1)	Definition :	Personal Portfolio (HTML5 + CSS3)
	Programming concept :	Build a single-page portfolio with sections: About, Skills, Projects, Contact. Requirements: Semantic HTML, responsive layout (flex/grid), a contact (no backend yet), accessible nav.
	Code:	**Index.html:-**

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8" />
 <meta name="viewport" content="width=device-width, initial-scale=1.0" />
 <meta name="description" content="Personal Portfolio of [Your Name]" />
 <title>My Portfolio</title>
 <link rel="stylesheet" href="styles.css" />
</head>
<body>

 <header>
  <nav class="navbar">
   <h1 class="logo">MyPortfolio</h1>
   <ul class="nav-links">
    <li><a href="#about">About</a></li>
    <li><a href="#skills">Skills</a></li>
    <li><a href="#projects">Projects</a></li>
    <li><a href="#contact">Contact</a></li>
   </ul>
  </nav>
 </header>

 <section id="about" class="about">
  <img src="assets/profile.jpg" alt="Profile photo" class="profile-img" />
  <div class="about-text">
   <h2>About Me</h2>
   <p>
    Hello! I'm <strong>Prince Patel</strong>, a passionate web developer who loves crafting modern and responsive websites using HTML5, CSS3, and JavaScript.
   </p>
  </div>
 </section>

 <section id="skills" class="skills">
```

```
  <h2>Skills</h2>
  <div class="skills-grid">
   <div class="skill-card">HTML5</div>
   <div class="skill-card">CSS3</div>
   <div class="skill-card">JavaScript</div>
   <div class="skill-card">Java</div>
   <div class="skill-card">Git & GitHub</div>
   <div class="skill-card">C++</div>
   <div class="skill-card">C</div>
  </div>
 </section>

 <section id="projects" class="projects">
  <h2>Projects</h2>
  <div class="project-grid">
   <div class="project-card">
    <img src="assets/project1.jpg" alt="Project 1 Screenshot" />
    <h3>Portfolio Website</h3>
    <p>Personal responsive portfolio showcasing my work.</p>
   </div>
   <!-- <div class="project-card">
    <img src="assets/project2.jpg" alt="Project 2 Screenshot" />
    <h3>To-Do App</h3>
    <p>Simple task manager built with JavaScript.</p>
   </div>
   <div class="project-card">
    <img src="assets/project3.jpg" alt="Project 3 Screenshot" />
    <h3>Landing Page</h3>
    <p>Modern landing page using Flexbox and Grid.</p>
   </div> -->
  </div>
 </section>

 <section id="contact" class="contact">
  <h2>Contact Me</h2>
  <form action="#" method="post">
   <label for="name">Name:</label>
   <input type="text" id="name" name="name" required />

   <label for="email">Email:</label>
   <input type="email" id="email" name="email" required />

   <label for="message">Message:</label>
   <textarea id="message" name="message" rows="5"
required></textarea>

   <button type="submit">Send Message</button>
```

```
    </form>
  </section>

  <footer>
    <p>&copy; 2025 Prince Patel. All rights reserved.</p>
  </footer>

</body>
</html>
```

Style.Css:-

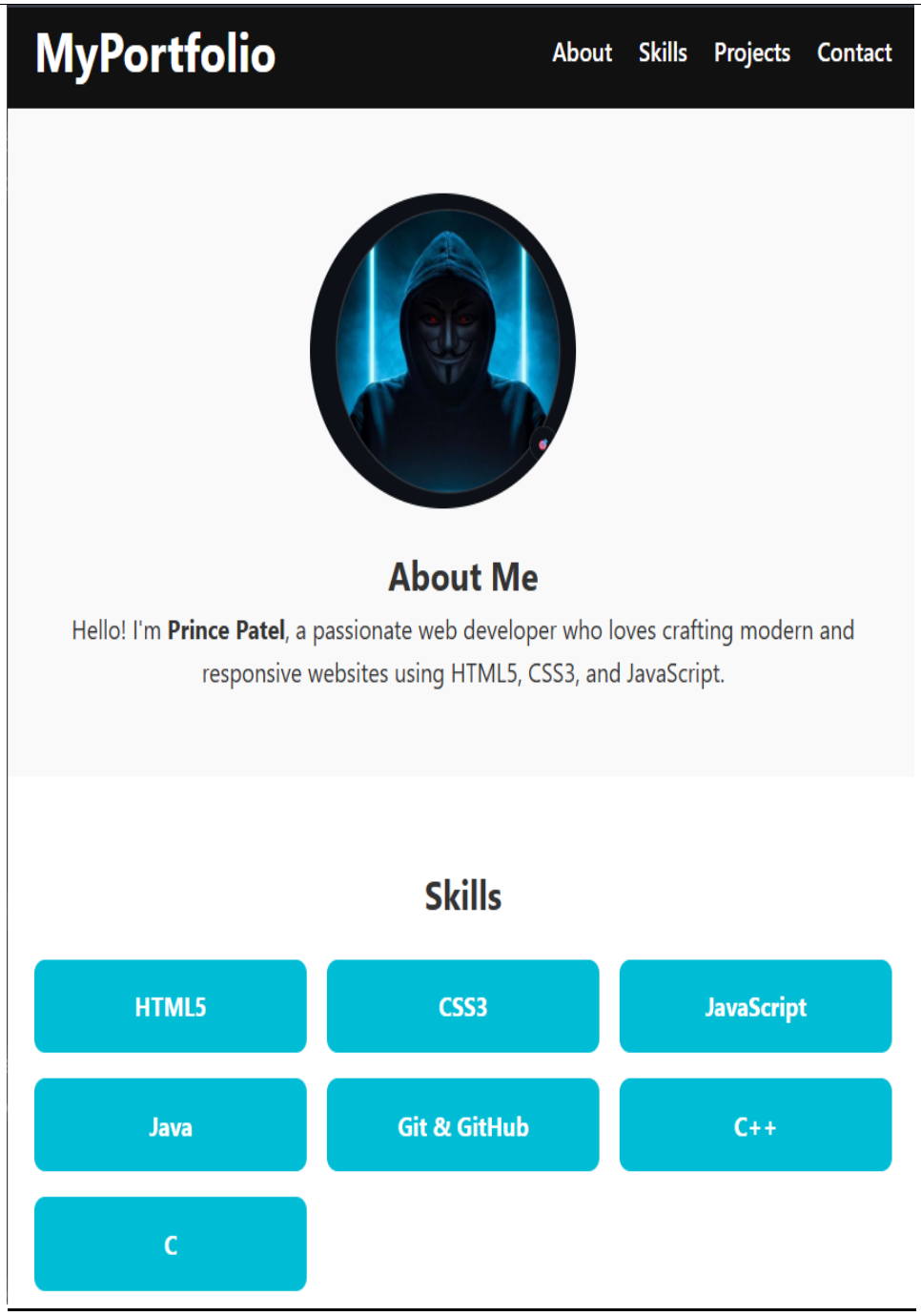
```
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}

body {
  font-family: 'Segoe UI', sans-serif;
  line-height: 1.6;
  color: #333;
}

header {
  background: #111;
  color: #fff;
  height: 60px;
}

.navbar {
  max-width: 1100px;
  margin: auto;
  display: flex;
  justify-content: space-between;
  align-items: center;
  padding: 0 20px;
}

.nav-links {
  list-style: none;
  display: flex;
  gap: 20px;
}

.nav-links a {
```

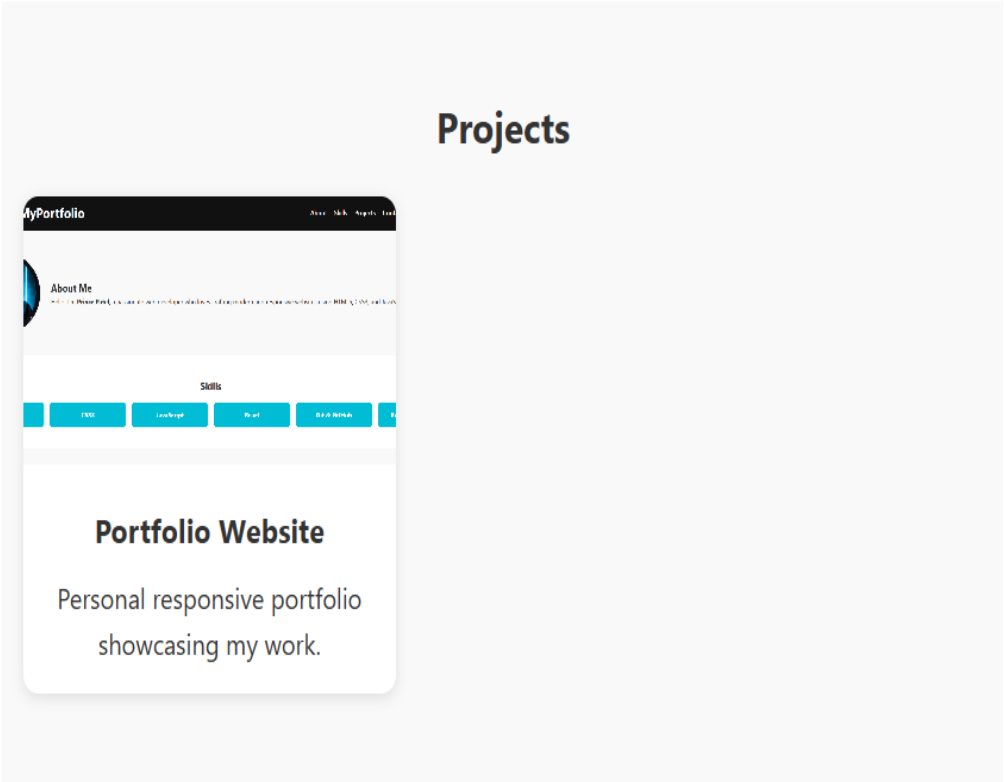
```
  color: #fff;
  text-decoration: none;
  font-weight: 500;
}

.nav-links a:hover {
  color: #00bcd4;
}

.about {
  display: flex;
  flex-wrap: wrap;
  align-items: center;
  padding: 50px 20px;
  background: #f9f9f9;
}

.profile-img {
  width: 200px;
  border-radius: 50%;
  margin-right: 30px;
}

.about-text {
  flex: 1;
  min-width: 250px;
}

.skills {
  padding: 50px 20px;
  text-align: center;
}

.skills-grid {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(150px, 1fr));
  gap: 15px;
  margin-top: 20px;
}

.skill-card {
  background: #00bcd4;
  color: #fff;
  padding: 15px;
  border-radius: 8px;
  font-weight: bold;
}
```

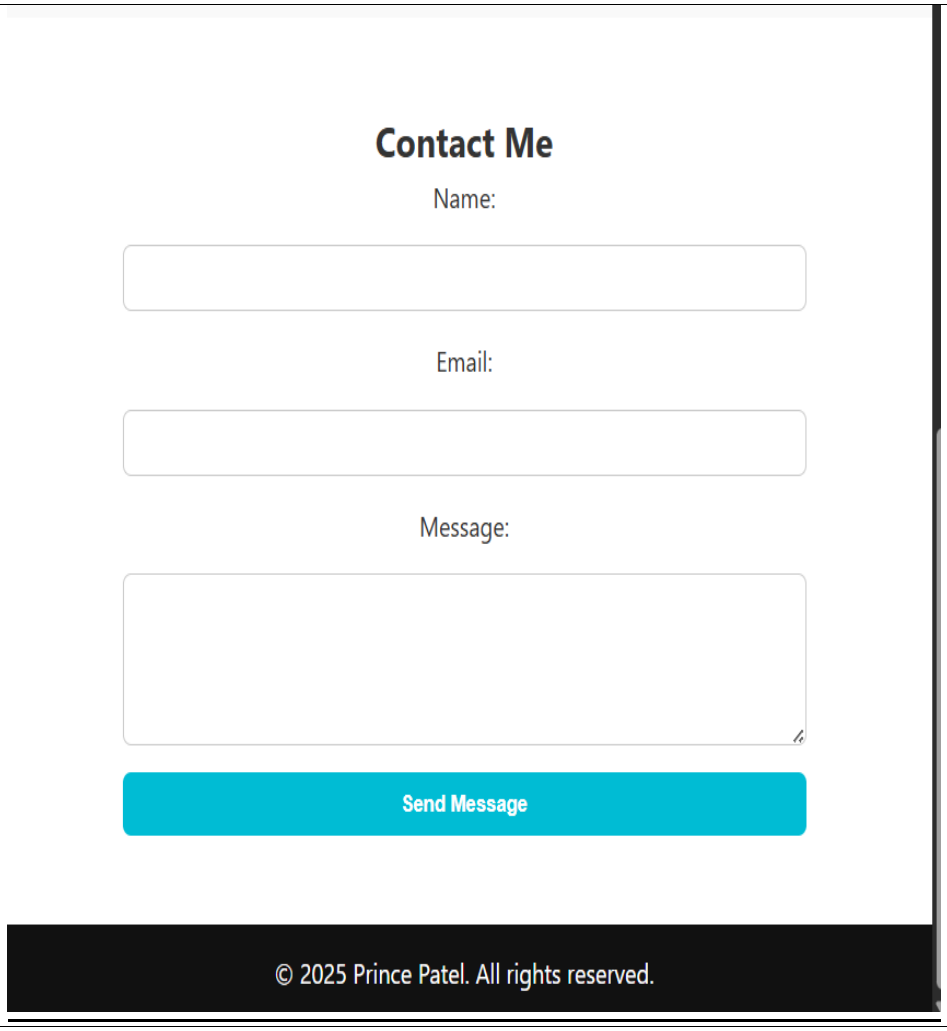
```
.projects {
  padding: 50px 20px;
  background: #f9f9f9;
  text-align: center;
}

.project-grid {
  display: grid;
  width: 250px;
  gap: 20px;
  margin-top: 20px;
}

.project-card {
  background: #fff;
  border-radius: 10px;
  overflow: hidden;
  box-shadow: 0 2px 8px rgba(0, 0, 0, 0.1);
}

.project-card img {
  width: 100%;
  height: 150px;
  object-fit: cover;
}

.project-card h3 {
  margin: 15px 0 10px;
}

.project-card p {
  padding: 0 15px 15px;
}

.contact {
  padding: 50px 20px;
  text-align: center;
}

form {
  max-width: 500px;
  margin: auto;
  display: flex;
  flex-direction: column;
  gap: 15px;
}
```

```
input, textarea {
  padding: 10px;
  border: 1px solid #ccc;
  border-radius: 6px;
  width: 100%;
}

button {
  padding: 10px;
  background: #00bcd4;
  border: none;
  color: white;
  font-weight: bold;
  border-radius: 6px;
  cursor: pointer;
}

button:hover {
  background: #0097a7;
}

footer {
  text-align: center;
  padding: 15px;
  background: #111;
  color: #fff;
}

@media (max-width: 768px) {
  .about {
    flex-direction: column;
    text-align: center;
  }

  .profile-img {
    margin-bottom: 20px;
  }
}

h1{
  padding-right: 90px;
  margin-right: 100px;
}
```

Screenshot:


MyPortfolio
About
Skills
Projects
Contact
About Me
Hello! I'm Prince Patel, a passionate web developer who loves crafting modern and responsive websites using HTML5, CSS3, and JavaScript.
Skills
HTML5
CSS3
JavaScript
Java
Git & GitHub
C++
C


Projects
MyPortfolio
About Me
Skills
Portfolio Website
Personal responsive portfolio showcasing my work.


Contact Me
Name:
Email:
Message:
Send Message
© 2025 Prince Patel. All rights reserved.

2)	Definition :	Color & Typography Playground (CSS)
	Programming concept :	Create a style guide page demonstrating color palette, font stacks, and heading/body typography. Requirements: Variables (custom properties), contrast-friendly palette, type scale (H1–H6).
	Code:	**Styleguide.html:-** <!DOCTYPE html> <html lang="en"> <head> <meta charset="UTF-8"> <meta name="viewport" content="width=device-width, initial-scale=1.0"> <title>Color & Typography Playground</title> <link rel="stylesheet" href="styleguide.css"> </head> <body> <header> Color & Typography Playground Explore consistent color themes and type hierarchy using CSS variables. </header> <section class="colors"> Color Palette --color-primary #2B6CB0 --color-secondary #68D391 --color-accent #F6AD55 --color-bg #F7FAFC --color-text #1A202C

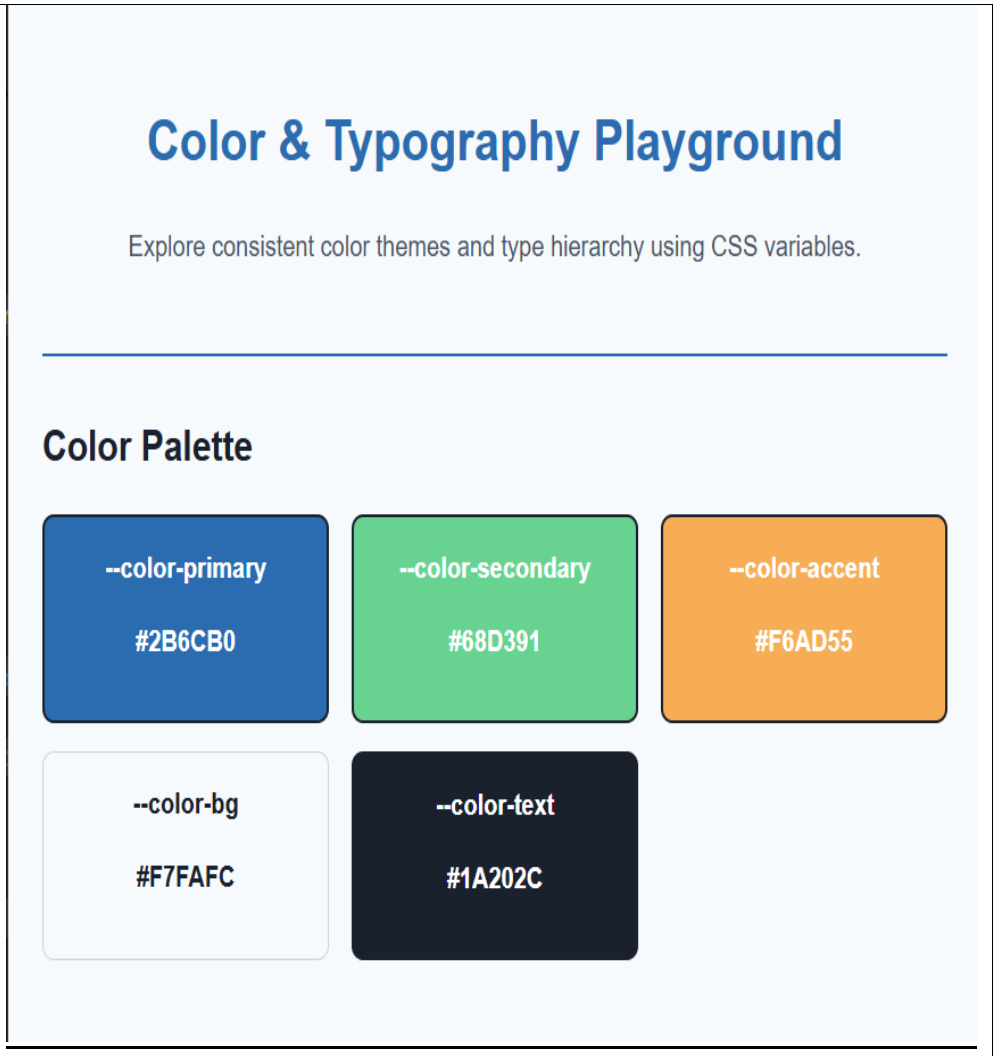
```
    </div>
  </section>

