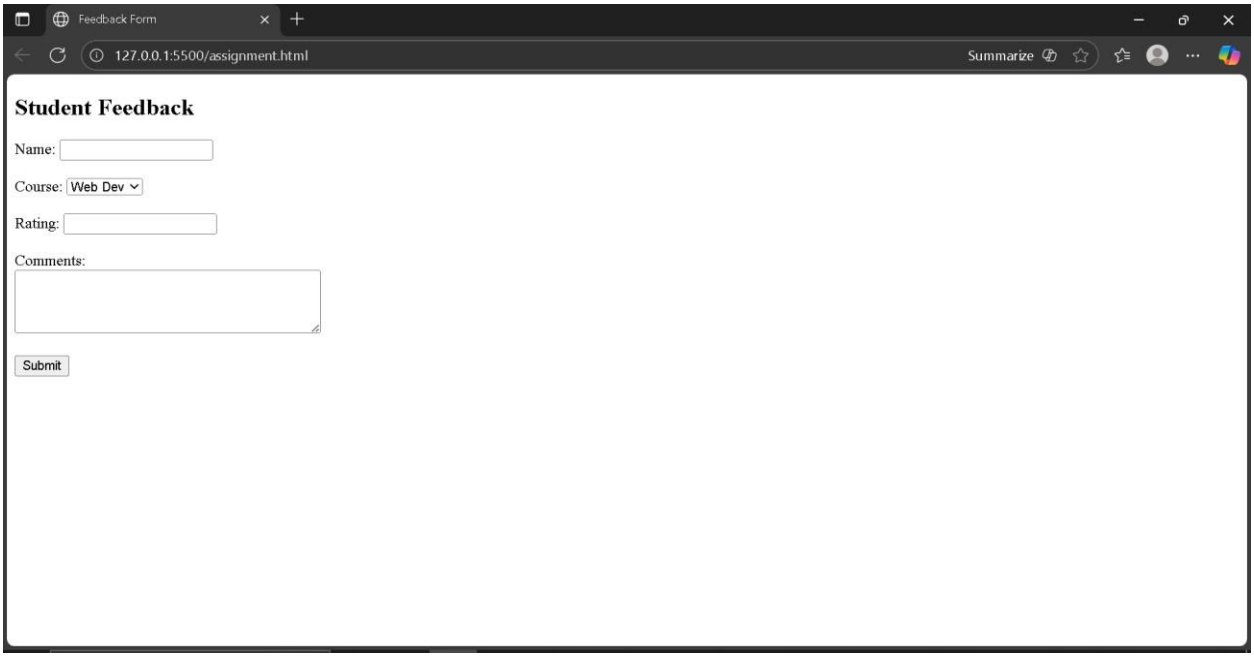
```
<!doctype html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Multi-section Webpage</title>
<style>
body { font-family: Arial, sans-serif; margin:0; } header, nav, article, aside,
footer { padding:20px; } header { background:#2c3e50; color:#fff; }
nav { background:#ecf0f1; }
.container { display:flex; gap:20px; padding:20px; } article { flex:3;
background:#f9f9f9; }
aside { flex:1; background:#fff; border-left:1px solid #ddd; }
footer { background:#2c3e50; color:#fff; text-align:center; padding:10px; }
a { color: #2980b9; text-decoration:none; margin-right:10px; }
</style>
</head>
<body>
<header><h1> News Portal</h1></header>
<nav>
<a href="#">Home</a>
<a href="#">About</a>
<a href="#">Contact</a>
</nav>


<div class="container">
<article>
<h2>Headlines</h2>
<p>This is the main content area. Use semantic tags for better structure and
accessibility.</p>
</article>
<aside>
<h3>Sidebar</h3>
<p>Related links and info.</p>
</aside>
</div>
<footer>© 2025 My News Portal</footer>
</body>
</html>
```



Q2. Create a student feedback form using HTML5 form elements (<input>, <select>, <textarea>, <datalist>)

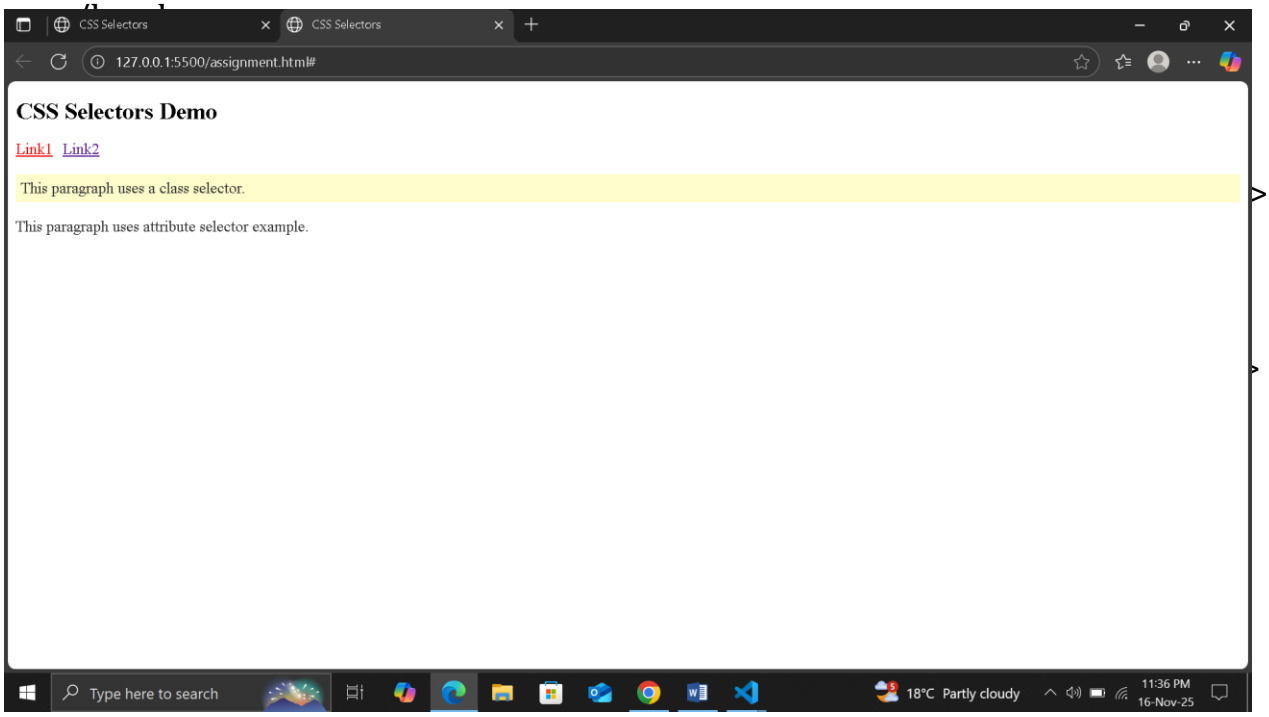
```
<!doctype html>
<html>
<head><meta charset="utf-8"><title>Feedback Form</title></head>
<body>
<h2>Student Feedback</h2>
<form id="feedback">
<label>Name: <input type="text" name="name" required></label><br><br>
<label>Course:
<select name="course">
<option>Web Dev</option><option>Database</option><option>OS</option>
</select>
</label><br><br>
<label>Rating:
<input list="ratings" name="rating">
<datalist id="ratings">
```

```
<option value="Excellent"><option value="Good"><option
value="Average"><option value="Poor">
</datalist>
</label><br><br>
<label>Comments:<br><textarea name="comments" rows="4"
cols="40"></textarea></label><br><br>
<button type="submit">Submit</button>
</form>
<script>
document.getElementById('feedback').addEventListener('submit', function(e){
e.preventDefault();
const fd = new FormData(this);
const obj = Object.fromEntries(fd.entries());
alert('Feedback submitted. Thank you!\\n' + JSON.stringify(obj, null, 2));
});
</script>
</body>
</html>
```



