

1. Web Page using all HTML Elements.

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Supermarket Online Store</title>
</head>
<body>
  <!-- Header Section -->
  <header>
    <h1><i><b>Supermarket Online Store</b></i></h1><br>
    <nav>
      <ul>
        <li><a href="#">Home</a></li>
        <li><a href="#">Shop</a></li>
        <li><a href="#">About Us</a></li>
      </ul>
    </nav>
  </header>

  <!-- Main Section -->
  <main style="text-align: center;">
    <h2><b>Welcome to Our Online Store</b></h2>
    <p>We offer a wide range of products to meet your everyday needs.</p>

    <!-- Product Categories Section -->
    <section>
      <h3><i>Product Categories</i></h3><br>
      <nav>
        <ul>
          <li><a href="#">Fresh Produce</a></li>
          <li><a href="#">Meat and Poultry</a></li>
          <li><a href="#">Dairy and Eggs</a></li>
          <li><a href="#">Bakery and Bread</a></li>
        </ul>
      </nav>
    </section>

    <!-- Featured Products Section -->
    <section><br><br>
    <h3><b>Featured Products</b></h3>
    <table>
      <tr>
        <td>
          <figure>
            
            <figcaption>Product Name</figcaption>
          </figure>
          <p>Product Description</p>
          <button>Add to Cart</button>
        </td>
        <td>
```

```

        <figure>
            
            <figcaption>Product Name</figcaption>
        </figure>
        <p>Product Description</p>
        <button>Add to Cart</button>
    </td>
    <td>
        <figure>
            
            <figcaption>Product Name</figcaption>
        </figure>
        <p>Product Description</p>
        <button>Add to Cart</button>
    </td>
</tr>
</table>
</section>

<!-- Contact Us Section -->
<section>
    <br><br>
    <h3><i>Contact Us</i></h3>
    <br>
    <form>
        <label for="name">Name:</label>
        <input type="text" id="name" name="name" required><br>

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

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

        <button type="submit">Send Message</button><br><br>
    </form>
</section>
</main>

<!-- Footer Section -->
<footer style="text-align: center;">
    <nav>
        <ul>
            <li><a href="#">Shipping and Delivery</a></li>
            <li><a href="#">Returns and Exchanges</a></li>
        </ul>
    </nav>
    <p>&copy; 2023 Supermarket Online Store. All rights reserved.</p>
</footer>

<style>

body {
    background-color: #f5f5f5;
    font-size: 16px;
    background-color: darkseagreen;
}

```

```

h1, h2, h3 {
    margin: 0;
    font-weight: normal;
    text-align: center;
}

nav ul {
    margin: 0;
    padding: 0;
    list-style: none;
    text-align: center;
}

nav ul li {
    display: inline-block;
    margin-right: 20px;
    text-align: center;
}

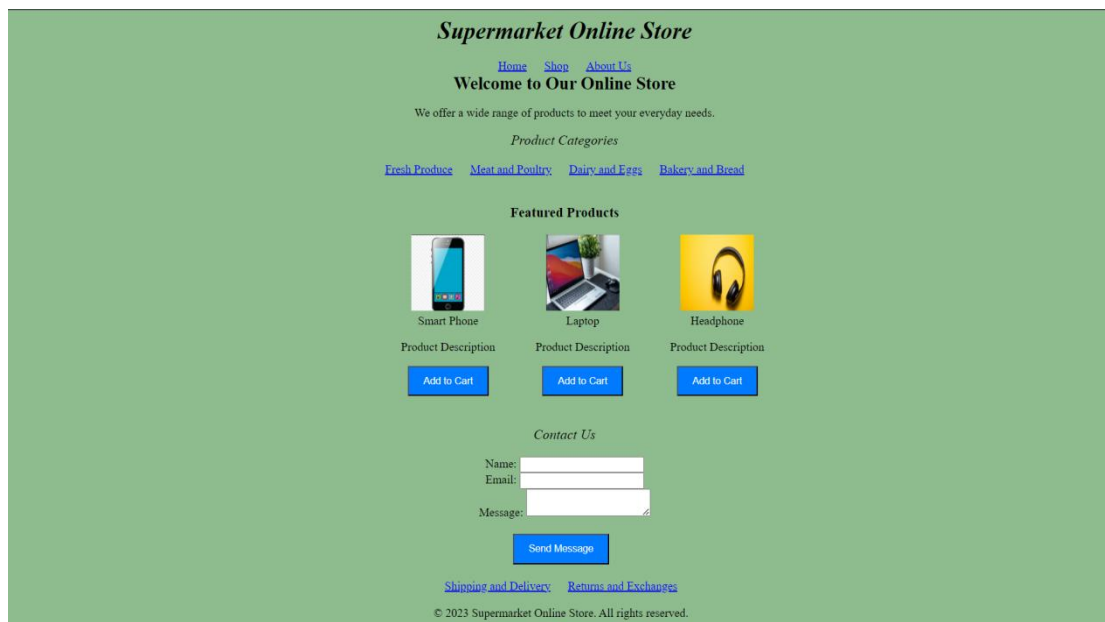
nav ul li:last-child {
    margin-right: 0;
    text-align: center;
}