  <section class="typography">
    <h2>Typography Scale</h2>
    <p class="description">Demonstration of heading hierarchy and body text styles.</p>

    <h1>Heading 1 – 2.5rem</h1>
    <h2>Heading 2 – 2rem</h2>
    <h3>Heading 3 – 1.75rem</h3>
    <h4>Heading 4 – 1.5rem</h4>
    <h5>Heading 5 – 1.25rem</h5>
    <h6>Heading 6 – 1rem</h6>

    <p>Body text uses the font stack defined in <code>--font-body</code>. This ensures a readable and consistent layout across browsers.</p>
    <p><strong>Example of bold text</strong>, <em>italic text</em>, and <a href="#">a link</a>.</p>
  </section>

  <footer>
    <p>Made with ❤️ using HTML5 & CSS3 variables.</p>
  </footer>
</body>
</html>
```

styleguide.css:-

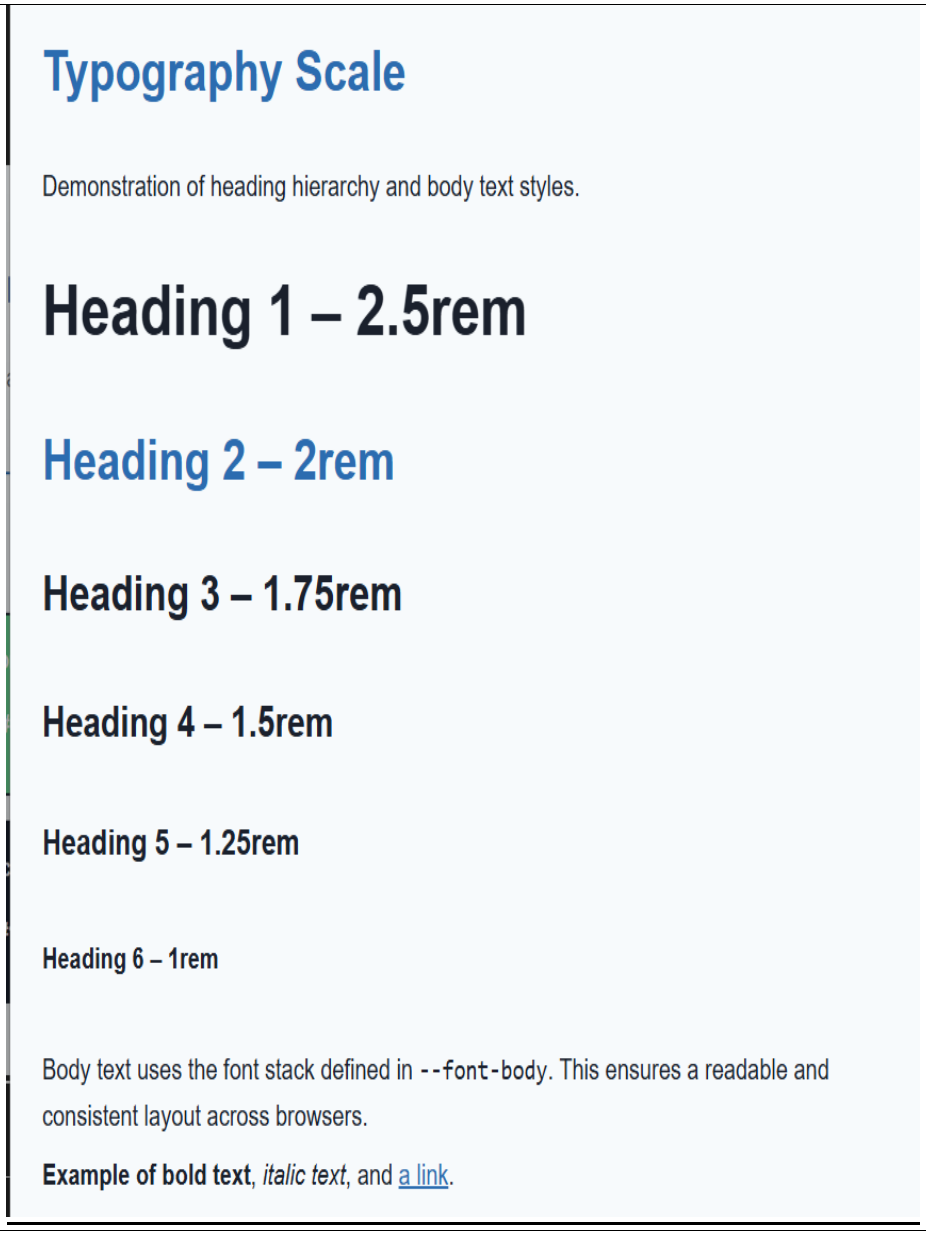
```
:root {
  --color-primary: #2B6CB0;
  --color-secondary: #68D391;
  --color-accent: #F6AD55;
  --color-bg: #F7FAFC;
  --color-text: #1A202C;

  --font-heading: 'Poppins', Arial, sans-serif;
  --font-body: 'Open Sans', Helvetica, sans-serif;

  --h1-size: 2.5rem;
  --h2-size: 2rem;
  --h3-size: 1.75rem;
  --h4-size: 1.5rem;
  --h5-size: 1.25rem;
```

```
    --h6-size: 1rem;
    --body-size: 1rem;
}

body {
    background-color: var(--color-bg);
    color: var(--color-text);
    font-family: var(--font-body);
    line-height: 1.6;
    margin: 0;
    padding: 0 1.5rem;
}

header {
    text-align: center;
    padding: 2rem 0;
    border-bottom: 2px solid var(--color-primary);
}
header h1 {
    font-family: var(--font-heading);
    color: var(--color-primary);
}
header p {
    color: #4A5568;
}

.colors {
    margin-top: 2rem;
}
.color-grid {
    display: grid;
    grid-template-columns: repeat(auto-fit, minmax(150px, 1fr));
    gap: 1rem;

}
.color-swatch {
    border-radius: 8px;
    padding: 1rem;
    color: white;
    text-align: center;
    font-weight: bold;
    border: #1A202C solid 2px;
}
.primary { background-color: var(--color-primary); }
.secondary { background-color: var(--color-secondary); }
.accent { background-color: var(--color-accent); }
.background {
```

```
    background-color: var(--color-bg);
    color: var(--color-text);
    border: 1px solid #CBD5E0;
}
.text {
    background-color: var(--color-text);
}

.typography {
    margin-top: 3rem;
}
.typography h2 {
    color: var(--color-primary);
    font-family: var(--font-heading);
}
.typography h1 { font-size: var(--h1-size); }
.typography h2 { font-size: var(--h2-size); }
.typography h3 { font-size: var(--h3-size); }
.typography h4 { font-size: var(--h4-size); }
.typography h5 { font-size: var(--h5-size); }
.typography h6 { font-size: var(--h6-size); }
.typography p {
    font-size: var(--body-size);
    margin: 0.5rem 0;
}
.typography a {
    color: var(--color-primary);
    text-decoration: underline;
}

footer {
    text-align: center;
    padding: 1.5rem 0;
    margin-top: 3rem;
    border-top: 1px solid #E2E8F0;
    color: #718096;
}
```

	Screenshot:	

Color & Typography Playground
Explore consistent color themes and type hierarchy using CSS variables.
Color Palette
--color-primary
#2B6CB0
--color-secondary
#68D391
--color-accent
#F6AD55
--color-bg
#F7FAFC
--color-text
#1A202C


Typography Scale
Demonstration of heading hierarchy and body text styles.
Heading 1 – 2.5rem
Heading 2 – 2rem
Heading 3 – 1.75rem
Heading 4 – 1.5rem
Heading 5 – 1.25rem
Heading 6 – 1rem
Body text uses the font stack defined in --font-body. This ensures a readable and consistent layout across browsers.
Example of bold text, italic text, and a link.

3)	Definition :	Interactive Quiz (JavaScript ES6)
	Programming concept :	A 5-question MCQ quiz with score and review. Requirements: Questions array (objects), next/prev controls, final score, no page reload, arrow functions & const/let.

	Code:	**quize.html:-**
		`<!DOCTYPE html>`
		`<html lang="en">`
		`<head>`
		`<meta charset="UTF-8">`
		`<meta name="viewport" content="width=device-width, initial-scale=1.0">`
		`<title>Interactive Quiz</title>`
		`<link rel="stylesheet" href="quiz.css">`
		`</head>`
		`<body>`
		`<header><h1>Interactive Quiz</h1></header>`
		
		`<main>`
		`<div id="quiz-container" class="quiz-box">`
		`<h3 id="question"></h3>`
		`<div id="options"></div>`
		
		`<div class="controls">`
		`<button id="prev" class="btn">Previous</button>`
		`<button id="next" class="btn">Next</button>`
		`</div>`
		
		`<div id="result"></div>`
		`</div>`
		`</main>`
		
		`<footer>Created by: Prince Patel [24CE092]</footer>`
		
		`<script src="quiz.js"></script>`
		`</body>`
		`</html>`
		
		quize.css:-
		`body {`
		`font-family: Arial, sans-serif;`
		`background-color: #f0f4f7;`
		`margin: 0;`
		`text-align: center;`

```
}

header {
  background-color: #0078d7;
  color: white;
  padding: 15px;
  font-size: 28px;
}

.quiz-box {
  background: white;
  display: inline-block;
  padding: 25px;
  margin-top: 40px;
  border-radius: 12px;
  box-shadow: 0 0 10px rgba(0,0,0,0.2);
  width: 350px;
  text-align: left;
}

.option-btn {
  display: block;
  width: 100%;
  padding: 10px;
  margin: 5px 0;
  border: 1px solid #ccc;
  background: #e9ecef;
  border-radius: 6px;
  cursor: pointer;
  text-align: left;
  transition: 0.3s;
}

.option-btn:hover {
  background: #d7e9ff;
}

.option-btn.selected {
  background: #0078d7;
  color: white;
  border-color: #005bb5;
}

.controls {
  margin-top: 15px;
  text-align: center;
}
```

```
.btn {
  padding: 8px 15px;
  margin: 5px;
  background-color: #0078d7;
  color: white;
  border: none;
  border-radius: 5px;
  cursor: pointer;
}