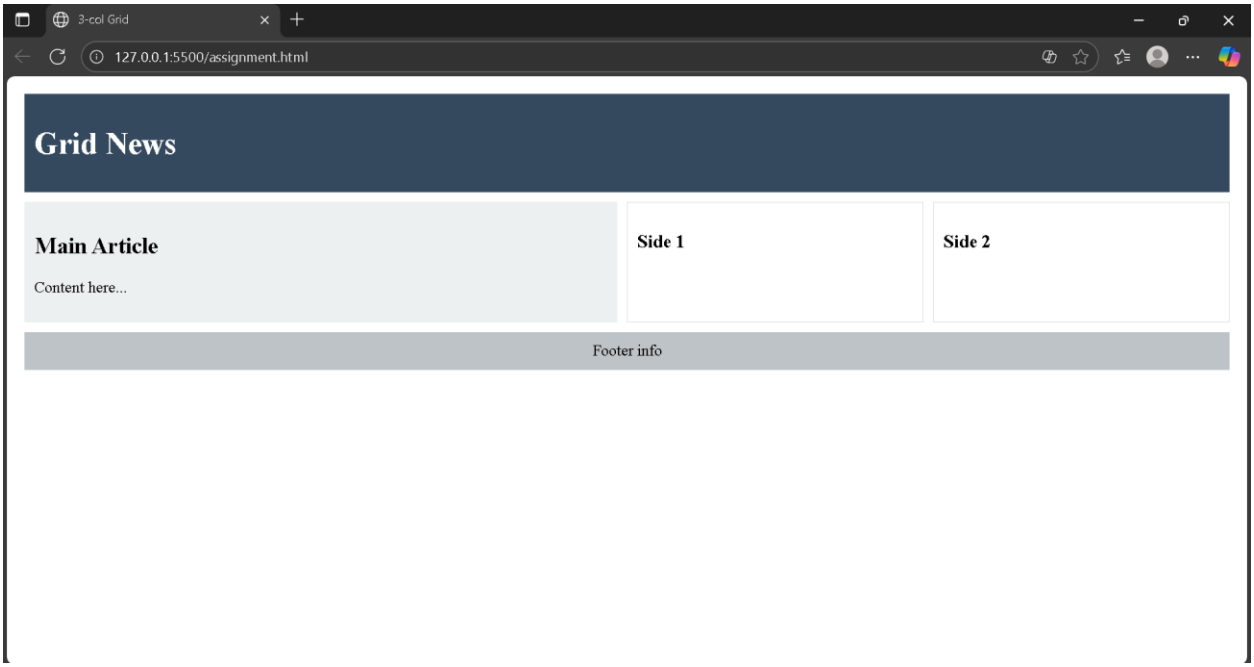
Q3. Apply CSS selectors (element, class, ID, attribute, descendant) to style a webpage differently

```
<!doctype html>
<html>
<head><meta charset="utf-8"><title>CSS Selectors</title>
<style>

/* element */ p { color: #333; }

/* class */ .highlight { background: #fffdcc; padding:5px; }

/* id */ #main-title { font-size:24px; }
/* attribute */ a[target="_blank"] { color: red; }

nav a { margin-right:10px; }

</style>
```



Q4. Build a CSS Grid layout for a 3-column news portal homepage

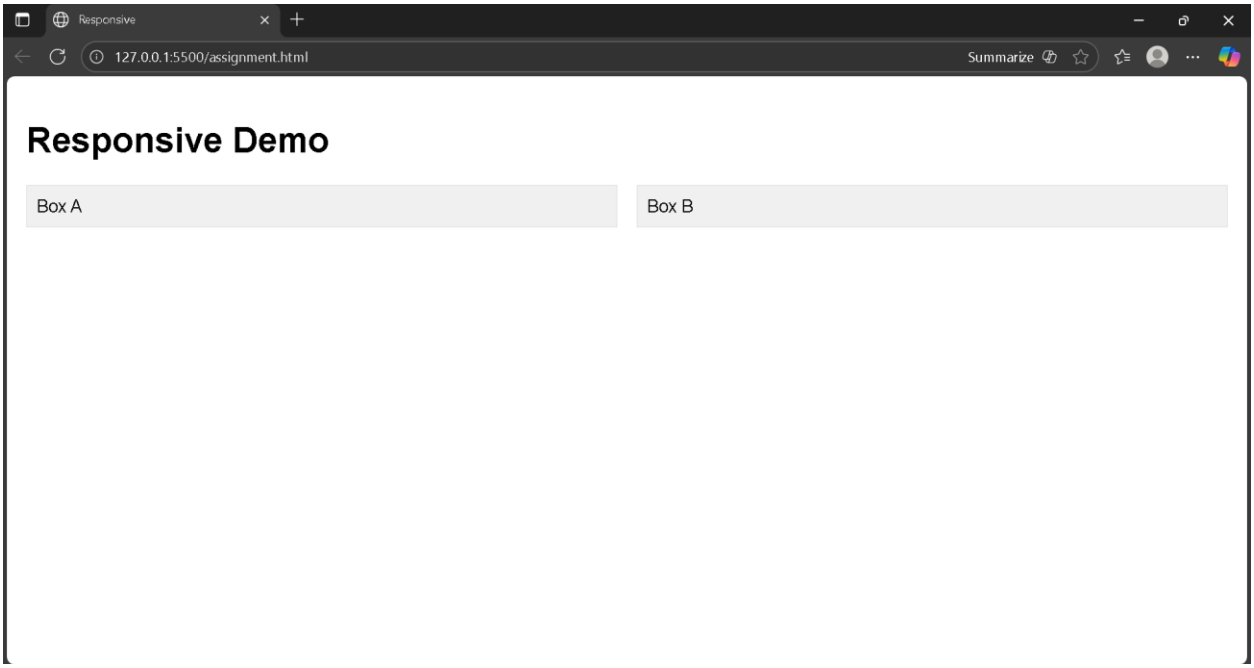
```
<!doctype html>
<html>
<head><meta charset="utf-8"><title>3-col Grid</title>
<style>
.container { display:grid; grid-template-columns: 2fr 1fr 1fr; gap:10px;
padding:10px; }
.header { grid-column:1 / -1; background:#34495e; color:#fff; padding:10px; }
.article { background:#ecf0f1; padding:10px; }

.footer { grid-column:1 / -1; text-align:center; padding:10px;
background:#bdc3c7; }
</style>
</head>

<body>

<div class="container">

<div class="header"><h1>Grid News</h1></div>
<div class="article"><h2>Main Article</h2><p>Content here...</p></div>
<div class="sidebar"><h3>Side 1</h3></div>

<div class="sidebar"><h3>Side 2</h3></div>
<div class="footer">Footer info</div>
</div>
</body>

</html>
```



Q5. Implement media queries to change font sizes and layout based on device width