button {
    background-color: #007bff;
    color: #fff;
    padding: 10px 20px;
    cursor: pointer;
}

table{
    margin-left: 500px;
}
</style>
</html>

```

Output



2. Create a web page with all types of Cascading style sheets and CSS Selectors

index.html

```
<!DOCTYPE html>
<html>
<head>
  <title>ABC Educational Institution</title>
  <link rel="stylesheet" href="style.css">
  <!-- Inline CSS -->
  <style>
    body {
      background-color: #f2f2f2;
    }

    h1 {
      color: #333;
      text-align: center;
    }

    p {
      font-size: 18px;
      line-height: 1.5;
    }
  </style>
  <!-- Internal CSS -->
  <style>
    .container {
      max-width: 960px;
      margin: 0 auto;
      padding: 20px;
      background-color: #fff;
      box-shadow: 0 0 10px rgba(0,0,0,0.2);
    }

    .section-header {
      font-size: 28px;
      margin-top: 50px;
      margin-bottom: 20px;
      text-align: center;
      color: #333;
    }

    .program {
      margin-bottom: 30px;
    }

    .program h3 {
      margin: 0;
      color: #333;
    }
    .program p {
      margin: 10px 0;
      font-size: 18px;
      line-height: 1.5;
    }
  </style>
</head>
<body>
  <div class="container">
    <h1 class="section-header">ABC Educational Institution</h1>
    <div class="program">
      <h3>Program 1</h3>
      <p>Description of Program 1</p>
    </div>
  </div>
</body>
</html>
```

```

</style>
<!-- External CSS -->
<link rel="stylesheet" type="text/css" href="style.css">
</head>
<body>
  <header>
    <h1 style="color: white;">ABC Educational Institution</h1>
    <nav>
      <ul>
        <li><a href="#">Home</a></li>
        <li><a href="#">Programs</a></li>
        <li><a href="#">Admissions</a></li>
        <li><a href="#">About</a></li>
      </ul>
    </nav>
  </header>

  <div class="container">
    <section id="hero">
      <h2>Welcome to ABC Institution</h2>
      <p>We offer a wide range of programs and opportunities</p>
      <button class="button">Apply Now</button>
    </section>

    <section id="programs">
      <h2 class="section-header">Our Programs</h2>
      <div class="program">
        <h3>Computer Science</h3>
        <p>Learn programming languages</p>
      </div>
      <div class="program">
        <h3>Business Administration</h3>
        <p>Develop your management and leadership skills.</p>
      </div>
      <div class="program">
        <h3>Nursing</h3>
        <p>Care for patients and promote health and wellness.</p>
      </div>
    </section>

    <section id="admissions">
      <h2 class="section-header">Admissions</h2>
      <p>Please submit your application and transcripts.</p>
    </section>

    <section id="about">
      <h2 class="section-header">About Us</h2>
      <p>We are a leading institution of higher education</p>
    </section>
  </div>

  <footer>
    <p>&copy; My Institution 2023</p>
  </footer>
</body>
</html>

```

style.css

```
/* Styles for header */
header {
    background-color: #333;
    color: #fff;
    padding: 20px;
}

header h1 {
    margin: 0;
    font-size: 36px;
}

nav {
    margin-top: 20px;
}

nav ul {
    list-style: none;
    margin: 0;
    padding: 0;
    display: flex;
    justify-content: center;
}

nav li {
    margin: 0 10px;
}

nav a {
    display: block;
    padding: 10px;
    color: #fff;
    text-decoration: none;
}

nav a:hover {
    background-color: #fff;
    color: #333;
}

/* Styles for hero section */
#hero {
    background-image: url('https://picsum.photos/960/400');
    background-size: cover;
    background-position: center;
    color: #fff;
    padding: 100px 20px;
    text-align: center;
}

#hero h2 {
    margin-top: 0;
    font-size: 48px;
}

#hero p {
    font-size: 24px;
```

```

        margin-bottom: 30px;
    }

    .button {
        background-color: #fff;
        color: #333;
        padding: 10px 20px;
        border: none;
        border-radius: 5px;
        font-size: 18px;
        cursor: pointer;
        transition: background-color 0.3s ease;
    }