.btn:hover {
  background-color: #005bb5;
}

footer {
  margin-top: 30px;
  padding: 10px;
  background-color: #0078d7;
  color: white;
}
```

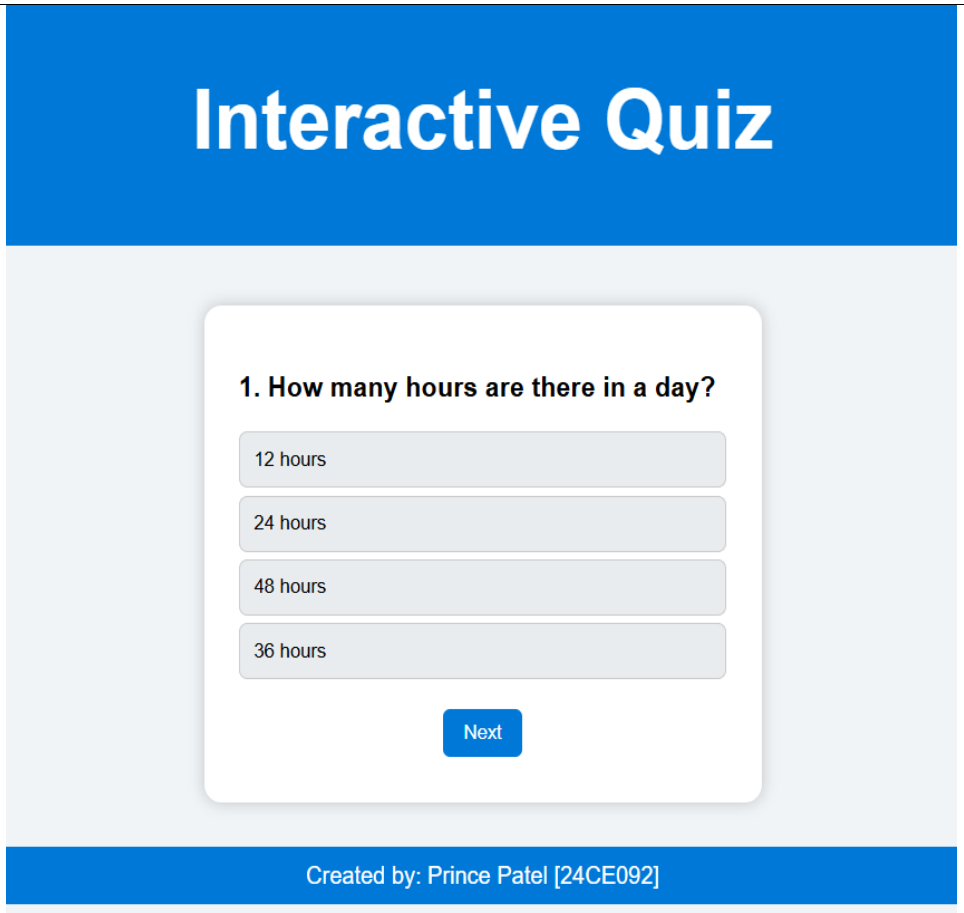
quize.js:-

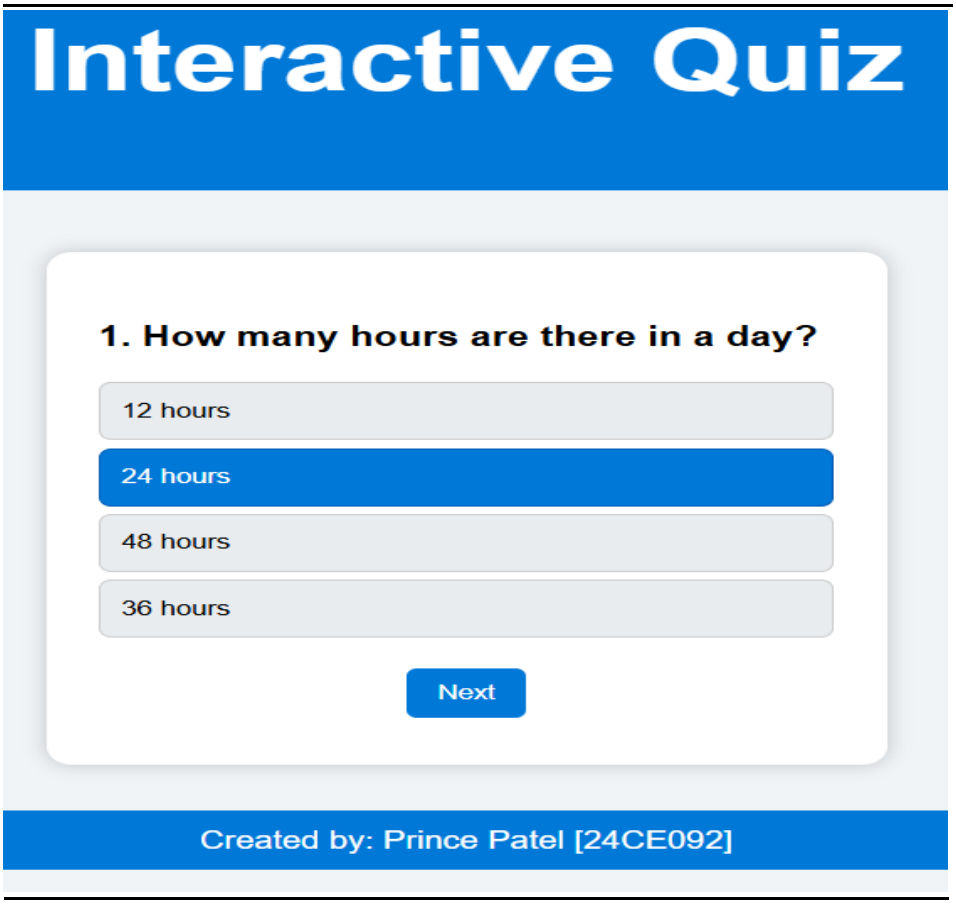
```
const questions = [
  {
    question: "How many hours are there in a day?",
    options: ["12 hours", "24 hours", "48 hours", "36 hours"],
    answer: "24 hours"
  },
  {
    question: "What is the capital of India?",
    options: ["Delhi", "Mumbai", "Surat", "Ahmedabad"],
    answer: "Delhi"
  },
  {
    question: "Which planet is known as the Red Planet?",
    options: ["Earth", "Mars", "Venus", "Jupiter"],
    answer: "Mars"
  },
  {
    question: "Which tag is used to include JavaScript in HTML?",
    options: ["<js>", "<javascript>", "<script>", "<code>"],
    answer: "<script>"
  }
];
```

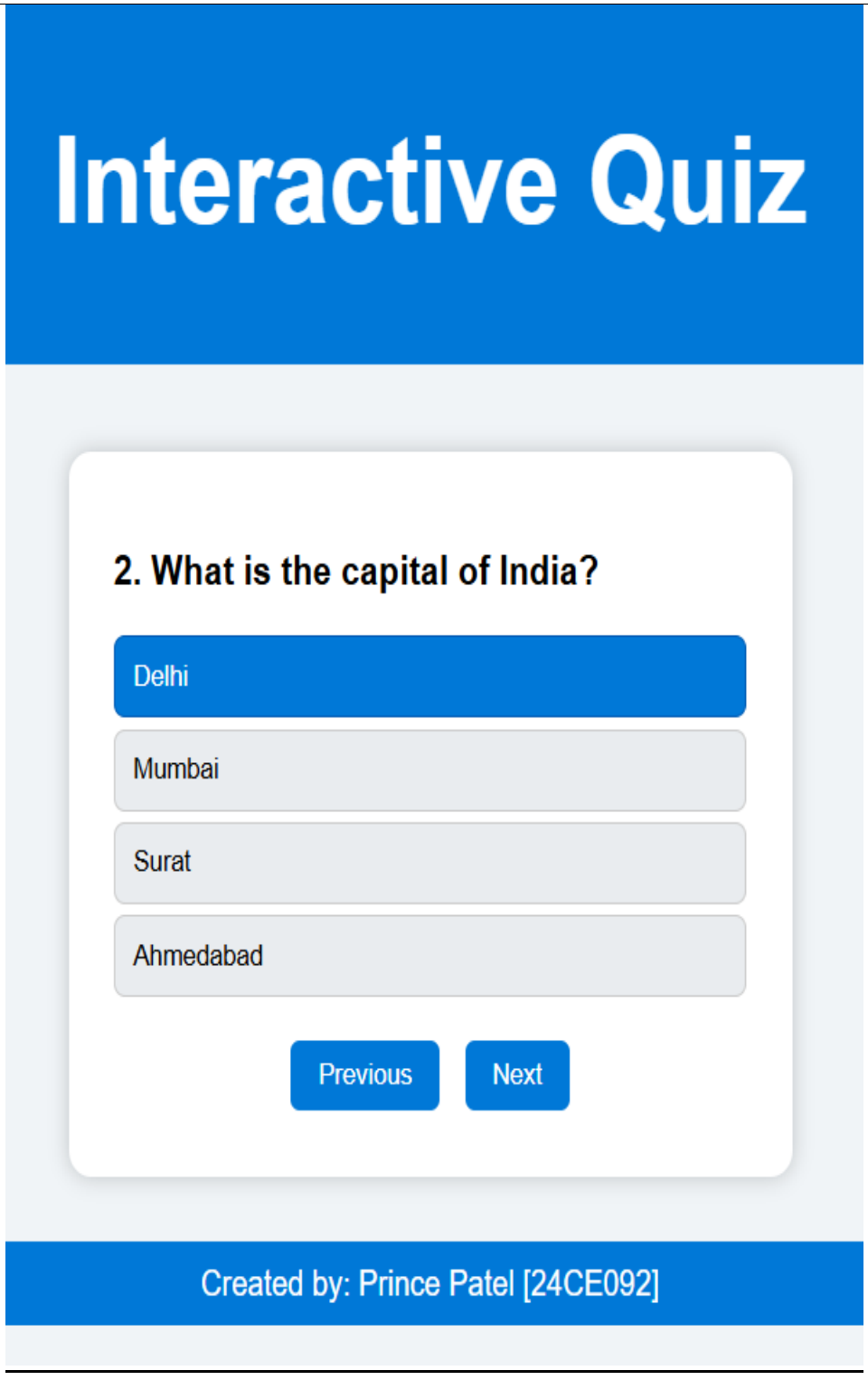
```
let currentQuestion = 0;
let score = 0;
let userAnswers = [];

const questionEl = document.getElementById("question");
const optionsEl = document.getElementById("options");
const nextBtn = document.getElementById("next");
const prevBtn = document.getElementById("prev");
const resultEl = document.getElementById("result");

const loadQuestion = () => {
 const q = questions[currentQuestion];
 questionEl.textContent = `${currentQuestion + 1}. ${q.question}`;
 optionsEl.innerHTML = "";

 q.options.forEach(option => {
  const btn = document.createElement("button");
  btn.textContent = option;
  btn.className = "option-btn";
  btn.onclick = () => selectAnswer(option);
  if (userAnswers[currentQuestion] === option)
btn.classList.add("selected");
  optionsEl.appendChild(btn);
 });

 prevBtn.style.display = currentQuestion === 0 ? "none" : "inline-block";
 nextBtn.textContent = currentQuestion === questions.length - 1 ?
"Finish" : "Next";
};

const selectAnswer = (option) => {
 userAnswers[currentQuestion] = option;
 loadQuestion();
}

nextBtn.addEventListener("click", () => {
 if (!userAnswers[currentQuestion]) {
  alert("Please select an answer before continuing!");
  return;
 }

 if (currentQuestion < questions.length - 1) {
  currentQuestion++;
  loadQuestion();
 } else {
  calculateScore();
```

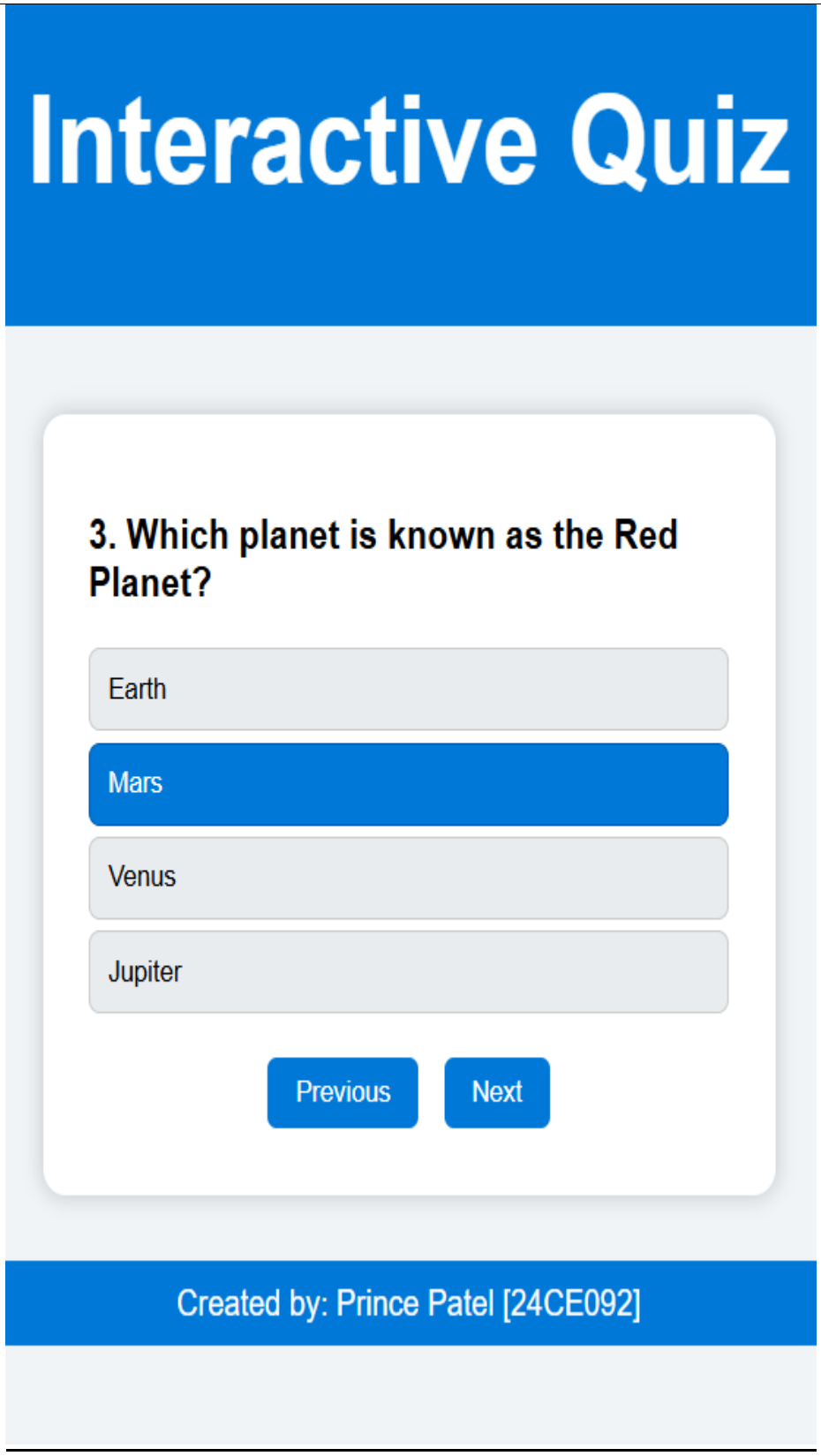
```
  }
});

prevBtn.addEventListener("click", () => {
 if (currentQuestion > 0) {
  currentQuestion--;
  loadQuestion();
 }
});

const calculateScore = () => {
 score = 0;
 questions.forEach((q, i) => {
  if (userAnswers[i] === q.answer) score++;
 });

 document.getElementById("quiz-container").innerHTML = `
  <h2>Your Score: ${score} / ${questions.length}</h2>
  <h3>Review:</h3>
  <ul>
   ${questions.map((q, i) => `
    <li>
     <strong>Q${i + 1}:</strong> ${q.question}<br>
     ✅ Correct: ${q.answer}<br>
     🟡 Your Answer: ${userAnswers[i] || "Not answered"}
    </li><br>
   `).join("")}
  </ul>
  <button class="btn" onclick="location.reload()">Restart Quiz</button>
 `;
};

loadQuestion();
```

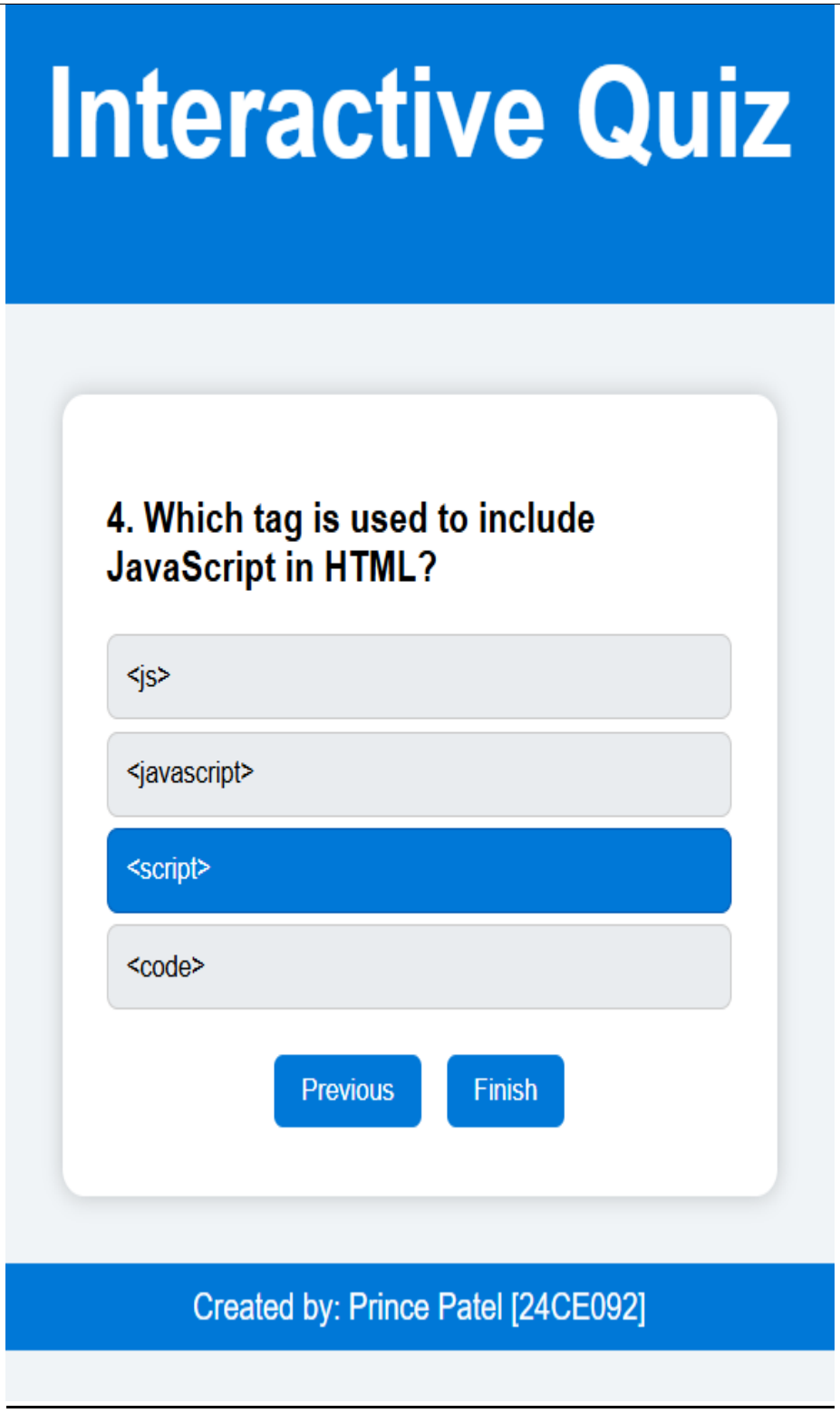
Screenshot:

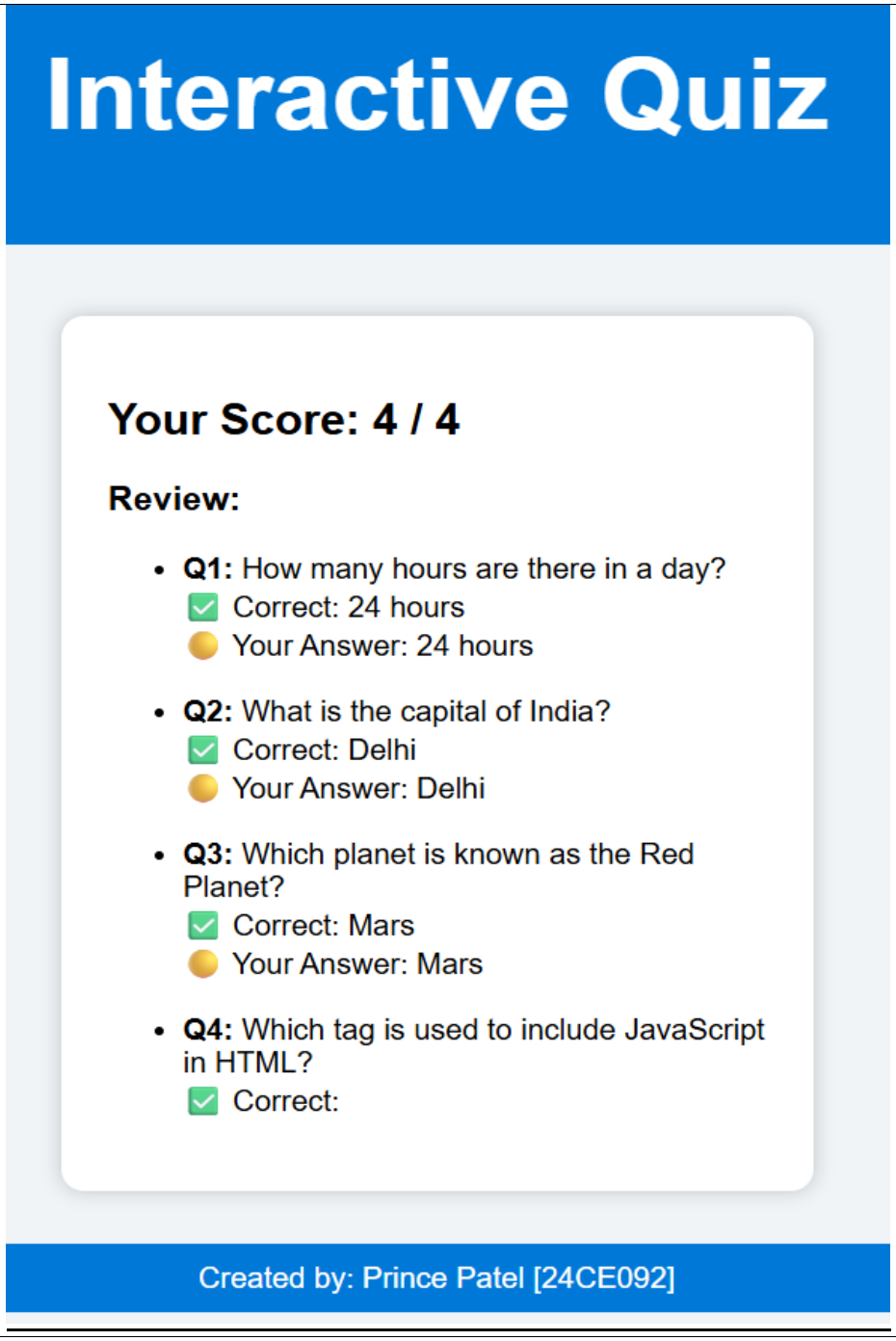

Interactive Quiz
1. How many hours are there in a day?
12 hours
24 hours
48 hours
36 hours
Next
Created by: Prince Patel [24CE092]


Interactive Quiz
1. How many hours are there in a day?
12 hours
24 hours
48 hours
36 hours
Next
Created by: Prince Patel [24CE092]


Interactive Quiz
2. What is the capital of India?
Delhi
Mumbai
Surat
Ahmedabad
Previous
Next
Created by: Prince Patel [24CE092]


Interactive Quiz
3. Which planet is known as the Red Planet?
Earth
Mars
Venus
Jupiter
Previous
Next
Created by: Prince Patel [24CE092]


Interactive Quiz
4. Which tag is used to include JavaScript in HTML?
<js>
<javascript>
<script>
<code>
Previous
Finish
Created by: Prince Patel [24CE092]


Interactive Quiz
Your Score: 4 / 4
Review:
Q1: How many hours are there in a day?
Correct: 24 hours
Your Answer: 24 hours
Q2: What is the capital of India?
Correct: Delhi
Your Answer: Delhi
Q3: Which planet is known as the Red Planet?
Correct: Mars
Your Answer: Mars
Q4: Which tag is used to include JavaScript in HTML?
Correct:
Created by: Prince Patel [24CE092]

4)	Definition :	DOM To-Do App (DOM + Events)
	Programming concept :	Add, complete, delete to-dos with persistent state (localStorage). Requirements: Event listeners, dynamic DOM (create/remove), filter: All/Active/Completed, save & load.

	Code:	**todo.html:-**
		`<!DOCTYPE html>`
		`<html lang="en">`
		`<head>`
		`<meta charset="UTF-8">`
		`<meta name="viewport" content="width=device-width, initial-scale=1.0">`
		`<title>DOM To-Do App</title>`
		`<link rel="stylesheet" href="todo.css">`
		`</head>`
		`<body>`
		`<header>`
		`<h1>My To-Do List</h1>`
		`</header>`
		
		`<main class="todo-container">`
		`<div class="input-section">`
		`<input type="text" id="todo-input" placeholder="Enter a new task...">`
		`<button id="add-btn" class="btn">Add</button>`
		`</div>`
		
		`<div class="filters">`
		`<button class="filter-btn active" data-filter="all">All</button>`
		`<button class="filter-btn" data-filter="active">Active</button>`
		`<button class="filter-btn" data-filter="completed">Completed</button>`
		`</div>`
		
		`<ul id="todo-list"></ul>`
		`</main>`
		
		`<footer>Created by: Prince Patel [24CE092]</footer>`
		
		`<script src="todo.js"></script>`
		`</body>`
		`</html>`
		
		todo.css:-
		`body {`
		`font-family: Arial, sans-serif;`
		`background: #f2f4f8;`
		`text-align: center;`