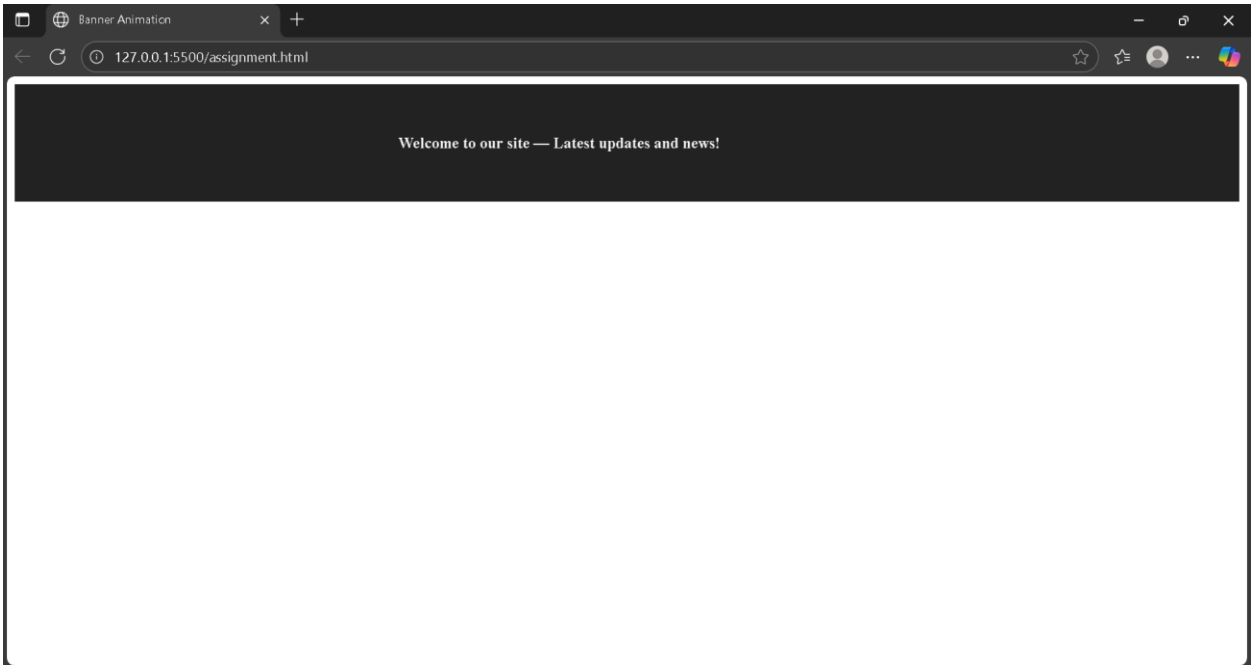
```
<!doctype html>
<html>
<head><meta charset="utf-8"><title>Responsive</title>
<style>
body { font-family: Arial; padding:20px; margin:0; }

.container { display:flex; gap:20px; }
.box { flex:1; padding:10px; background:#f0f0f0; border:1px solid #ddd; }
@media (max-width:600px){ body{font-size:14px;}
.container{flex-direction:column;}

}
```

```
@media (min-width:601px){ body{font-size:18px;}
}
</style>

</head>

<body>

<h1>Responsive Demo</h1>
<div class="container">

<div class="box">Box A</div>

<div class="box">Box B</div>

</div>

</body>
</html>
```



Q6. Design a banner animation using @keyframes and CSS3 animations

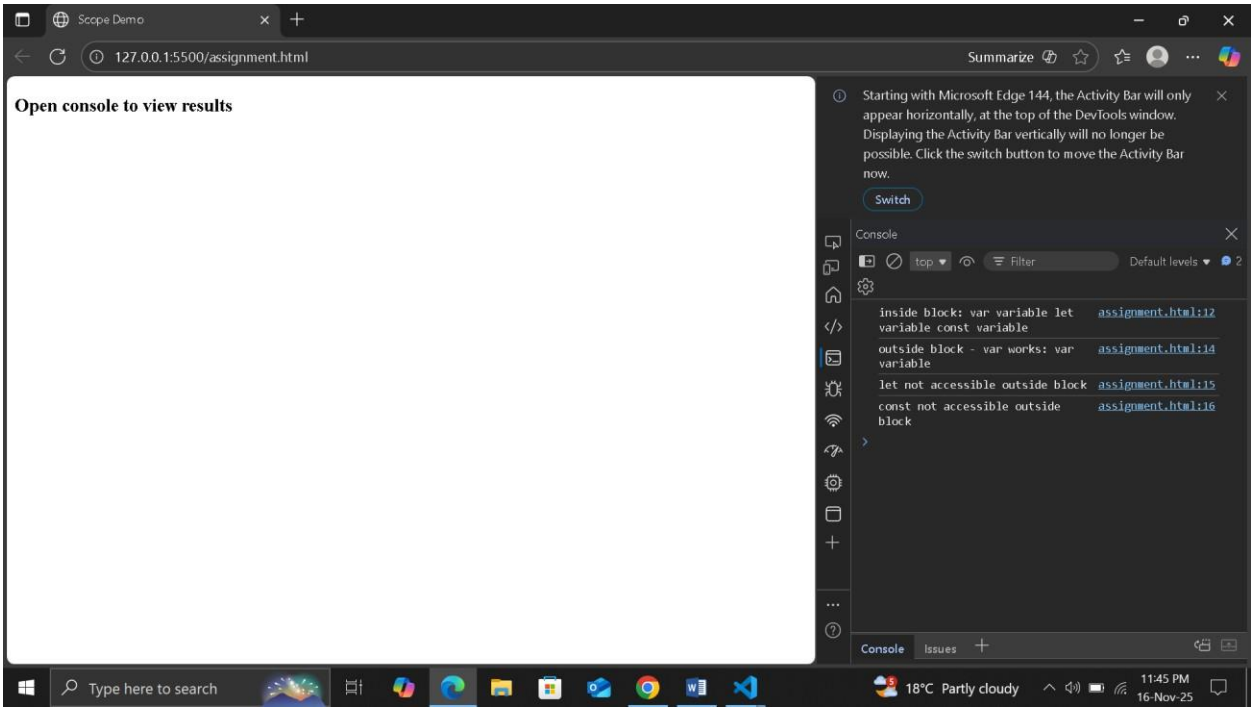
```
<!doctype html>
<html>

<head><meta charset="utf-8"><title>Banner Animation</title>

<style>

.banner { width:100%; height:120px; display:flex; align-items:center; justify-content:center;  background:#222;  color:#fff; overflow:hidden; }
@keyframes slide { 0% {transform:translateX(100%);} 100% {transform:translateX(-100%);} }

.marquee { display:inline-block; white-space:nowrap; animation:slide 8s linear infinite; font-weight:bold; }
</style>

</head>

<body>

<div class="banner"><div class="marquee">Welcome to our site — Latest updates and news!</div></div>
</body>

</html>
```



Q7. Demonstrate block scope vs function scope using let, var, and const

```
<!doctype html>

<html>
<head><meta charset="utf-8"><title>Scope Demo</title></head>

<body>

<h3>Open console to view results</h3>

<script>

function testScope(){ if(true){
var a = 'var variable'; let b = 'let variable';
```

```
const c = 'const variable'; console.log('inside block:', a, b, c);
}
console.log('outside block - var works:', a); try{ console.log('outside block -
let:',
;}catch(e){ console.log('let not accessible outside block'); }

try{ console.log('outside block - const:',
;}catch(e){ console.log('const not accessible outside block'); }

}

testScope();

</script>

</body></html>
```



Q8. Use JavaScript objects and arrays to store and display student record

```
<!doctype html>

<html>

<head><meta charset="utf-8"><title>Students</title></head>
<body>
```

```
<h2>Students List</h2>

<ul id="list"></ul>

<script>
const students = [

{name:’vijay’, roll:1, marks:85},

{name:'raj', roll:2, marks:78},

{name:'kale', roll:3, marks:92}

];
const ul = document.getElementById('list'); students.forEach(s => {
const li = document.createElement('li'); li.textContent = `${s.roll} - ${s.name}
(${s.marks})`; ul.appendChild(li);
});

</script>

</body>

</html>
```