    .button:hover {
        background-color: #333;
        color: #fff;
    }

    /* Styles for programs section */
    #programs {
        padding: 50px 20px;
    }

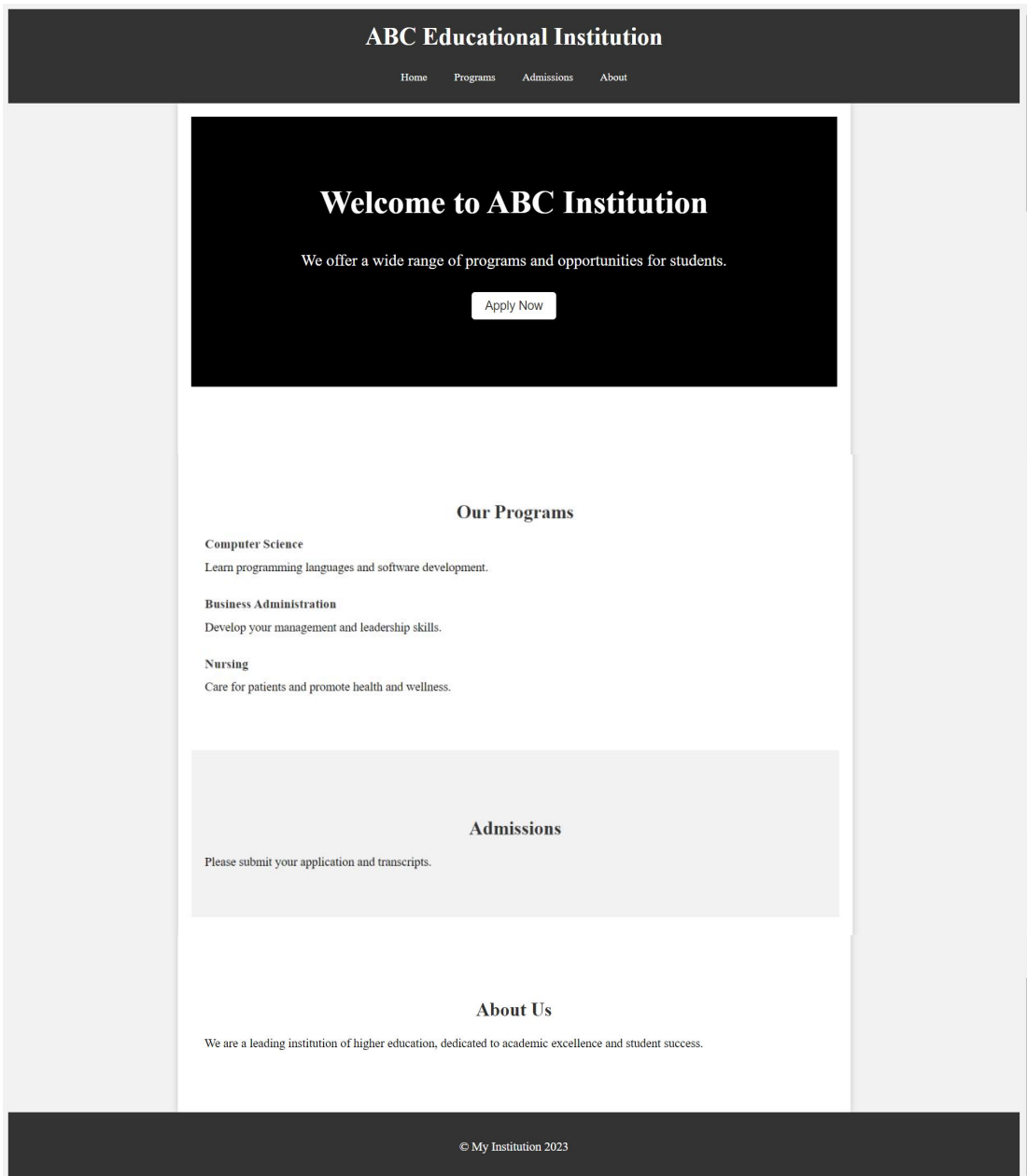
    /* Styles for admissions section */
    #admissions {
        background-color: #f2f2f2;
        padding: 50px 20px;
    }

    /* Styles for about section */
    #about {
        padding: 50px 20px;
    }

    /* Styles for footer */
    footer {
        background-color: #333;
        color: #fff;
        text-align: center;
        padding: 20px;
        font-size: 16px;
    }

```

Output



3. Write Client-Side Scripts for Validating Web Form Controls using DHTML