```
  margin: 0;
  padding: 0;
}

header {
  background: #0078d7;
  color: white;
  padding: 15px;
  font-size: 28px;
}

.todo-container {
  background: white;
  display: inline-block;
  padding: 20px;
  margin-top: 30px;
  border-radius: 10px;
  box-shadow: 0 0 10px rgba(0,0,0,0.2);
  width: 350px;
}

.input-section {
  display: flex;
  justify-content: center;
  margin-bottom: 15px;
}

#todo-input {
  width: 70%;
  padding: 8px;
  border: 1px solid #ccc;
  border-radius: 5px;
}

.btn {
  background-color: #0078d7;
  color: white;
  border: none;
  border-radius: 5px;
  padding: 8px 12px;
  margin-left: 5px;
  cursor: pointer;
}

.btn:hover {
  background-color: #005bb5;
}
```

```
.filters {
  margin-bottom: 15px;
}

.filter-btn {
  background: #e9ecef;
  border: none;
  padding: 6px 12px;
  margin: 2px;
  border-radius: 5px;
  cursor: pointer;
}

.filter-btn.active {
  background: #0078d7;
  color: white;
}

ul {
  list-style: none;
  padding: 0;
  text-align: left;
}

.todo-item {
  background: #f9f9f9;
  margin: 6px 0;
  padding: 8px 12px;
  border-radius: 5px;
  display: flex;
  justify-content: space-between;
  align-items: center;
}

.todo-item.completed span {
  text-decoration: line-through;
  color: #888;
}

.delete-btn {
  background: transparent;
  border: none;
  cursor: pointer;
  font-size: 16px;
}
```

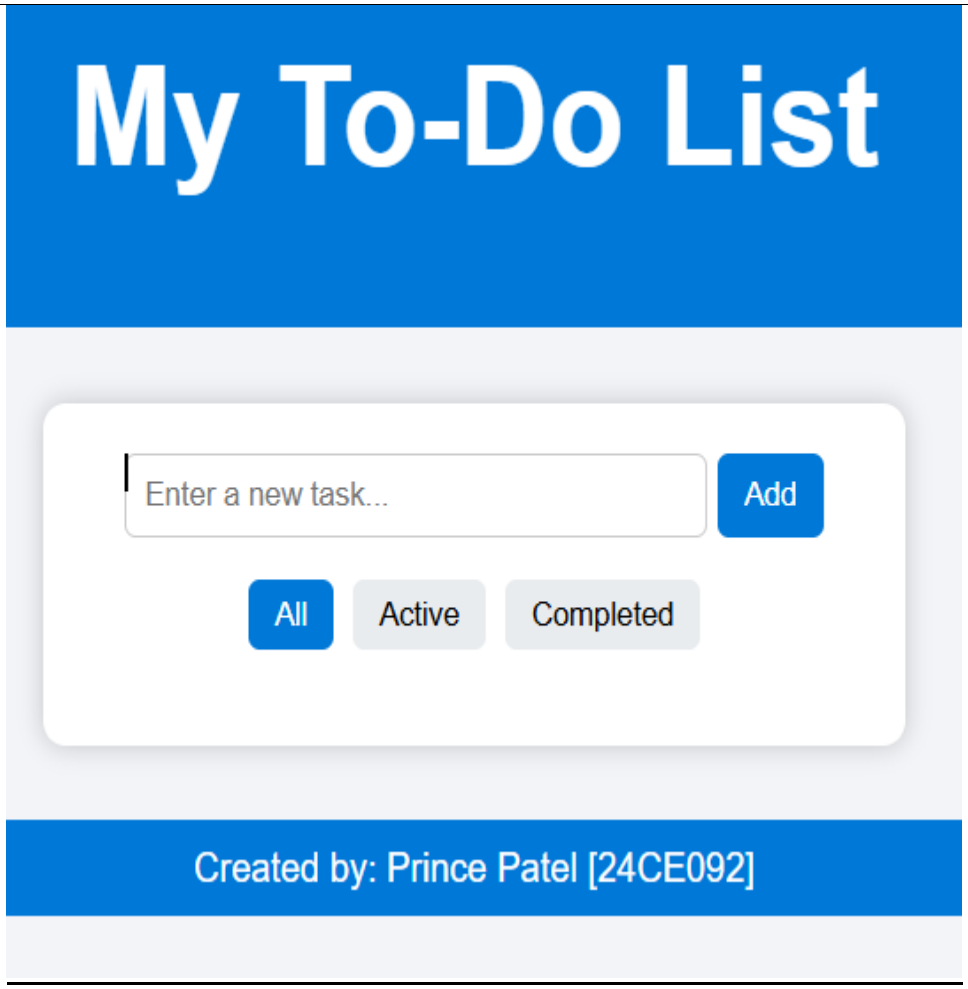
```
footer {
  background: #0078d7;
  color: white;
  padding: 10px;
  margin-top: 30px;
}
```

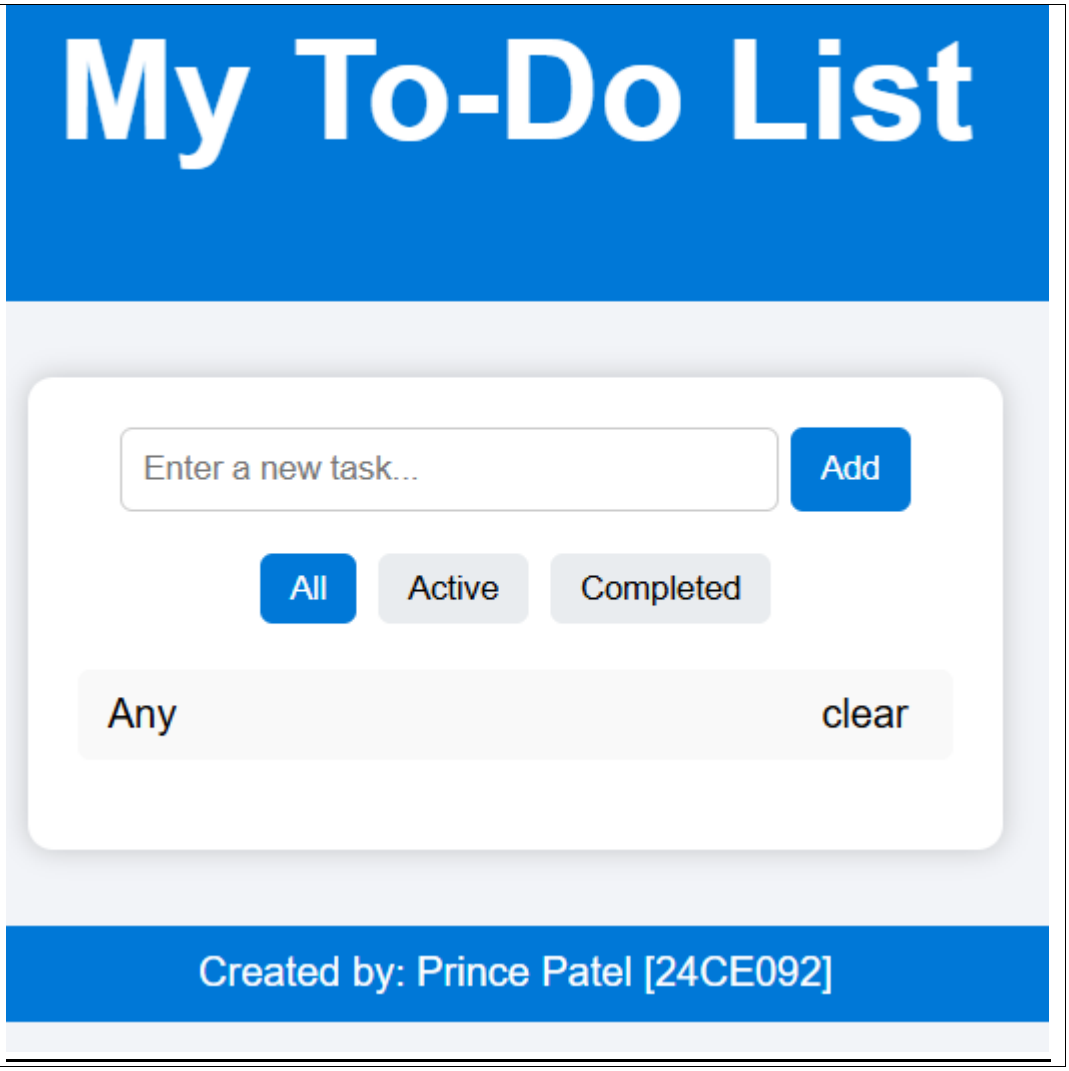
todo.js:-

```
const input = document.getElementById('todo-input');
const addBtn = document.getElementById('add-btn');
const list = document.getElementById('todo-list');
const filterButtons = document.querySelectorAll('.filter-btn');

let todos = JSON.parse(localStorage.getItem('todos')) || [];
let currentFilter = 'all';

const renderTodos = () => {
  list.innerHTML = '';

  const filtered = todos.filter(todo => {
    if (currentFilter === 'active') return !todo.completed;
    if (currentFilter === 'completed') return todo.completed;
    return true;
  });

  filtered.forEach((todo, index) => {
    const li = document.createElement('li');
    li.className = 'todo-item';
    if (todo.completed) li.classList.add('completed');

    const span = document.createElement('span');
    span.textContent = todo.text;
    span.addEventListener('click', () => toggleTodo(index));

    const delBtn = document.createElement('button');
    delBtn.textContent = 'clear';
    delBtn.className = 'delete-btn';
    delBtn.addEventListener('click', () => deleteTodo(index));

    li.appendChild(span);
    li.appendChild(delBtn);
    list.appendChild(li);
  });

  saveTodos();
```

```
};

const addTodo = () => {
 const text = input.value.trim();
 if (text === '') {
  alert('Please enter a task!');
  return;
 }

 todos.push({ text, completed: false });
 input.value = '';
 renderTodos();
};

const toggleTodo = (index) => {
 todos[index].completed = !todos[index].completed;
 renderTodos();
};

const deleteTodo = (index) => {
 todos.splice(index, 1);
 renderTodos();
};

filterButtons.forEach(btn => {
 btn.addEventListener('click', () => {
  filterButtons.forEach(b => b.classList.remove('active'));
  btn.classList.add('active');
  currentFilter = btn.getAttribute('data-filter');
  renderTodos();
 });
});

const saveTodos = () => {
 localStorage.setItem('todos', JSON.stringify(todos));
};

addBtn.addEventListener('click', addTodo);
input.addEventListener('keypress', e => {
 if (e.key === 'Enter') addTodo();
});

renderTodos();
```

Screenshot:


My To-Do List
Enter a new task...
Add
All
Active
Completed
Created by: Prince Patel [24CE092]


My To-Do List
Enter a new task...
Add
All
Active
Completed
Any
clear
Created by: Prince Patel [24CE092]

5)	Definition :	Data Dashboard Cards (Numbers/Arrays/Array Methods)
	Programming concept :	Render cards from a dataset and compute KPIs. Requirements: Given array of objects (e.g., sales), use map/filter/reduce/sort to calculate totals, averages, top N; render to DOM.
	Code:	**dashboard.html:-** <!DOCTYPE html> <html lang="en"> <head> <meta charset="UTF-8"> <meta name="viewport" content="width=device-width, initial-scale=1.0"> <title>Sales Data Dashboard</title> <link rel="stylesheet" href="dashboard.css"> </head> <body> <header> 📊 Sales Dashboard </header> <main> <section id="kpi-cards" class="card-container"></section> <section id="top-products" class="table-section"> 🏆 Top 3 Products <table id="product-table"> <thead> <tr> <th>Product</th> <th>Units Sold</th> <th>Revenue ($)</th> </tr> </thead> <tbody></tbody> </table> </section> </main> <footer>Created by: Prince Patel [24CE092]</footer> <script src="dashboard.js"></script> </body> </html>

dashboard.css-