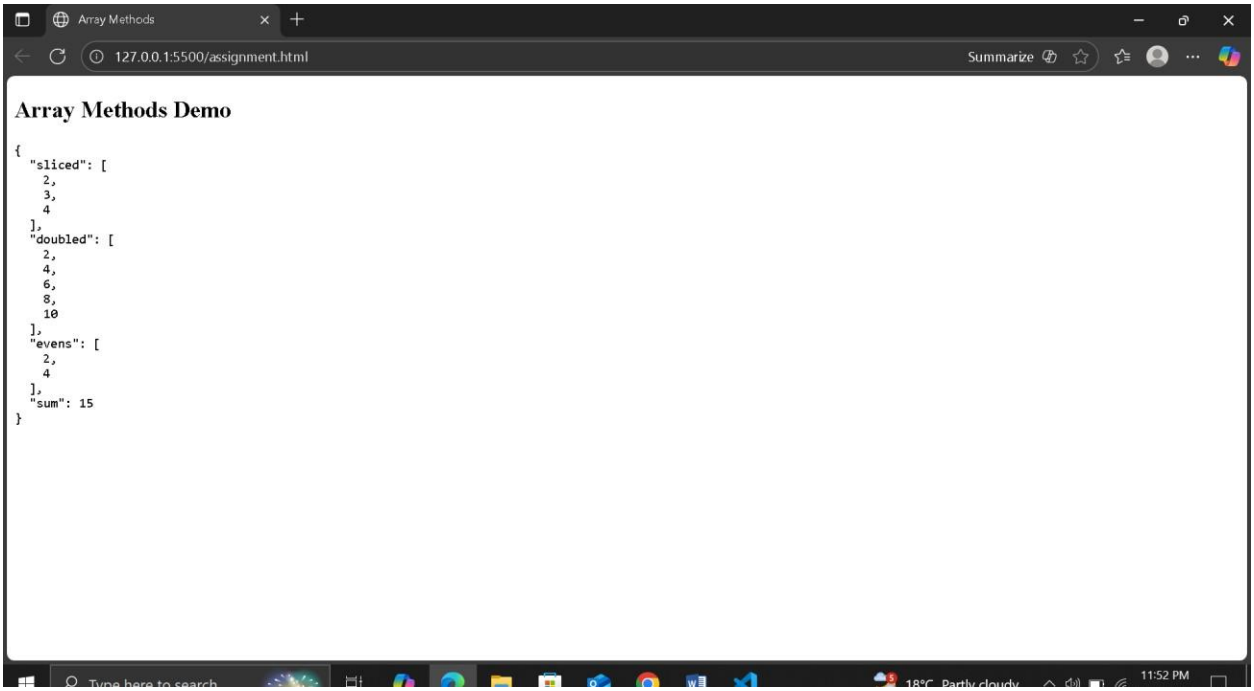
Students List

- 1 - vijay (85)
- 2 - raj (78)
- 3 - kale (92)

Q9. Implement string and array methods (slice, map, filter, reduce) on sample data

```
<!doctype html>

<html>
<head><meta charset="utf-8"><title>Array Methods</title></head>

<body>

<h2>Array Methods Demo</h2>

<pre id="out"></pre>

<script>
const  nums  = [1,2,3,4,5]; const sliced = nums.slice(1,4);
const doubled = nums.map(n => n*2);
```

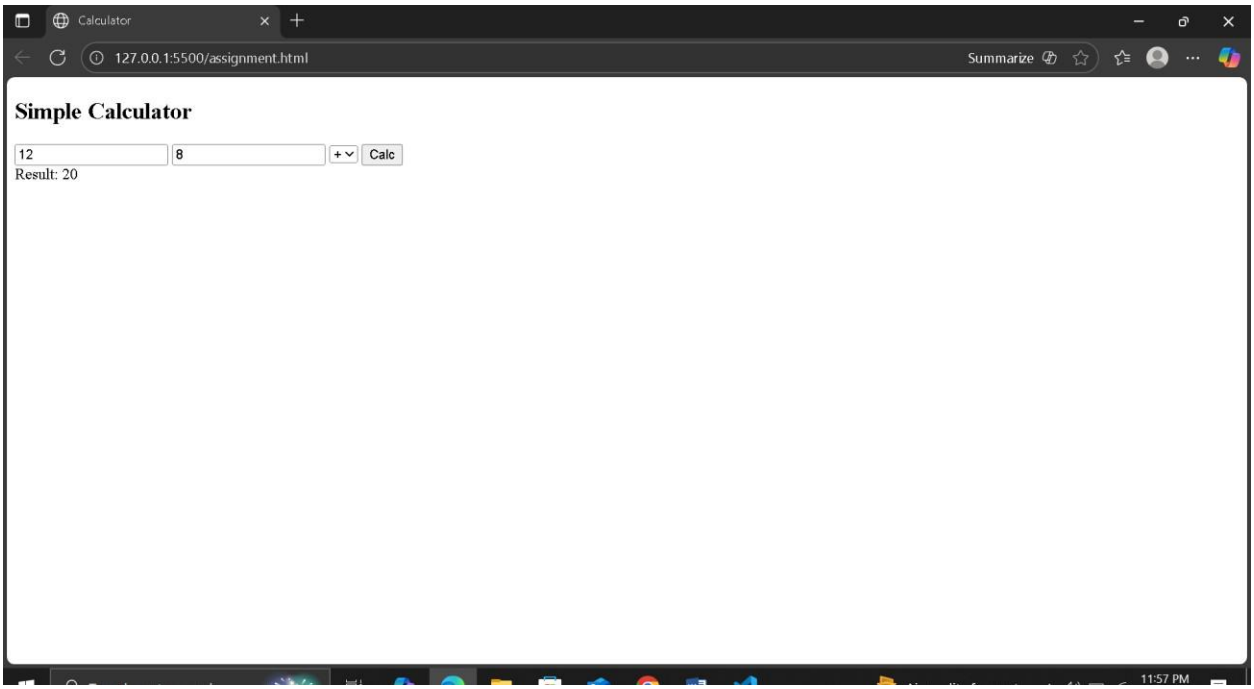
```
const evens = nums.filter(n => n%2===0); const sum =
nums.reduce((a,b)=>a+b,0);
document.getElementById('out').textContent =
JSON.stringify({sliced,doubled,evens,sum}, null, 2);
</script>

</body>

</html>
```



Q10. Create a simple calculator using basic JS operators and functions

```
<!doctype html>

<html>
<head><meta charset="utf-8"><title>Calculator</title></head>

<body>
```

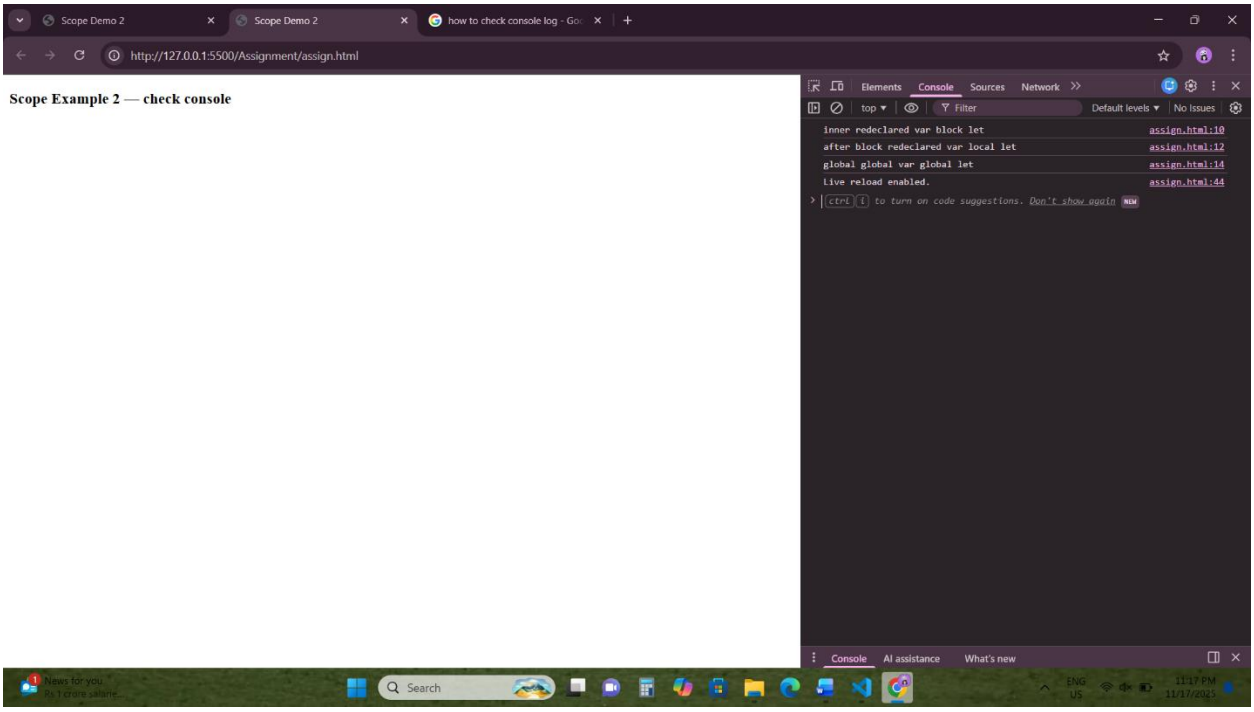
```
<h2>Simple Calculator</h2>

<input id="a" type="number" placeholder="a">

<input id="b" type="number" placeholder="b">
<select id="op">

<option value="+">+</option><option value="-">-</option><option
value="*">*</option><option value="/">/</option>
</select>
<button id="go">Calc</button>

<div id="res"></div>

<script>

document.getElementById('go').addEventListener('click', ()=>{

const a=+document.getElementById('a').value,
b=+document.getElementById('b').value,
op=document.getElementById('op').value;
let r;

if(op=='+') r=a+b; else if(op=='-') r=a-b; else if(op=='*') r=a*b; else r=(b!==0?
a/b : 'Error');
document.getElementById('res').textContent = 'Result: ' + r;

});

</script>

</body>
</html>
```