```
<!DOCTYPE html>
<html>
<head>
  <title>Form Validation using DHTML</title>
  <style>
    body {
      background-color: #f0f0f0;
    }

    form {
      width: 400px;
      height: 250px;
      margin: auto;
      margin-top: 200px;
      padding: 20px;
      background-color: white;
      border: 1px solid #ccc;
      border-radius: 5px;
      box-shadow: 0px 0px 5px #ccc;
    }

    label {
      display: block;
      margin-bottom: 10px;
    }

    input[type="text"], input[type="email"] {
      width: 100%;
      padding: 5px;
      margin-bottom: 10px;
      border: 1px solid #ccc;
      border-radius: 5px;
      box-sizing: border-box;
    }

    button[type="submit"] {
      display: block;
      margin-top: 20px;
      padding: 10px;
      background-color: #4CAF50;
      color: white;
      border: none;
      border-radius: 5px;
      cursor: pointer;
    }

    button[type="submit"]:hover {
      background-color: #3e8e41;
    }

    #title{
      margin-left: 500px;
    }
  </style>
</head>
<body>
```

```

<h1 id="title"><b>Form validation using Dynamic HTML</b></h1>
  <form name="myForm" onsubmit="return validateForm()">
    <label for="firstName">First Name:</label>
    <input type="text" id="firstName" name="firstName"><br>

    <label for="lastName">Last Name:</label>
    <input type="text" id="lastName" name="lastName"><br>

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

    <button type="submit">Submit</button>
  </form>
  <script type="text/javascript">
    function validateForm() {
      // Get the values of the form elements
      var firstName = document.forms["myForm"]["firstName"].value;
      var lastName = document.forms["myForm"]["lastName"].value;
      var email = document.forms["myForm"]["email"].value;

      // Validate the first name
      if (firstName == "") {
        alert("Please enter your first name");
        return false;
      }

      // Validate the last name
      else if (lastName == "") {
        alert("Please enter your last name");
        return false;
      }

      // Validate the email address
      else if (email == "") {
        alert("Please enter your email address");
        return false;
      }
      else if (!isValidEmail(email)) {
        alert("Please enter a valid email address");
        return false;
      }
      else{
        alert("Form submitted successfully");
      }

      return true;
    }

    function isValidEmail(email) {
      // Regular expression to validate email address
      var emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
      return emailRegex.test(email);
    }
  </script>
</body>
</html>

```

Output

Form validation using Dynamic HTML

First Name:

Last Name:

Email:

4. Design the following using JavaScript and DOM

a. Image Slideshow

Index.html

```
<!DOCTYPE html>
<html>
<head>
  <title>Image Slideshow</title>
  <style>
    .slideshow-container {
      position: relative;
      width: 100%;
      height: 400px;
      margin-top: 100px;
      overflow: hidden;
    }

    .slide {
      display: none;
      position: absolute;
      top: 0;
      left: 0;
      margin-left: 200px;
      width: 70%;
      height: 100%;
    }

    .slide img {
      width: 100%;
      height: 100%;
      object-fit: cover;
    }

    #prevButton,
    #nextButton {
      display: inline-block;
      padding: 10px 20px;
      margin-left: 440px;
      font-size: 16px;
      border: none;
      background-color: #ddd;
      color: #333;
      cursor: pointer;
    }

    #prevButton:hover,
    #nextButton:hover {
      background-color: #ccc;
    }
  </style>
</head>
<body style="background-color: #cabebe;">
  <h1 style="margin-left: 600px;">Image Slide Show</h1>
  <div class="slideshow-container">
    <div class="slide">
      
    </div>
  </div>
</body>
</html>
```

```

</div>
<div class="slide">
  
</div>
<div class="slide">
  
</div>
</div>

<button id="prevButton">Previous</button>
<button id="nextButton">Next</button>

<script>
  document.addEventListener("DOMContentLoaded", function(event) {
    var slides = document.getElementsByClassName("slide");
    var slideIndex = 0;

    function showSlide() {
      // Hide all slides
      for (var i = 0; i < slides.length; i++) {
        slides[i].style.display = "none";
      }

      // Show the current slide
      slides[slideIndex].style.display = "block";
    }

    function nextSlide() {
      slideIndex++;
      if (slideIndex >= slides.length) {
        slideIndex = 0; // Reset to the first slide
      }
      showSlide();
    }

    function prevSlide() {
      slideIndex--;
      if (slideIndex < 0) {
        slideIndex = slides.length - 1; // Go to the last slide
      }
      showSlide();
    }