```
const salesData = [
  { product: "Laptop", units: 30, price: 750 },
  { product: "Phone", units: 80, price: 500 },
  { product: "Tablet", units: 45, price: 300 },
  { product: "Monitor", units: 25, price: 200 },
  { product: "Keyboard", units: 60, price: 50 },
  { product: "Mouse", units: 75, price: 40 },
];

const totalRevenue = salesData
  .map(item => item.units * item.price)
  .reduce((sum, value) => sum + value, 0);

const totalUnits = salesData.reduce((sum, item) => sum + item.units, 0);

const avgRevenue = (totalRevenue / salesData.length).toFixed(2);

const topProduct = salesData.reduce((max, item) =>
  item.units > max.units ? item : max
);

const top3 = [...salesData]
  .sort((a, b) => (b.units * b.price) - (a.units * a.price))
  .slice(0, 3);

const kpiContainer = document.getElementById("kpi-cards");
const kpiData = [
  { title: "Total Revenue", value: `$${totalRevenue.toLocaleString()}` },
  { title: "Total Units Sold", value: totalUnits },
  { title: "Average Revenue/Product", value: `$${avgRevenue}` },
  { title: "Top Product", value: topProduct.product },
];

kpiContainer.innerHTML = kpiData
  .map(kpi => `
    <div class="card">
      <h3>${kpi.title}</h3>
      <p>${kpi.value}</p>
    </div>
  `)
  .join("");

const tableBody = document.querySelector("#product-table tbody");
```

```
tableBody.innerHTML = top3
 .map(item => `
  <tr>
   <td>${item.product}</td>
   <td>${item.units}</td>
   <td>$${(item.units * item.price).toLocaleString()}</td>
  </tr>
 `)
 .join("");
```

Screenshot:

6)	Definition :	String Utilities Library
	Programming concept :	Build small utility functions: toTitleCase, trimExtraSpaces, wordFrequency, maskEmail. Requirements: Pure functions, unit-like test page showing inputs/outputs.

	Code:	**demo.html:-**
		<!DOCTYPE html>
		<html lang="en">
		<head>
		<meta charset="UTF-8">
		<meta name="viewport" content="width=device-width, initial-scale=1.0">
		<title>String Utilities Demo</title>
		<style>
		body {
		font-family: Arial, sans-serif;
		background-color: #f5f7fa;
		text-align: center;
		margin: 0;
		padding: 0;
		}
		
		header {
		background-color: #0078d7;
		color: white;
		padding: 15px;
		font-size: 28px;
		}
		
		main {
		display: flex;
		flex-direction: column;
		align-items: center;
		margin-top: 30px;
		gap: 20px;
		}
		
		.test-box {
		background: white;
		border-radius: 10px;
		box-shadow: 0 0 10px rgba(0,0,0,0.1);
		width: 400px;
		text-align: left;
		padding: 15px;
		}

```
  .test-box h3 {
    color: #0078d7;
    border-bottom: 1px solid #ddd;
    padding-bottom: 5px;
  }

  code {
    display: block;
    background: #f0f0f0;
    padding: 8px;
    border-radius: 5px;
    margin: 5px 0;
    white-space: pre-wrap;
  }

  footer {
    margin-top: 40px;
    background: #0078d7;
    color: white;
    padding: 10px;
  }
 </style>
</head>
<body>
 <header>🧩 String Utilities Library Demo</header>

 <main id="demo"></main>

 <footer>Created by: Prince Patel [24CE092]</footer>

 <script type="module">
  import { toTitleCase, trimExtraSpaces, wordFrequency, maskEmail } from
'./string-utils.js';

  const demoContainer = document.getElementById("demo");

  // Example inputs
  const examples = [
   {
    title: "1️⃣ toTitleCase()",
    input: "hello world from charusat university",
    output: toTitleCase("hello world from charusat university")
   },
   {
    title: "2️⃣ trimExtraSpaces()",
    input: "  This  is  a  spaced  text!  ",
```

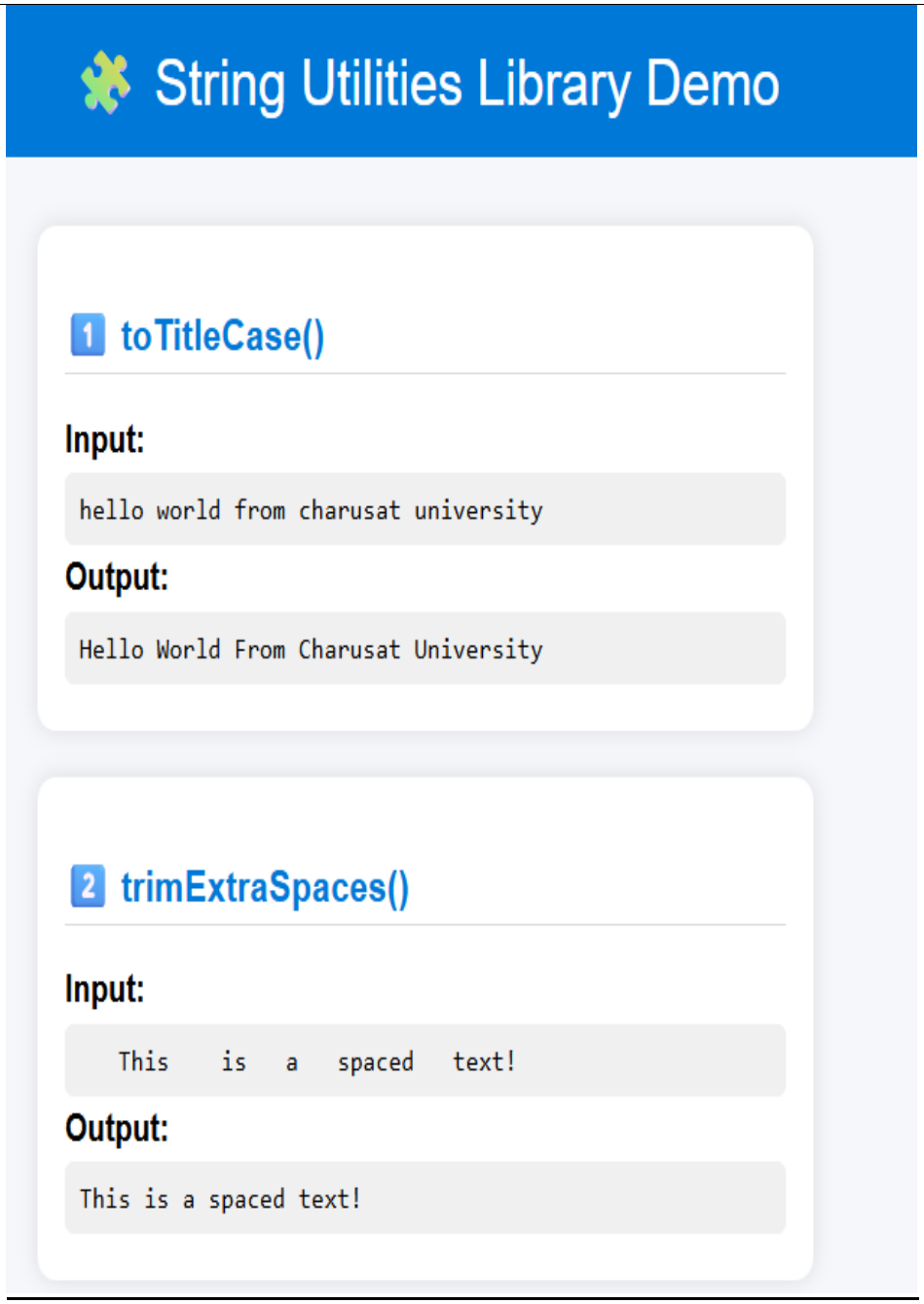
```
      output: trimExtraSpaces("  This  is  a  spaced  text!  ")
    },
    {
      title: "3 wordFrequency()",
      input: "This is a test. This test is simple.",
      output: JSON.stringify(wordFrequency("This is a test. This test is
simple."), null, 2)
    },
    {
      title: "4 maskEmail()",
      input: "princepatel123@gmail.com",
      output: maskEmail("princepatel123@gmail.com")
    }
  ];

  // Render results
  demoContainer.innerHTML = examples.map(ex => `
    <div class="test-box">
      <h3>${ex.title}</h3>
      <strong>Input:</strong>
      <code>${ex.input}</code>
      <strong>Output:</strong>
      <code>${ex.output}</code>
    </div>
  `).join("");
 </script>
</body>
</html>
```

string-util.css:-

```
export const toTitleCase = (str) => {
 return str
  .toLowerCase()
  .split(" ")
  .filter(word => word.trim() !== "")
  .map(word => word[0].toUpperCase() + word.slice(1))
  .join(" ");
};

export const trimExtraSpaces = (str) => {
 return str.trim().replace(/\s+/g, " ");
};

export const wordFrequency = (str) => {
 const words = trimExtraSpaces(str.toLowerCase()).split(" ");
```

```
  const freq = {};
  words.forEach(w => {
    if (w) freq[w] = (freq[w] || 0) + 1;
  });
  return freq;
};

export const maskEmail = (email) => {
  const [user, domain] = email.split("@");
  if (!domain) return "Invalid Email";
  const maskedUser = user[0] + "*".repeat(Math.max(0, user.length - 2)) +
user.slice(-1);
  return `${maskedUser}@${domain}`;
};
```

Screenshot:


String Utilities Library Demo
1 toTitleCase()
Input:
hello world from charusat university
Output:
Hello World From Charusat University
2 trimExtraSpaces()
Input:
This is a spaced text!
Output:
This is a spaced text!

3 wordFrequency()

Input:

```
This is a test. This test is simple.
```

Output:

```
{
  "this": 2,
  "is": 2,
  "a": 1,
  "test.": 1,
  "test": 1,
  "simple.": 1
}
```

4 maskEmail()

Input:

```
princepatel123@gmail.com
```

Output:

```
p***********3@gmail.com
```

Created by: Prince Patel [24CE092]

7)	Definition :	JSON Importer + Renderer
	Programming concept :	Parse a JSON file of events and render them grouped by month. Requirements: Fetch local JSON (via fetch or inline data), defensive parsing, grouping logic, formatted dates

	Code:	**Events.html:-**
		<!DOCTYPE html>
		<html lang="en">
		<head>
		<meta charset="UTF-8" />
		<meta name="viewport" content="width=device-width, initial-scale=1.0" />
		<title>Events Dashboard</title>
		<link rel="stylesheet" href="events.css">
		<style>
		body {
		font-family: Arial, sans-serif;
		background: #f4f6fa;
		margin: 0;
		padding: 0;
		}
		
		header {
		background: #0078d7;
		color: white;
		text-align: center;
		padding: 15px;
		font-size: 26px;
		}
		
		main {
		padding: 20px;
		display: flex;
		flex-direction: column;
		align-items: center;
		}
		
		.month-card {
		background: white;
		width: 80%;
		margin-bottom: 25px;
		border-radius: 10px;
		box-shadow: 0 0 8px rgba(0,0,0,0.1);
		padding: 15px 20px;
		}

```
    .month-card h2 {
      color: #0078d7;
      border-bottom: 2px solid #0078d7;
      padding-bottom: 5px;
    }

    .event {
      padding: 10px 0;
      border-bottom: 1px solid #eee;
    }

    .event:last-child {
      border-bottom: none;
    }

    footer {
      background: #0078d7;
      color: white;
      text-align: center;
      padding: 10px;
    }
  </style>
</head>
<body>
  <header>📅 Event Renderer</header>

  <main>
    <div id="events-container"></div>
  </main>

  <footer>Created by: Prince Patel [24CE092]</footer>

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

Events.js:-

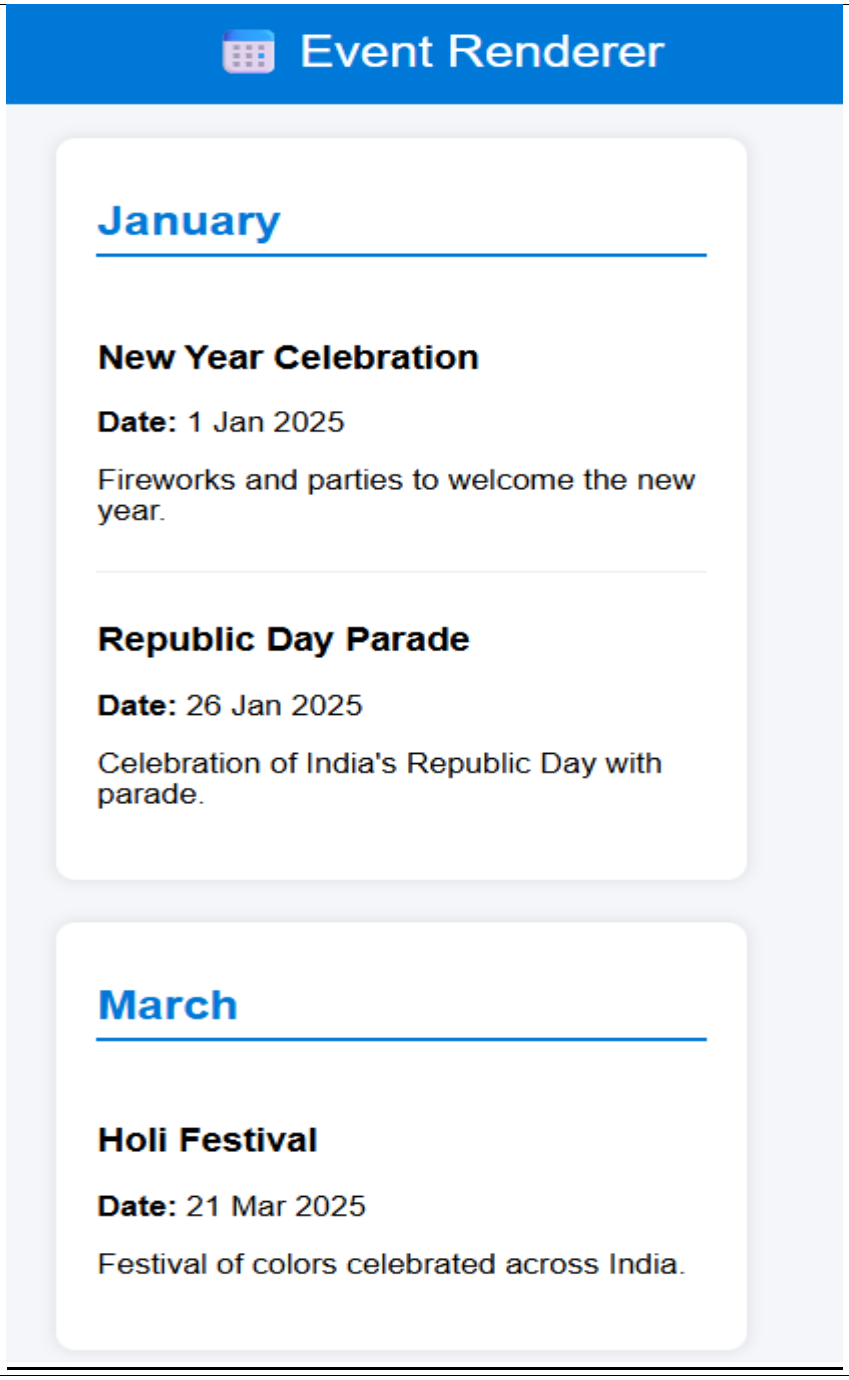
```
async function loadEvents() {
  const container = document.getElementById("events-container");
  container.innerHTML = "<p>Loading events...</p>";

  try {
```

```
    const response = await fetch("events.json");
    if (!response.ok) throw new Error("Failed to load events file");

    const events = await response.json();

    if (!Array.isArray(events)) throw new Error("Invalid JSON format");

    const grouped = groupByMonth(events);

    container.innerHTML = renderEvents(grouped);
  } catch (err) {
    container.innerHTML = `<p style="color:red;">Error:
${err.message}</p>`;
  }
}

function groupByMonth(events) {
  const months = [
    "January", "February", "March", "April", "May", "June",
    "July", "August", "September", "October", "November", "December"
  ];

  const grouped = {};
  events.forEach(ev => {
    const date = new Date(ev.date);
    const monthName = months[date.getMonth()] || "Unknown";
    if (!grouped[monthName]) grouped[monthName] = [];
    grouped[monthName].push(ev);
  });

  return Object.keys(grouped)
    .sort((a, b) => months.indexOf(a) - months.indexOf(b))
    .reduce((acc, key) => {
      acc[key] = grouped[key];
      return acc;
    }, {});
}

function renderEvents(grouped) {
  let html = "";
  for (const [month, events] of Object.entries(grouped)) {
    html += `
      <div class="month-card">
        <h2>${month}</h2>
        ${events.map(ev => `
          <div class="event">
            <h3>${ev.title}</h3>
```

```
          <p><strong>Date:</strong> ${formatDate(ev.date)}</p>
          <p>${ev.description}</p>
        </div>
      `).join("")}
    </div>
   `;
 }
 return html;
}

function formatDate(dateStr) {
 const date = new Date(dateStr);
 return date.toLocaleDateString("en-IN", {
  year: "numeric", month: "short", day: "numeric"
 });
}

document.addEventListener("DOMContentLoaded", loadEvents);
```

Events.json:-

```
[
 {
  "title": "New Year Celebration",
  "date": "2025-01-01",
  "description": "Fireworks and parties to welcome the new year."
 },
 {
  "title": "Republic Day Parade",
  "date": "2025-01-26",
  "description": "Celebration of India's Republic Day with parade."
 },
 {
  "title": "Holi Festival",
  "date": "2025-03-21",
  "description": "Festival of colors celebrated across India."
 },
 {
  "title": "Independence Day",
  "date": "2025-08-15",
  "description": "Commemorating India's independence."
 },
 {
  "title": "Diwali Festival",
  "date": "2025-11-03",
```

<table>
<tr><td></td><td></td><td>
<pre>
  "description": "Festival of lights celebrated with joy."
 },
 {
  "title": "Christmas Eve",
  "date": "2025-12-24",
  "description": "Celebration of Christmas Eve."
 }
]
</pre>
</td></tr>
<tr><td></td><td>Screenshot:</td><td> 
</td></tr>
</table>

August

Independence Day

Date: 15 Aug 2025

Commemorating India's independence.

November

Diwali Festival

Date: 3 Nov 2025

Festival of lights celebrated with joy.

December

Christmas Eve

Date: 24 Dec 2025

Celebration of Christmas Eve.