Q11. Demonstrate block scope vs function scope using let, var, and const (alternate example)

```
<!doctype html>
<html>
<head><meta charset="utf-8"><title>Scope Demo 2</title></head>
<body>
<h3>Scope Example 2 — check console</h3>
<script>
var x = 'global var'; let y = 'global let'; function demo(){
var x = 'local var'; let y = 'local let'; if(true){
```

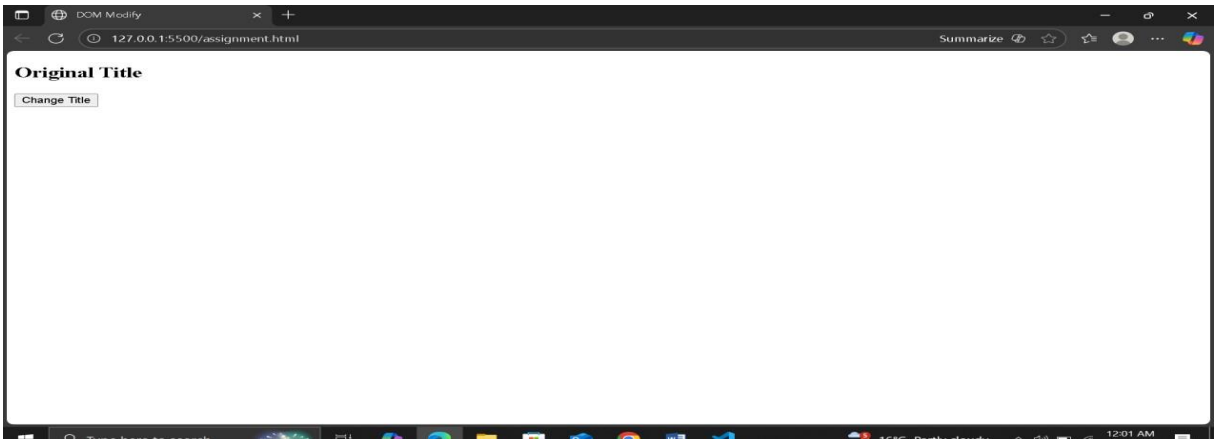
```
var x = 'redeclared var'; let y = 'block let'; console.log('inner', x, y);
}
console.log('after block', x, y);
}
demo(); console.log('global', x, y);
</script>
</body>
</html>
```

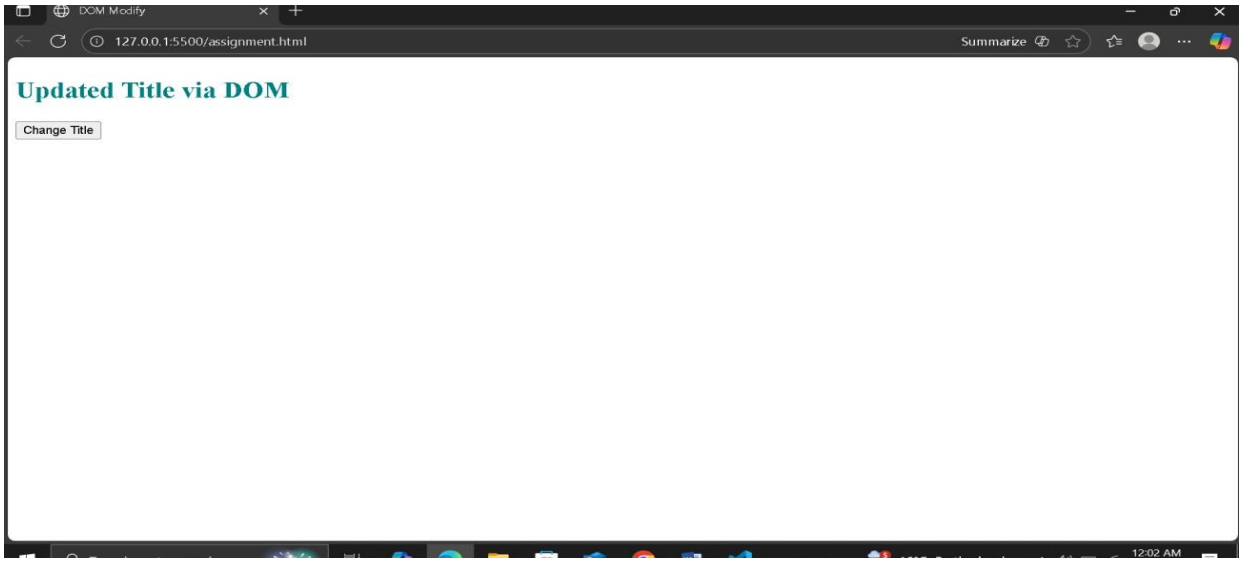


Q12. Access and modify the text content and style of HTML elements using DOM methods

```
<!doctype html>
<htm>
<head>
<meta charset="utf-8"><title>DOM Modify</title></head>
<body>
<h2 id="title">Original Title</h2>
<button id="btn">Change Title</button>
```

```
<script> document.getElementById('btn').addEventListener('click', ()=>{
const t = document.getElementById('title'); t.textContent = 'Updated Title via
DOM'; t.style.color = 'teal';
t.style.fontSize = '28px';
});
</script>
</body>
</html>
```