Created by: Prince Patel [24CE092]

8)	Definition :	Custom Iterable (Advanced JS Objects)
	Programming concept :	Make a custom iterable Range(start, end, step) usable in for...of and spread [...]. Requirements: Implement [Symbol.iterator], return iterator with next(), tests that show iteration and spreading

	Code:	**demo.html:-**
		<!DOCTYPE html>
		<html lang="en">
		<head>
		<meta charset="UTF-8" />
		<meta name="viewport" content="width=device-width, initial-scale=1.0" />
		<title>Custom Iterable Demo</title>
		<style>
		body {
		font-family: Arial, sans-serif;
		background-color: #f5f7fa;
		text-align: center;
		margin: 0;
		padding: 0;
		}
		
		header {
		background-color: #0078d7;
		color: white;
		padding: 15px;
		font-size: 28px;
		}
		
		main {
		margin-top: 30px;
		display: flex;
		flex-direction: column;
		align-items: center;
		gap: 25px;
		}
		
		.box {
		background: white;
		border-radius: 10px;
		box-shadow: 0 0 10px rgba(0,0,0,0.1);
		width: 400px;
		text-align: left;
		padding: 20px;
		}

```
    code {
      display: block;
      background: #f0f0f0;
      padding: 10px;
      border-radius: 6px;
      white-space: pre-wrap;
      font-size: 15px;
    }

    footer {
      margin-top: 40px;
      background: #0078d7;
      color: white;
      padding: 10px;
    }
  </style>
</head>
<body>
  <header>🔁 Custom Iterable: Range(start, end, step)</header>

  <main id="demo"></main>

  <footer>Created by: Prince Patel [24CE092]</footer>

  <script type="module">
    import { Range, testIteration, testSpread } from './iterable.js';

    const demo = document.getElementById("demo");

    const range1 = new Range(1, 5);
    const range2 = new Range(10, 50, 10);
    const range3 = new Range(10, 0, -2);

    demo.innerHTML = `
      <div class="box">
        <h3>1️⃣ for...of Loop</h3>
        <code>
for (const n of new Range(1, 5)) console.log(n);
→ [${testIteration().join(", ")}]
        </code>
      </div>

      <div class="box">
        <h3>2️⃣ Spread Operator</h3>
        <code>
[...new Range(10, 50, 10)]
```

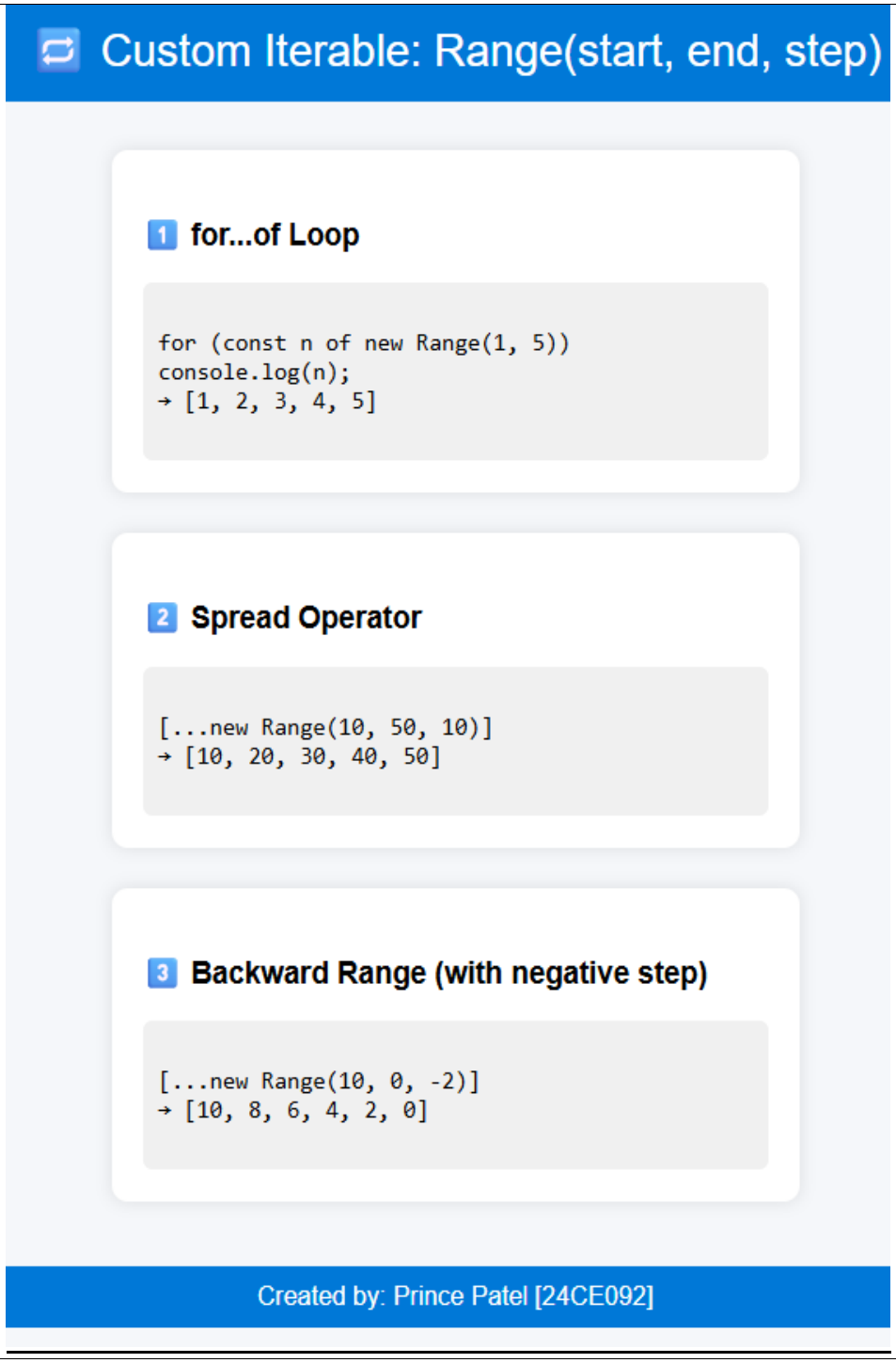
```
→ [${testSpread().join(", ")}]
      </code>
    </div>

    <div class="box">
      <h3> 3 Backward Range (with negative step)</h3>
      <code>
[...new Range(10, 0, -2)]
→ [${[...range3].join(", ")}]
      </code>
    </div>
  `;
 </script>
</body>
</html>
```

iterable.js:-

```
export class Range {
 constructor(start, end, step = 1) {
  if (step === 0) throw new Error("Step cannot be zero");
  this.start = start;
  this.end = end;
  this.step = step;
 }

 [Symbol.iterator]() {
  let current = this.start;
  const end = this.end;
  const step = this.step;

  return {
   next() {
    if ((step > 0 && current <= end) || (step < 0 && current >= end)) {
     const result = { value: current, done: false };
     current += step;
     return result;
    }
    return { done: true };
   }
  };
 }
}

export function testIteration() {
 const range = new Range(1, 5);
```

```
  const values = [];
  for (const num of range) {
    values.push(num);
  }
  return values;
}

export function testSpread() {
  const range = new Range(10, 50, 10);
  return [...range];
}
```

Screenshot:


Custom Iterable: Range(start, end, step)
1 for...of Loop
for (const n of new Range(1, 5))
console.log(n);
→ [1, 2, 3, 4, 5]
2 Spread Operator
[...new Range(10, 50, 10)]
→ [10, 20, 30, 40, 50]
3 Backward Range (with negative step)
[...new Range(10, 0, -2)]
→ [10, 8, 6, 4, 2, 0]
Created by: Prince Patel [24CE092]

9)	Definition :	PHP Form Handling (Server-side Validation)
	Programming concept :	Registration form (name, email, age, password). Requirements: index.php shows form, submit.php validates (required, email format, age ≥ 17, password ≥ 6), sticky form values, success page; no JS validation (server-side only).
	Code:	**Index.php:-** <!DOCTYPE html> <html lang="en"> <head> <meta charset="UTF-8"> <title>Registration Form</title> <link rel="stylesheet" href="style.css"> </head> <body> <header> PHP Registration Form</header> <main> <form action="submit.php" method="post"> <label for="name">Full Name:</label> <input type="text" name="name" id="name" required> <label for="email">Email:</label> <input type="email" name="email" id="email" required> <label for="age">Age:</label> <input type="number" name="age" id="age" required min="1"> <label for="password">Password:</label> <input type="password" name="password" id="password" required> <button type="submit">Register</button> </form> </main> <footer>Created by: Prince Patel [24CE092]</footer> </body> </html> **style.css:-** /* style.css */ body { font-family: Arial, sans-serif; background: #f5f7fa;

```
  margin: 0;
  padding: 0;
}

header {
  background: #0078d7;
  color: white;
  text-align: center;
  padding: 15px;
  font-size: 26px;
}

main {
  display: flex;
  justify-content: center;
  margin-top: 40px;
}

form {
  background: white;
  padding: 25px;
  border-radius: 10px;
  box-shadow: 0 0 10px rgba(0,0,0,0.1);
  width: 400px;
}

label {
  display: block;
  margin-top: 15px;
  font-weight: bold;
}

input[type="text"],
input[type="email"],
input[type="number"],
input[type="password"] {
  width: 100%;
  padding: 8px;
  margin-top: 5px;
  border-radius: 6px;
  border: 1px solid #ccc;
}

button {
  margin-top: 20px;
  width: 100%;
  padding: 10px;
```

```
  background: #0078d7;
  color: white;
  font-size: 16px;
  border: none;
  border-radius: 6px;
  cursor: pointer;
}

button:hover {
  background: #005fa3;
}

.error {
  color: red;
  font-size: 14px;
}

.success {
  color: green;
  text-align: center;
  font-size: 20px;
  background: white;
  margin: 60px auto;
  width: 400px;
  padding: 20px;
  border-radius: 10px;
  box-shadow: 0 0 10px rgba(0,0,0,0.1);
}

footer {
  background: #0078d7;
  color: white;
  text-align: center;
  padding: 10px;
  margin-top: 40px;
}
```

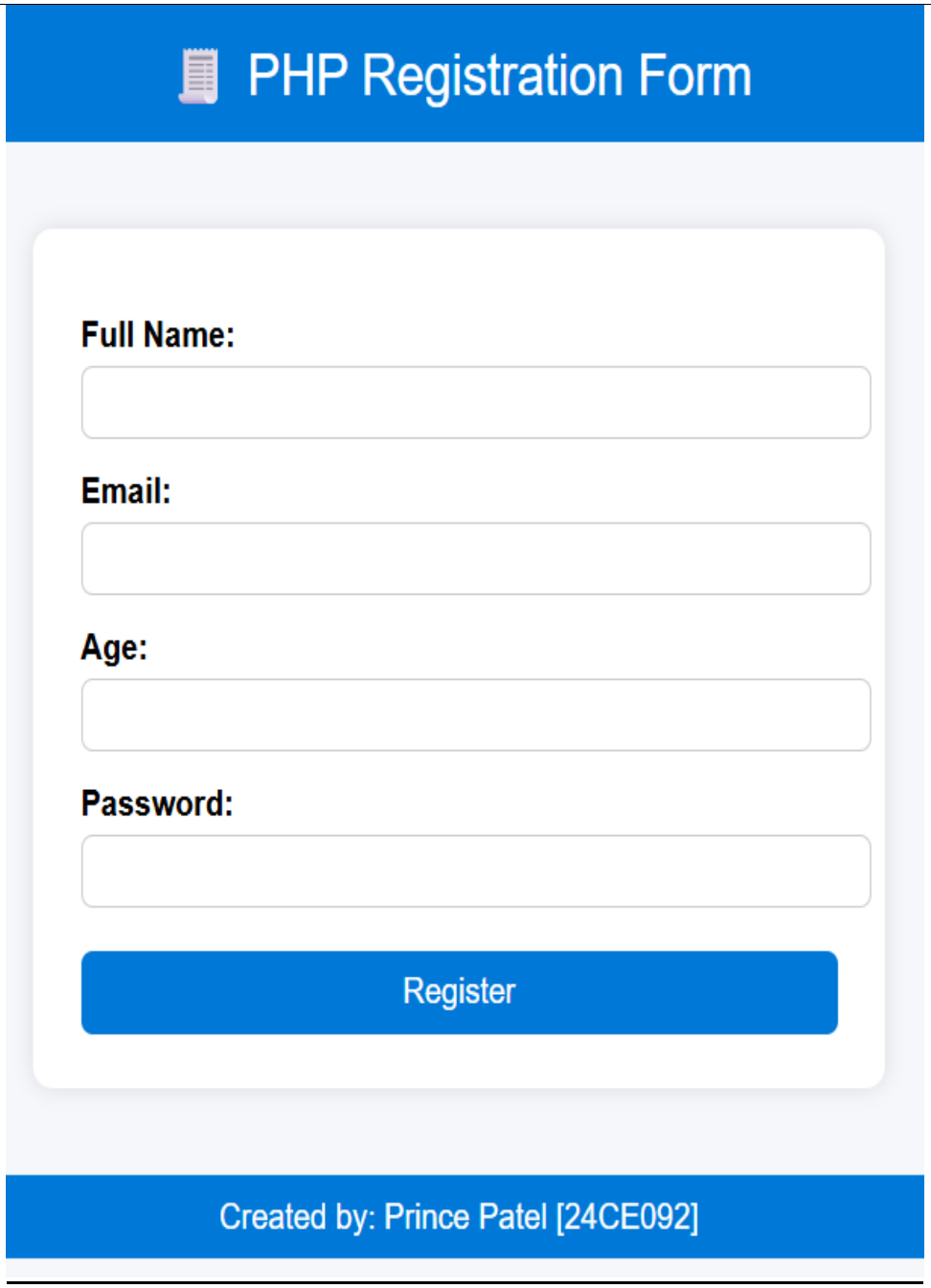
submit.php:-

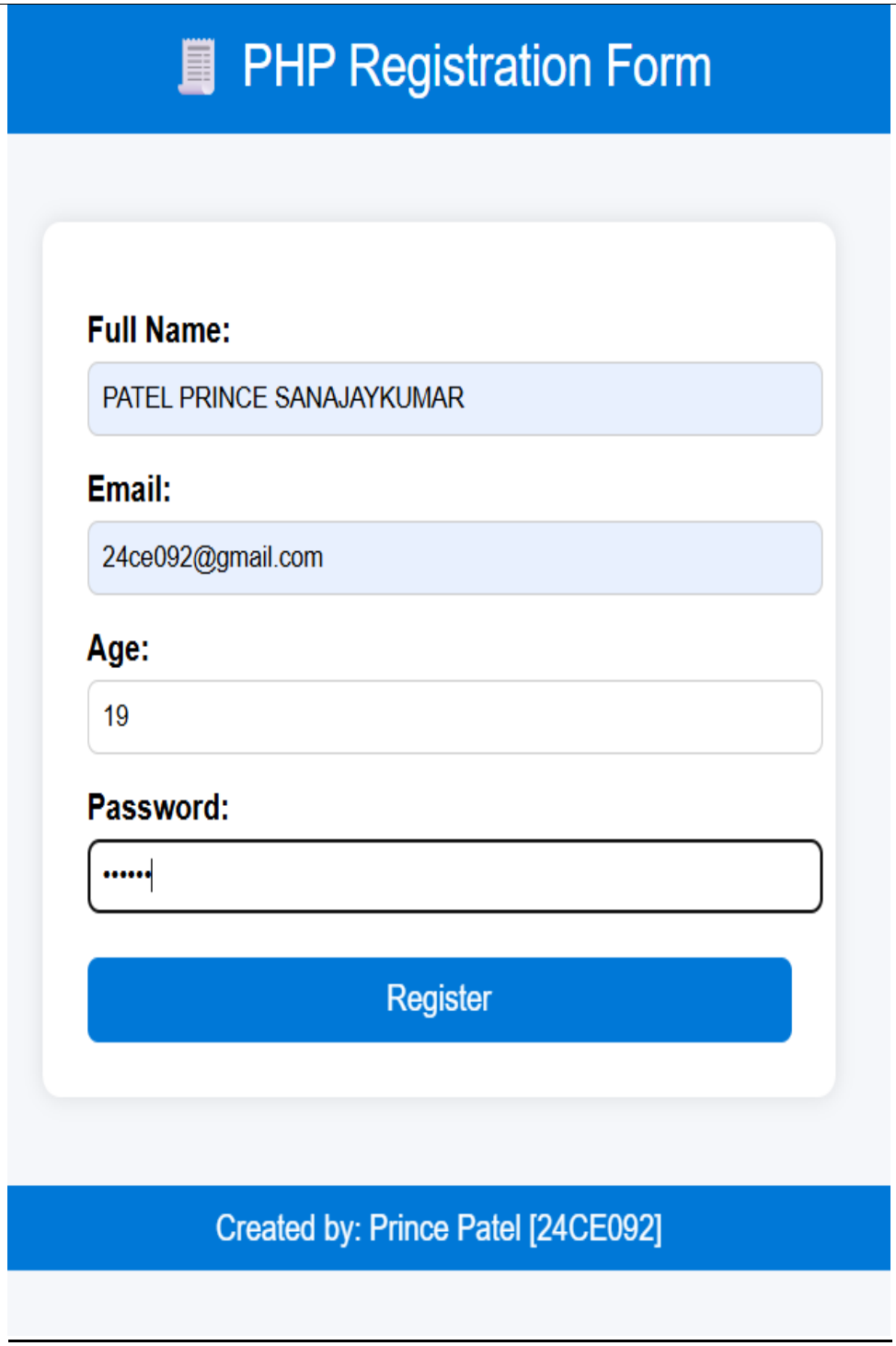
```
<?php
$name = $_POST['name'] ?? '';
$email = $_POST['email'] ?? '';
$age = $_POST['age'] ?? '';
$password = $_POST['password'] ?? '';

$errors = [];

if ($_SERVER["REQUEST_METHOD"] == "POST") {
```