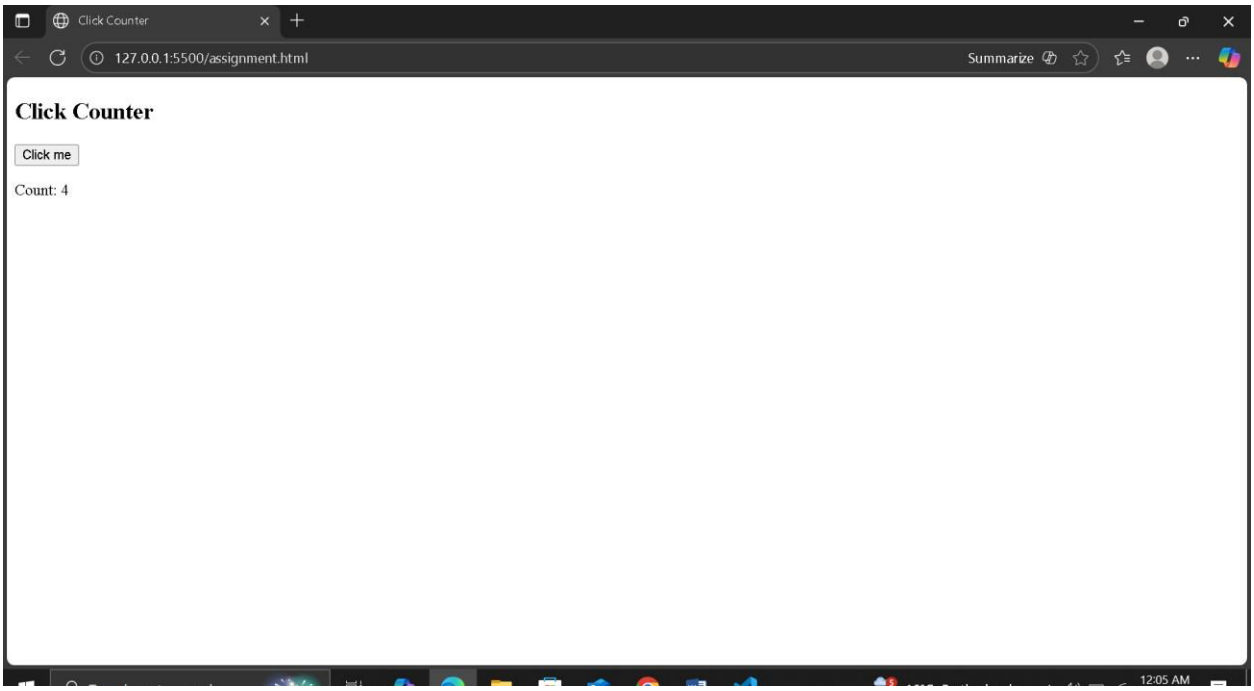




Q13. Create a click counter using event listeners (addEventListener)

```
<!doctype html>
<html>
<head><meta charset="utf-8"><title>Click Counter</title></head>
<body>
<h2>Click Counter</h2>
<button id="btn">Click me</button>
```

```
<p>Count: <span id="c">0</span></p>
<script> let c=0;
document.getElementById('btn').addEventListener('click', ()=>
document.getElementById('c').textContent = ++c);
</script>
</body>
</html>
```



Q14. Implement a to-do list app where users can add and delete tasks dynamically

```
<!doctype html>
<html>
<head><meta charset="utf-8"><title>Todo App</title>
<style>.done{text-decoration:line-through; color:gray;}</style>
</head>
<body>
<h2>ToDo List</h2>
<input id="task" placeholder="New task"><button id="add">Add</button>
<ul id="list"></ul>
<script> document.getElementById('add').addEventListener('click', ()=>{
const v = document.getElementById('task').value.trim(); if(!v) return;
const li = document.createElement('li'); li.textContent = v + ' ';
const del = document.createElement('button'); del.textContent='Delete';
del.onclick = ()=> li.remove();
li.onclick = (e)=> { if(e.target === li) li.classList.toggle('done'); };
li.appendChild(del); document.getElementById('list').appendChild(li);
document.getElementById('task').value='';
});
</script>
</body>
</html>
```

ToDo List

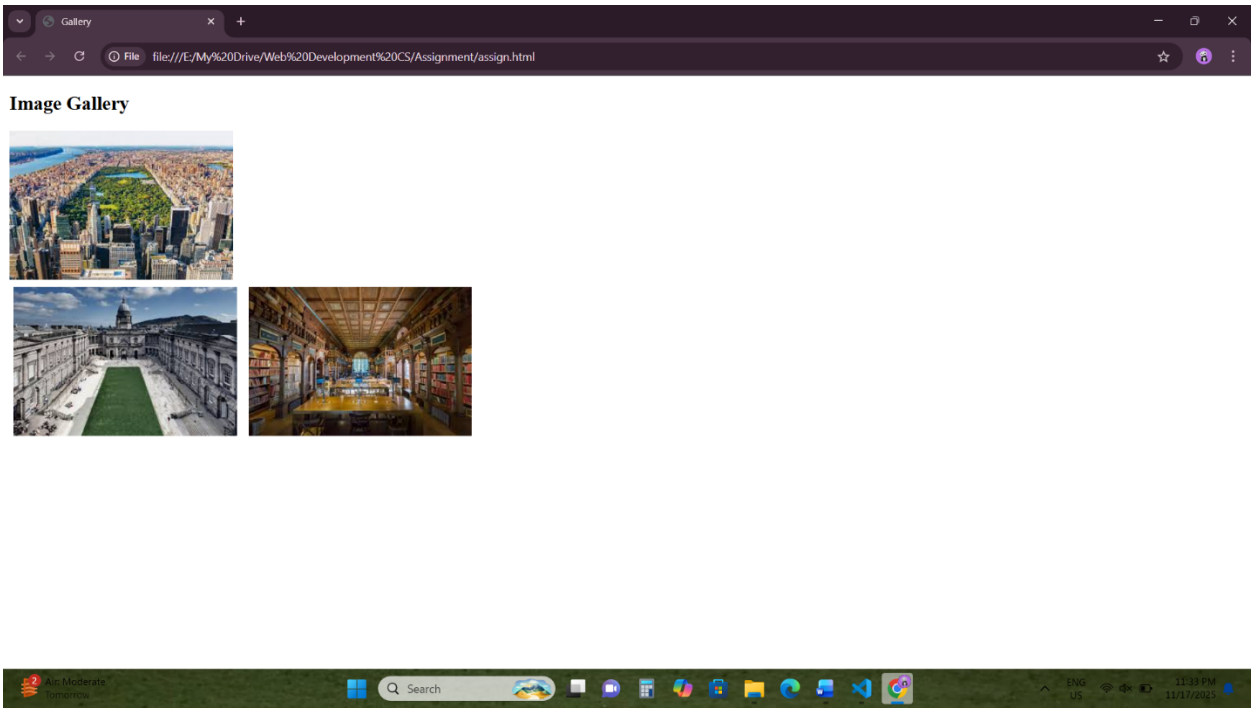
New task Add

To-Do List

New task Add

Q15. Build an image gallery where clicking a thumbnail changes the main displayed image

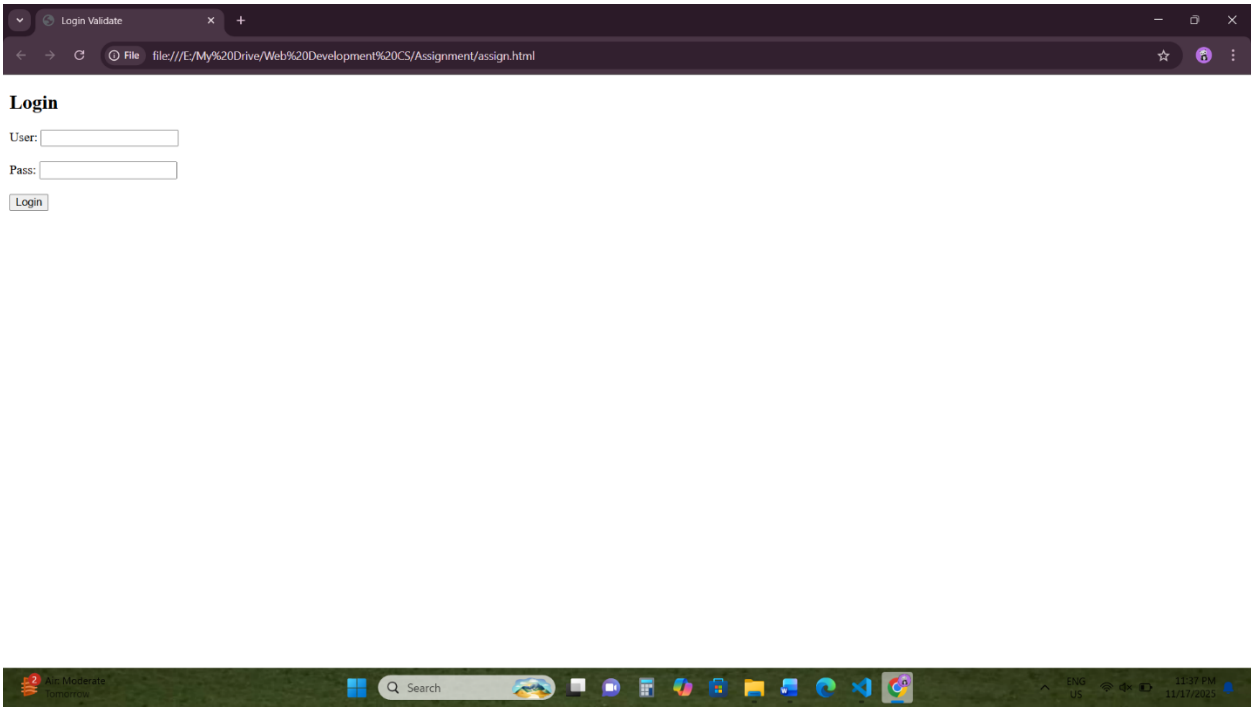
```
<!doctype html>
<html>
<head><meta charset="utf-8"><title>Gallery</title></head>
<body>
<h2>Image Gallery</h2>
<img id="main" src="E:\My Drive\Web Development CS\download (1).jpeg"
alt="main"><br>
<img class="thumb" src="E:\My Drive\Web Development
CS\Assignment\download (2).jpeg" style="cursor:pointer;margin:5px;">
<img class="thumb" src="E:\My Drive\Web Development
CS\Assignment\download (3).jpeg" style="cursor:pointer;margin:5px;">
<script> document.querySelectorAll('.thumb').forEach(t=>{
t.addEventListener('click', ()=> document.getElementById('main').src
= t.src.replace('100x60','400x200'));
});
</script>
</body>
</html>
```