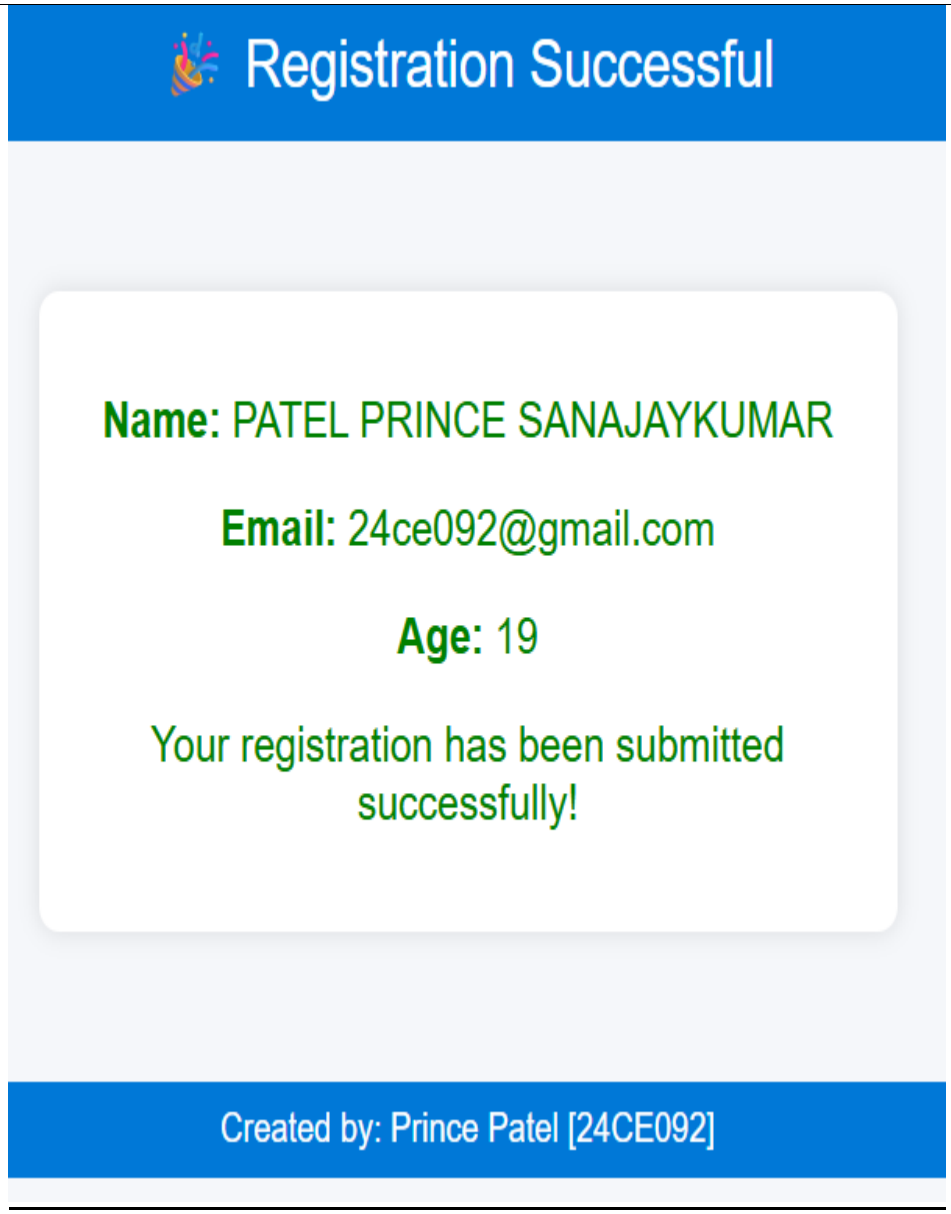
```
  if (empty($name)) {
    $errors['name'] = "Name is required.";
  }

  if (empty($email)) {
    $errors['email'] = "Email is required.";
  } elseif (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
    $errors['email'] = "Invalid email format.";
  }

  if (empty($age)) {
    $errors['age'] = "Age is required.";
  } elseif ($age < 17) {
    $errors['age'] = "You must be at least 17 years old.";
  }

  if (empty($password)) {
    $errors['password'] = "Password is required.";
  } elseif (strlen($password) < 6) {
    $errors['password'] = "Password must be at least 6 characters.";
  }

  if (!empty($errors)) {
    ?>
    <!DOCTYPE html>
    <html lang="en">
    <head>
     <meta charset="UTF-8">
     <title>Form Validation</title>
     <link rel="stylesheet" href="style.css">
    </head>
    <body>
     <header>📜 PHP Registration Form (Validation Errors)</header>
     <main>
      <form action="submit.php" method="post">
       <label for="name">Full Name:</label>
       <input type="text" name="name" id="name" value="<?=
htmlspecialchars($name) ?>">
       <span class="error"><?= $errors['name'] ?? '' ?></span>

       <label for="email">Email:</label>
       <input type="email" name="email" id="email" value="<?=
htmlspecialchars($email) ?>">
       <span class="error"><?= $errors['email'] ?? '' ?></span>

       <label for="age">Age:</label>
```

```
            <input type="number" name="age" id="age" value="<?=
htmlspecialchars($age) ?>">
            <span class="error"><?= $errors['age'] ?? '' ?></span>

            <label for="password">Password:</label>
            <input type="password" name="password" id="password">
            <span class="error"><?= $errors['password'] ?? '' ?></span>

            <button type="submit">Register</button>
          </form>
        </main>
        <footer>Created by: Prince Patel [24CE092]</footer>
      </body>
      </html>
      <?php
    } else {
      ?>
      <!DOCTYPE html>
      <html lang="en">
      <head>
        <meta charset="UTF-8">
        <title>Registration Successful</title>
        <link rel="stylesheet" href="style.css">
      </head>
      <body>
        <header>🎉 Registration Successful</header>
        <div class="success">
          <p><strong>Name:</strong> <?= htmlspecialchars($name) ?></p>
          <p><strong>Email:</strong> <?= htmlspecialchars($email) ?></p>
          <p><strong>Age:</strong> <?= htmlspecialchars($age) ?></p>
          <p>Your registration has been submitted successfully!</p>
        </div>
        <footer>Created by: Prince Patel [24CE092]</footer>
      </body>
      </html>
      <?php
    }
}
?>
```

Screenshot:


PHP Registration Form
Full Name:
Email:
Age:
Password:
Register
Created by: Prince Patel [24CE092]


PHP Registration Form
Full Name:
PATEL PRINCE SANAJAYKUMAR
Email:
24ce092@gmail.com
Age:
19
Password:
Register
Created by: Prince Patel [24CE092]


Registration Successful
Name: PATEL PRINCE SANAJAYKUMAR
Email: 24ce092@gmail.com
Age: 19
Your registration has been submitted successfully!
Created by: Prince Patel [24CE092]

10)	Definition :	PHP Sessions & Cookies (Login Mini-App)
	Programming concept :	Simple login/logout with session; remember email via cookie. Requirements: Hardcode 1 user (for demo), login.php sets session on success, dashboard.php greets user, logout.php destroys session; "Remember me" sets a cookie

	Code:	**Login.php:-**
		`<?php`
		`session_start();`
		
		`$users = [`
		`"admin" => ["password" => "admin@123", "role" => "admin"],`
		`"student1" => ["password" => "stud123", "role" => "user"],`
		`"student2" => ["password" => "pass321", "role" => "user"]`
		`];`
		
		`if (isset($_GET['action']) && $_GET['action'] === "logout") {`
		`session_unset();`
		`session_destroy();`
		`header("Location: ?page=login");`
		`exit;`
		`}`
		
		`$error = "";`
		`if ($_SERVER["REQUEST_METHOD"] == "POST") {`
		`$username = $_POST['username'];`
		`$password = $_POST['password'];`
		
		`if (isset($users[$username]) && $users[$username]['password'] === $password) {`
		`$_SESSION['username'] = $username;`
		`$_SESSION['role'] = $users[$username]['role'];`
		`header("Location: ?page=dashboard");`
		`exit;`
		`} else {`
		`$error = "Invalid username or password!";`
		`}`
		`}`
		
		`$page = $_GET['page'] ?? "login";`
		`?>`
		
		`<!DOCTYPE html>`
		`<html>`
		`<head>`
		`<title>Simple Auth System</title>`

```
</head>
<body>

<?php if ($page === "login" && !isset($_SESSION['username'])): ?>
  <h2>Login</h2>
  <form method="POST">
    Username: <input type="text" name="username" required><br><br>
    Password: <input type="password" name="password"
required><br><br>
    <button type="submit">Login</button>
  </form>
  <p style="color:red;"><?php echo $error; ?></p>

<?php elseif ($page === "dashboard" && isset($_SESSION['username'])):
?>
  <h2>Welcome, <?php echo $_SESSION['username']; ?>!</h2>
  <p>Your role: <?php echo $_SESSION['role']; ?></p>
  <a href="?action=logout">Logout</a>

<?php else: ?>
  <?php header("Location: ?page=login"); exit; ?>
<?php endif; ?>

</body>
</html>
```

Logout.php:-

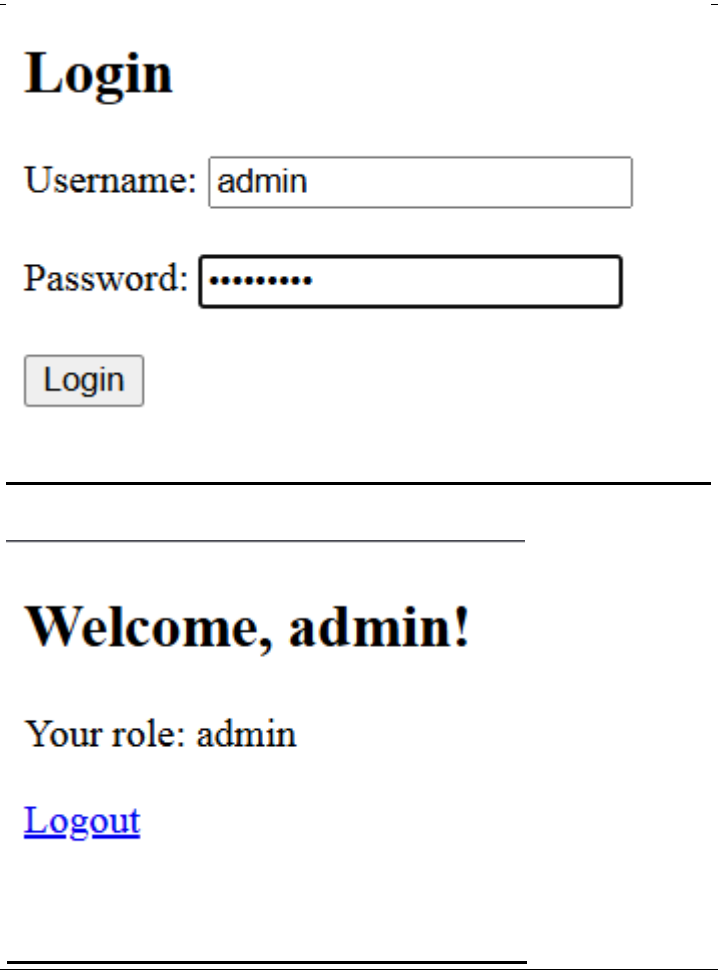
```
<?php
session_start();
session_unset();
session_destroy();
header("Location: login.php");
exit;
?>
```

Dashboard.php:-

```
<?php
session_start();
if (!isset($_SESSION['username'])) {
  header("Location: login.php");
  exit;
}
?>
```

```
<!DOCTYPE html>
<html>
<head><title>Dashboard</title></head>
<body>
   <h2>Welcome, <?php echo $_SESSION['username']; ?>!</h2>
   <p>Your role: <?php echo $_SESSION['role']; ?></p>
   <a href="logout.php">Logout</a>
</body>
</html>
```

Screenshot:



11)	Definition :	PHP File I/O: Guestbook
	Programming concept :	Append guest entries to a text/CSV file and list them. Requirements: guestbook.php (form), save.php (append), list.php (read & display); file locking; basic sanitization
	Code:	**Guestbook.php:-** <!DOCTYPE html> <html lang="en"> <head> <meta charset="UTF-8"> <title>Guestbook</title> <link rel="stylesheet" href="style.css"> </head> <body> 📖 Guestbook <form action="save.php" method="post"> <label>Name:</label> <input type="text" name="name" required> <label>Message:</label> <textarea name="message" rows="4" cols="40" required></textarea> <button type="submit">Submit</button> </form> View Guest Entries → </body> </html> **List.php:-** <?php $file = 'guestbook.txt'; ?> <!DOCTYPE html> <html lang="en"> <head> <meta charset="UTF-8"> <title>Guestbook Entries</title> <link rel="stylesheet" href="style.css"> </head> <body> 📋 Guestbook Entries <?php

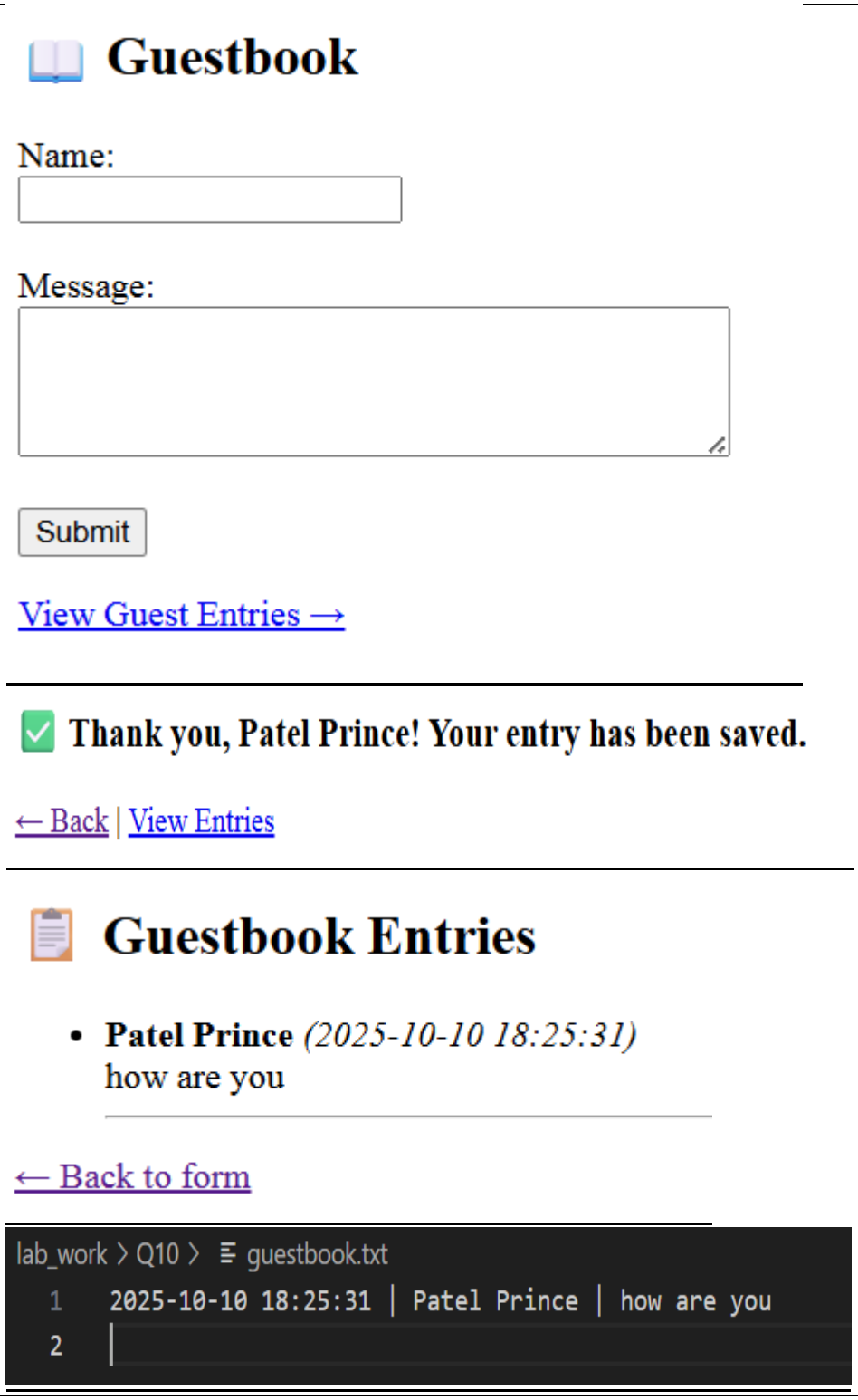
```
 if (file_exists($file)) {
    $lines = file($file, FILE_IGNORE_NEW_LINES |
FILE_SKIP_EMPTY_LINES);
    if (count($lines) > 0) {
       echo "<ul>";
       foreach (array_reverse($lines) as $line) { // newest first
          [$date, $name, $message] = explode(" | ", $line);
          echo "<li><strong>$name</strong>
<em>($date)</em><br>$message</li><hr>";
       }
       echo "</ul>";
    } else {
       echo "<p>No entries yet.</p>";
    }
 } else {
    echo "<p>No entries yet.</p>";
 }
 ?>

 <p><a href="guestbook.php">← Back to form</a></p>
</body>
</html>
```

Save.php:-

```
<?php
$name = trim(htmlspecialchars($_POST['name'] ?? ''));
$message = trim(htmlspecialchars($_POST['message'] ?? ''));

if ($name === '' || $message === '') {
   die(" ⚠️ Please fill out all fields. <a href='guestbook.php'>Go
back</a>");
}
$file = 'guestbook.txt';
$entry = date("Y-m-d H:i:s") . " | " . $name . " | " . $message . PHP_EOL;

if ($fp = fopen($file, 'a')) {
   if (flock($fp, LOCK_EX)) {
      fwrite($fp, $entry);
      fflush($fp);
      flock($fp, LOCK_UN);
   }
   fclose($fp);
}

echo "<h3> ✅ Thank you, $name! Your entry has been saved.</h3>";
```

```
echo "<a href='guestbook.php'>← Back</a> | <a href='list.php'>View Entries</a>";
?>
```

Screenshot:



12)	Definition :	PHP OOP Mini-Project
	Programming concept :	Build Product class with properties, constructor, methods (getFinalPrice(tax, discount)), and a DigitalProduct subclass (no shipping). Requirements: Encapsulation (getters/setters), inheritance, method override, small demo script.

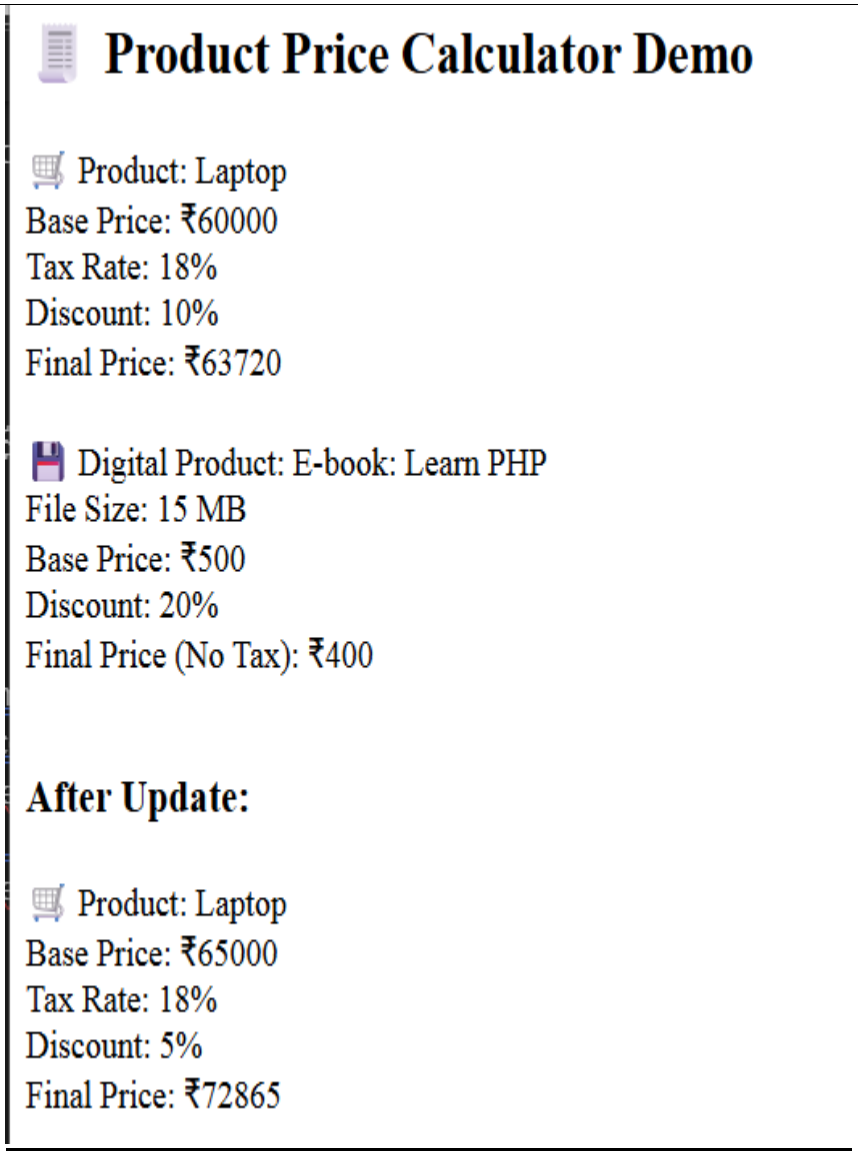
	Code:	**Demo.php:-**
		`<?php`
		`require_once "Product.php";`
		`require_once "DigitalProduct.php";`
		
		`$physical = new Product("Laptop", 60000, 18, 10);`
		`$digital = new DigitalProduct("E-book: Learn PHP", 500, 0, 20, 15);`
		
		`echo "<h2> 📜 Product Price Calculator Demo</h2>";`
		`$physical->displayInfo();`
		`$digital->displayInfo();`
		
		`$physical->setPrice(65000);`
		`$physical->setDiscount(5);`
		
		`echo "<h3>After Update:</h3>";`
		`$physical->displayInfo();`
		`?>`
		
		Digitalproduct.php:-
		`<?php`
		`require_once "Product.php";`
		
		`class DigitalProduct extends Product {`
		`  private $fileSize;`
		
		`  public function __construct($name, $price, $taxRate, $discount, $fileSize) {`
		`  parent::__construct($name, $price, $taxRate, $discount);`
		`  $this->fileSize = $fileSize;`
		`  }`
		
		`  public function getFileSize() {`
		`  return $this->fileSize;`
		`  }`
		
		`  public function setFileSize($fileSize) {`
		`  $this->fileSize = $fileSize;`
		`  }`

```
    public function getFinalPrice() {
        $priceAfterDiscount = $this->getPrice() - ($this->getPrice() * $this->getDiscount() / 100);
        return round($priceAfterDiscount, 2);
    }

    public function displayInfo() {
        echo "💾 Digital Product: {$this->getName()}<br>";
        echo "File Size: {$this->fileSize} MB<br>";
        echo "Base Price: ₹{$this->getPrice()}<br>";
        echo "Discount: {$this->getDiscount()}%<br>";
        echo "Final Price (No Tax): ₹" . $this->getFinalPrice() . "<br><br>";
    }
}
?>
```

Product.php:-

```
<?php
class Product {
    private $name;
    private $price;
    private $taxRate;
    private $discount;

    public function __construct($name, $price, $taxRate = 0, $discount = 0) {
        $this->name = $name;
        $this->price = $price;
        $this->taxRate = $taxRate;
        $this->discount = $discount;
    }

    public function getName() {
        return $this->name;
    }

    public function setName($name) {
        $this->name = $name;
    }

    public function getPrice() {
        return $this->price;
    }

    public function setPrice($price) {
```

```
        $this->price = $price;
    }

    public function getTaxRate() {
        return $this->taxRate;
    }

    public function setTaxRate($taxRate) {
        $this->taxRate = $taxRate;
    }

    public function getDiscount() {
        return $this->discount;
    }

    public function setDiscount($discount) {
        $this->discount = $discount;
    }

    public function getFinalPrice() {
        $priceWithTax = $this->price + ($this->price * $this->taxRate / 100);
        $finalPrice = $priceWithTax - ($priceWithTax * $this->discount / 100);
        return round($finalPrice, 2);
    }

    public function displayInfo() {
        echo "🛒 Product: {$this->name}<br>";
        echo "Base Price: ₹{$this->price}<br>";
        echo "Tax Rate: {$this->taxRate}%<br>";
        echo "Discount: {$this->discount}%<br>";
        echo "Final Price: ₹" . $this->getFinalPrice() . "<br><br>";
    }
}
?>
```

	Screenshot:	 Product Price Calculator Demo Product: Laptop Base Price: ₹60000 Tax Rate: 18% Discount: 10% Final Price: ₹63720 Digital Product: E-book: Learn PHP File Size: 15 MB Base Price: ₹500 Discount: 20% Final Price (No Tax): ₹400 After Update: Product: Laptop Base Price: ₹65000 Tax Rate: 18% Discount: 5% Final Price: ₹72865

13)	Definition :	MySQL CRUD with PHP (PDO or MySQLi)
	Programming concept :	"Students" CRUD: list, add, edit, delete. Requirements: DB schema (students(id, name, email, branch)), parameterized queries (prepared statements), server-side validation, basic pagination

	Code:	**Config.php:-**
		`<?php`
		`$localhost="localhost";`
		`$username="root";`
		`$password="";`
		`$dbname="schema";`
		
		`$conn = new mysqli($localhost,$username,$password,$dbname);`
		
		`$sql = "CREATE TABLE IF NOT EXISTS student (`
		`id INT AUTO_INCREMENT PRIMARY KEY,`
		`name VARCHAR(100),`
		`email VARCHAR(100) UNIQUE,`
		`branch VARCHAR(100)`
		`)";`
		
		`$conn->query($sql);`
		
		`if($conn->connect_error){`
		`die("EError : ".$conn->connect_error);`
		`}`
		`echo"connection successfully";`
		`?>`
		
		Create.php:-
		`<?php`
		
		`include "config.php";`
		
		`if (isset($_POST["insert"])) {`
		`$name = $_POST["name"];`
		`$email = $_POST["email"];`
		`$branch = $_POST["branch"];`
		
		`$stmt = $conn->prepare("INSERT INTO student(name, email, branch) VALUES (?, ?, ?)");`
		`$stmt->bind_param("sss", $name, $email, $branch);`
		
		`if ($stmt->execute()) {`

```
        echo "<br><h4>Student data inserted successfully</h4>";
    } else {
        echo "Error: " . $stmt->error;
    }
}
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Create</title>
</head>
<body>
    <h3>Student Data Insert</h3>
    <form action="create.php" method="POST">
      Name: <input type="text" placeholder="name" name="name"
required><br><br>
      email: <input type="email" placeholder="email" name="email"
required><br><br>
      branch: <input type="text" placeholder="branch" name="branch"
required><br><br>

    <button type="submit" name="insert"> add data </button>
    </form>
</body>
</html>
```

Edit.php:-

```
<?php

include "config.php";

if (isset($_POST["update"])) {
    $id = $_POST["id"];
    $name = $_POST["name"];
    $email = $_POST["email"];
    $branch = $_POST["branch"];

    $stmt = $conn->prepare("UPDATE student SET name = ?, email = ?,
branch = ? WHERE id = ?");
    $stmt->bind_param("isss", $id,$name, $email, $branch);

    if ($stmt->execute()) {
```

```
        echo "<br><h3>Student data updated successfully</h3>";
    } else {
        echo "Error: " . $stmt->error;
    }
}
?>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Document</title>
</head>
<body>

<form action="edit.php" method="POST">
    id: <input type="number" name="id" required><br><br>
    Name: <input type="text" name="name"  required><br><br>
    Email: <input type="text" name="email" required><br><br>
    Branch: <input type="text" name="branch" required><br><br>
    <button type="submit" name="update">Update</button>
</form>
</body>
</html>
```

Delete.php:-

```
<?php
include "config.php";
if (isset($_POST['delete'])) {
    $id = $_POST['id'];

    $stmt = $conn->prepare("DELETE FROM student WHERE id=?");

    $stmt->bind_param( "i",$id);

    if ($stmt->execute()) {
        echo "<p> student deleted successfully!</p>";
    } else {
        echo "<p> Error: " . $stmt->error . "</p>";
    }
}
?>

<!DOCTYPE html>
<html lang="en">
<head>
```

```
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Document</title>
</head>
<body>

<form action="delete.php" method="POST">
    id: <input type="number" name="id" required><br><br>
    <button type="submit" name="delete">Update</button>
</form>


</body>
</html>
```

list.php:-

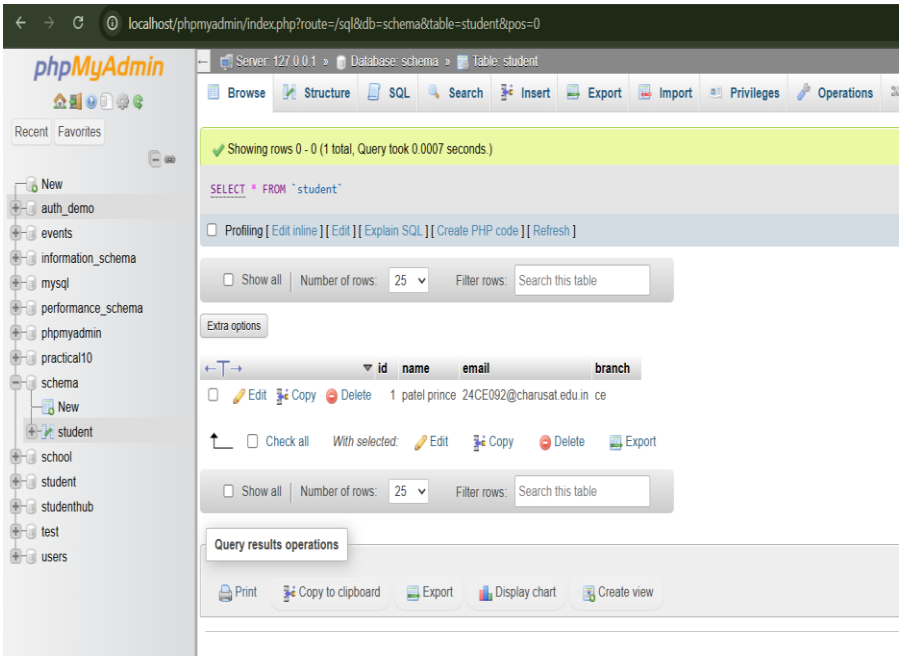
```
<?php
include "config.php"; // assumes $conn is defined here

$sql = "SELECT id, name, email, branch FROM student ORDER BY id ASC";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
    echo "<h3>Student List</h3>";
    echo "<table border='1' cellpadding='10'>";
    echo
"<tr><th>ID</th><th>Name</th><th>Email</th><th>Branch</th></tr>";

    while ($row = $result->fetch_assoc()) {
        echo "<tr>";
        echo "<td>" . htmlspecialchars($row["id"]) . "</td>";
        echo "<td>" . htmlspecialchars($row["name"]) . "</td>";
        echo "<td>" . htmlspecialchars($row["email"]) . "</td>";
        echo "<td>" . htmlspecialchars($row["branch"]) . "</td>";
        echo "</tr>";
    }

    echo "</table>";
} else {
    echo "No student records found.";
}
?>
```

	Screenshot:	connection successfully **Student Data Insert** Name: prince patel email: 24CE092@charusat.edu.in branch: ce add data --- **Student data inserted successfully** **Student Data Insert** Name: name email: email branch: branch add data --- id: Delete ---

student deleted successfully!

id:

Delete



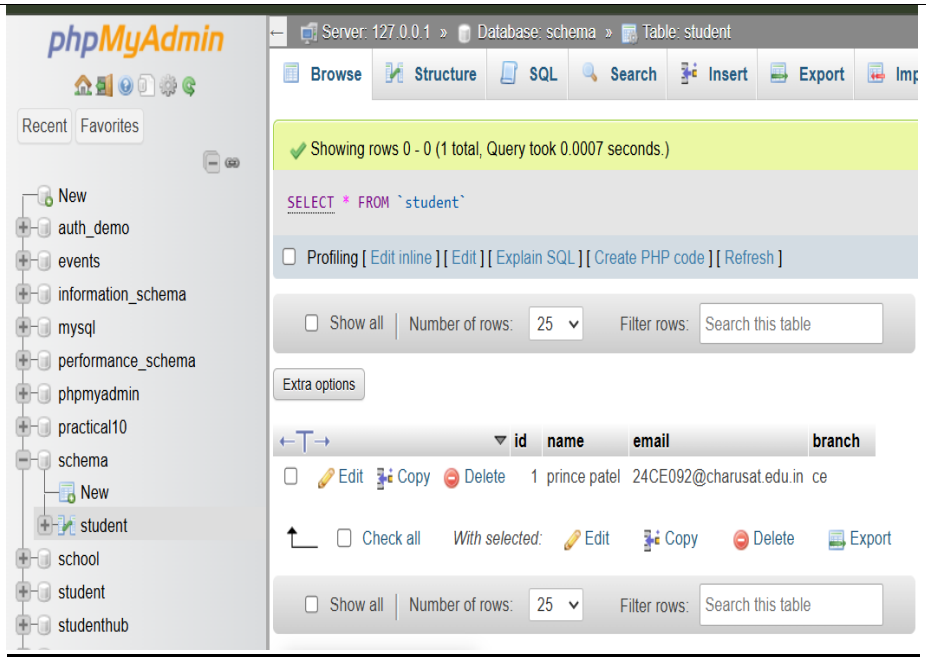
id: 1

Name: prince patel

Email: 24CE092@charusat.edu.in

Branch: ce

Update



Student List

ID	Name	Email	Branch
1	prince patel	24CE092@charusat.edu.in	ce