Q16. Validate a login form using DOM events and display inline error messages

```
<!doctype html>
<html>
<head><meta charset="utf-8"><title>Login Validate</title></head>
<body>
<h2>Login</h2>
<form id="login">
<label>User: <input id="user" required></label><span id="uerr"
style="color:red"></span><br><br>
<label>Pass: <input id="pass" type="password" required></label><span
id="perr" style="color:red"></span><br><br>
<button type="submit">Login</button>
</form>
<script>
document.getElementById('login').addEventListener('submit', function(e){
e.preventDefault();
const u=document.getElementById('user').value,
p=document.getElementById('pass').value;
let ok=true;
document.getElementById('uerr').textContent='';
document.getElementById('perr').textContent='';
if(u.length<3){document.getElementById('uerr').textContent=' Username too
short'; ok=false; }
```

```
if(p.length<6){document.getElementById('perr').textContent=' Password must be 6+ chars'; ok=false; }
if(ok) alert('Logged in (demo)');
});
</script>
</body>
</html>
```



Q17. Write a program using Promises to simulate a delayed API response (e.g., "Loading user data...")

```
<!doctype html>
<html>
<head>
   <meta charset="utf-8">
   <title>Promises</title>
</head>
<body>

<h2>Promise Demo</h2>
<div id="status">Click to load</div>
<button id="load">Load User</button>

<script>

function fakeApi(){
   return new Promise((res, rej) => {
      setTimeout(() => res({ name: 'Nilam', id: 101 }), 1500);
   });
}

document.getElementById('load').addEventListener('click', async () => {
   document.getElementById('status').textContent = 'Loading user data...';

   const user = await fakeApi();

   document.getElementById('status').textContent = 'Loaded: ' + user.name;
});

</script>

</body>
</html>
```

Q18. Convert the above Promise code into async/await syntax

```
<!doctype html>
<html>
<head><meta charset="utf-8"><title>Async/Await</title></head>
<body>
<h2>Async/Await Demo</h2>
<div id="status">Click to load</div>
<button id="load">Load User</button>
<script>
function fakeApi(){ return new Promise(res =>
```

```
setTimeout(()=>res({name:'vijay'}),12111)); }
document.getElementById('load').addEventListener('click', async ()=>{
document.getElementById('status').textContent =  'Loading...'; const
 user = await fakeApi();
document.getElementById('status').textContent = 'User: ' + user.name;
});
</script>
</body>

</html>
```

19. Create a class Student with methods to calculate grades and display info

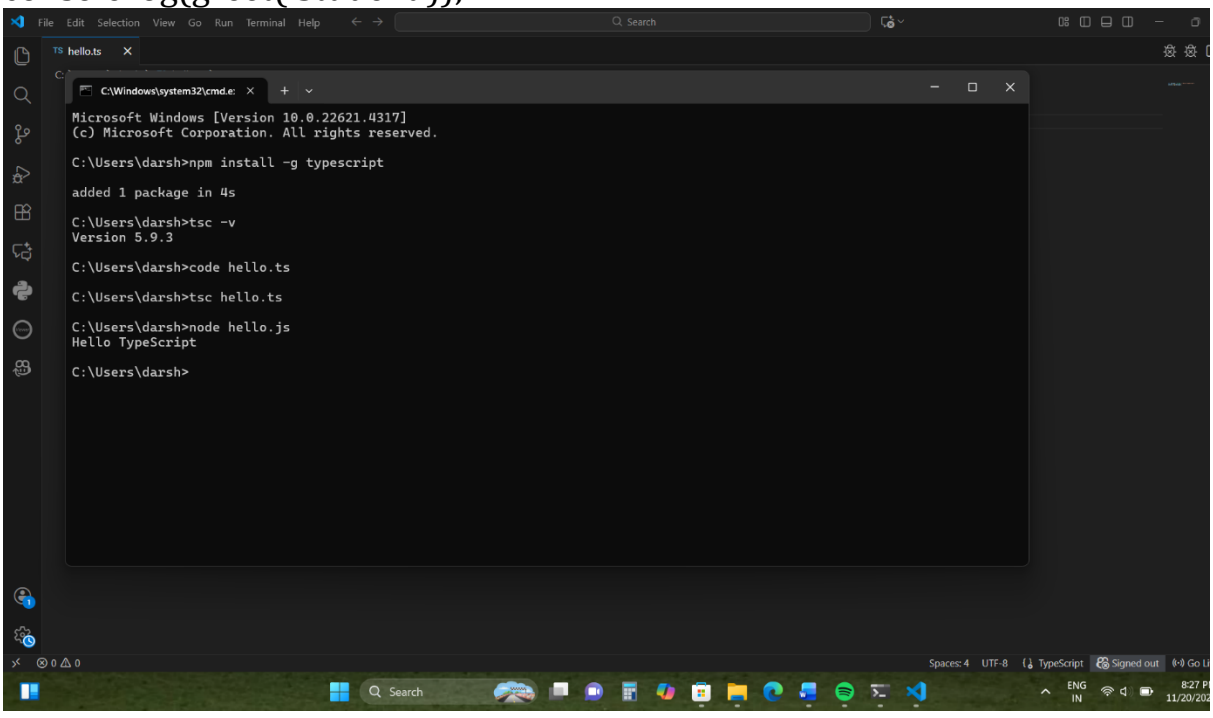
```
<!doctype html>
<html>
<head><meta charset="utf-8"><title>Student Class</title></head>
<body>
<h2>Student Class</h2>
<pre id="out"></pre>
<script>
class Student {
constructor(name, marks){ this.name=name; this.marks=marks; } avg(){
return this.marks.reduce((a,b)=>a+b,0)/this.marks.length; }
grade(){ const a=this.avg(); return a>=75?'A':a>=60?'B':a>=40?'C':'F'; }
info(){ return {name:this.name, avg:this.avg().toFixed(2), grade:this.grade()}; }
}
const s = new Student('vijay',[88,82,91]);
document.getElementById('out').textContent = JSON.stringify(s.info(), null, 2);
</script>
</body>
</html>
```

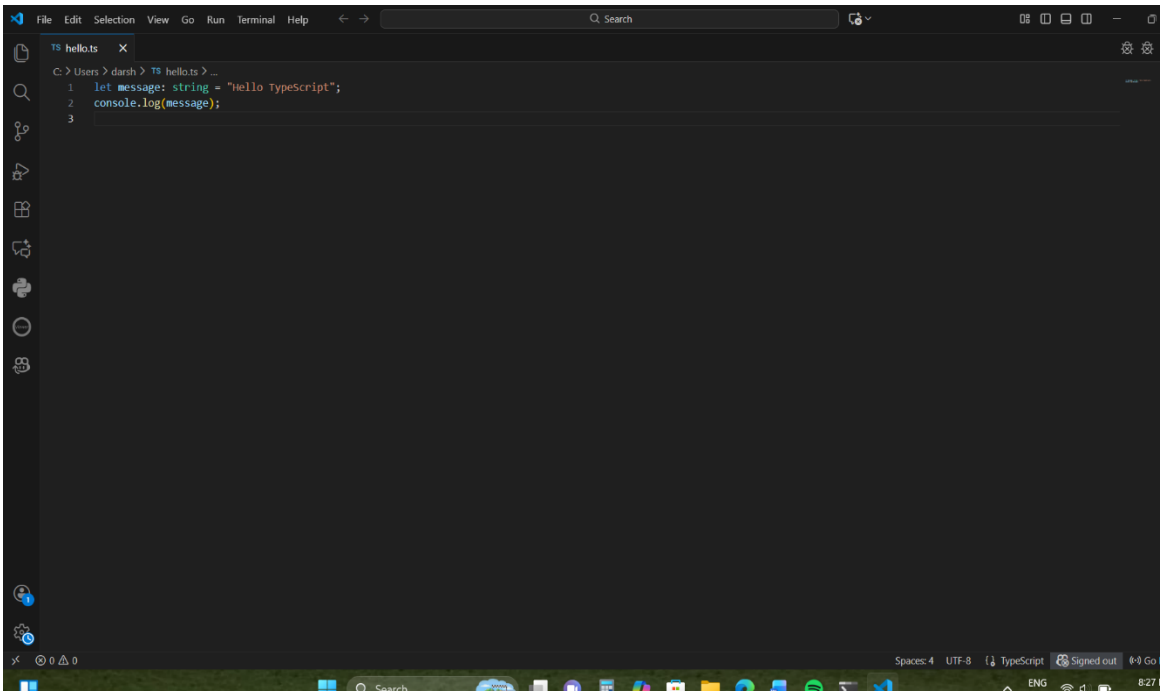
Student Class


```
{
  "name": "vijay",
  "avg": "87.00",
  "grade": "A"
}
```

Q20. Install TypeScript and write a "Hello TypeScript" program using tsc

```
// hello.ts
// To compile: tsc hello.ts
const greet = (name: string): string => `Hello, ${name} (from TypeScript)`;
console.log(greet('Student'));
```





Q21. Demonstrate type annotations for variables, arrays, and functions (types.ts)

```
// variable type annotations
let studentName: string = "Alex";
let age: number = 20;
let isPassed: boolean = true;

// array type annotations
let marks: number[] = [85, 90, 78];
let subjects: string[] = ["Java", "Python", "DBMS"];

// function type annotations
function add(a: number, b: number): number {
  return a + b;
}

function greet(name: string): string {
  return "Hello " + name;
}

// function call outputs
console.log(studentName);
```

```
console.log(age);
console.log(isPassed);
console.log(marks);
console.log(subjects);
console.log(add(10, 20));
console.log(greet("Alex"));
```

OUTPUT

```
Alex
20
true
[ 85, 90, 78 ]
[ 'Java', 'Python', 'DBMS' ]
30
Hello Alex
```

Q22. Write a small app that uses TypeScript classes and modules to manage a student list

student.ts — Class Module

```
// Student.ts
export class Student {
  id: number;
  name: string;
  course: string;

  constructor(id: number, name: string, course: string) {
    this.id = id;
    this.name = name;
    this.course = course;
  }

  getInfo(): string {
    return `ID: ${this.id}, Name: ${this.name}, Course: ${this.course}`;
  }
}
```

StudentManager.ts — Manager Module

```
// StudentManager.ts
import { Student } from "./Student";

export class StudentManager {
  private students: Student[] = [];

  addStudent(student: Student): void {
    this.students.push(student);
  }

  removeStudent(id: number): void {
    this.students = this.students.filter(s => s.id !== id);
  }

  listStudents(): void {
    console.log("---- Student List ----");
    this.students.forEach(s => console.log(s.getInfo()));
  }
}
```

app.ts — Main Application File

```
// app.ts
import { Student } from "./Student";
import { StudentManager } from "./StudentManager";

let manager = new StudentManager();

// Adding students
manager.addStudent(new Student(1, "Ravi", "BCA"));
manager.addStudent(new Student(2, "Priya", "MCA"));
manager.addStudent(new Student(3, "Arjun", "B.Tech"));

// Show list
manager.listStudents();

// Remove a student
manager.removeStudent(2);

console.log("\nAfter Removal:");
manager.listStudents();

//Output
```

---- Student List ----
ID: 1, Name: vijay, Course: BCA
ID: 2, Name: jay, Course: MCA
ID: 3, Name: Arjun, Course: B.Tech

After Removal:
---- Student List ----
ID: 1, Name: jay, Course: BCA
ID: 3, Name: Arjun, Course: B.Tech

Q23. Set up a new Angular project using Angular CLI (commands)

Step 1: Install Node.js
To work with Angular, Node.js must be installed.
I downloaded and installed **Node.js (LTS Version)** from the official website.
Step 2: Verify Node.js and npm Installation
After installation, I checked the versions using:
node -v
npm -v
Step 3: Install Angular CLI
Angular CLI (Command Line Interface) is installed using npm:
npm install -g @angular/cli
Step 4: Create a New Angular Project
I created a project using the following command:
ng new my-angular-app
During project creation, I selected:

- **Routing:** Yes
- **Stylesheet:** CSS

Step 5: Open the Project Folder
cd my-angular-app
Step 6: Run the Angular Application
To start the development server:
ng serve --open
This automatically opens the browser.
Step 7: View Output
The Angular app runs at:
http://localhost:4200/
The default Angular welcome page is displayed.

Q24. Create a component to display a welcome message and student information (welcome.component.ts)

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

@Component({
  selector: 'app-student-info',
  templateUrl: './student-info.component.html',
  styleUrls: ['./student-info.component.css']
})
export class StudentInfoComponent {
  welcomeMessage = "Welcome to the Student Portal";

  studentName = "Nilam";
  studentCourse = "MCA";
  studentRoll = B18;
}
```

student-info.component.html

html

Copy code

```
<h2>{{ welcomeMessage }}</h2>

<p>Name: {{ studentName }}</p>
<p>Course: {{ studentCourse }}</p>
<p>Roll No: {{ studentRoll }}</p>
```

student-info.component.css

css

Copy code

```
h2 {
  color: blue;
}
p {
  font-size: 16px;
}
```

app.component.html

html

Copy code

```
<app-student-info></app-student-info>
```

25. Implement two-way data binding for form inputs using [(ngModel)]

(Angular template)

FILE: app.module.ts

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { FormsModule } from '@angular/forms';
import { AppComponent } from './app.component';

@NgModule({
  declarations: [AppComponent],
  imports: [BrowserModule, FormsModule],
  bootstrap: [AppComponent]
})
export class AppModule {}
```

FILE: app.component.ts

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

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html'
})
export class AppComponent {
  studentName = "";
}
```

FILE: app.component.html

```
<h2>Two-Way Data Binding Demo</h2>

<label>Enter Student Name:</label>
<input type="text" [(ngModel)]="studentName">

<p>You Typed: {{ studentName }}</p>
```

Q26. Create a custom component for reusable cards (e.g., student or product card) (card.component.ts)

1. student-card.component.ts

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

@Component({
```

```
  selector: 'app-student-card',
  template: `<div><h4>{{name}}</h4><p>{{course}}</p></div>`
})
export class StudentCardComponent {
  @Input() name = "";
  @Input() course = "";
}
```

---

2. app.component.ts

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

@Component({
  selector: 'app-root',
  template: `
    <app-student-card name="Snehal" course="MCA"></app-student-card>
    <app-student-card name="Rohit" course="BCA"></app-student-card>
  `
})
export class AppComponent {}
```

---

3. app.module.ts

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import { StudentCardComponent } from './student-card.component';

@NgModule({
  declarations: [AppComponent, StudentCardComponent],
  imports: [BrowserModule],
  bootstrap: [AppComponent]
})
export class AppModule {}
```