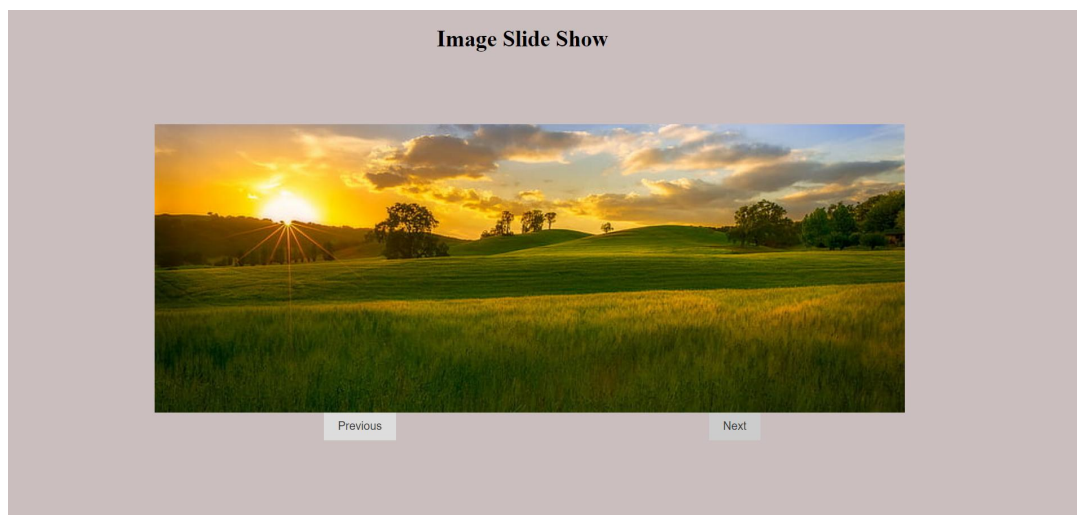
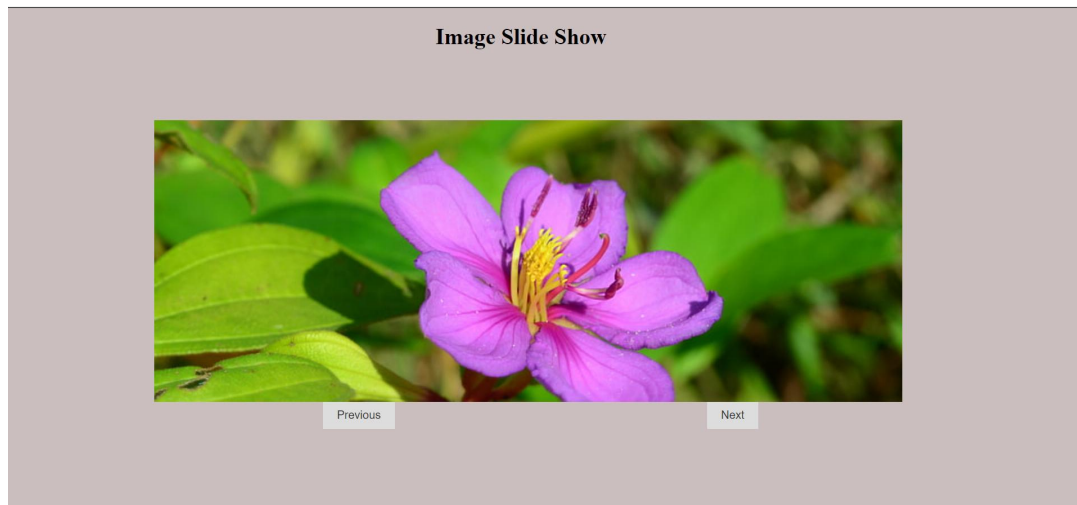
    // Show the first slide
    showSlide();

    // Event listener for next button
    var nextButton = document.getElementById("nextButton");
    nextButton.addEventListener("click", nextSlide);

    // Event listener for previous button
    var prevButton = document.getElementById("prevButton");
    prevButton.addEventListener("click", prevSlide);
  });
</script>
</body>
</html>

```

Output



b. Digital clock

Index.html

```
<!DOCTYPE html>
<html>

<head>
  <meta charset="UTF-8">
  <title>Digital Clock App</title>
  <style>
    *{
      padding: 0;
      margin: 0;
      box-sizing: border-box;
    }

    body{
      background-color: black;
    }

    .clock{
      text-decoration: double;
      font-size: 50px;
      color: white;
      margin-left: 0px;
      text-align: center;
      margin-top: 10px;
    }

    .container{
      height: 150px;
      width: 400px;
      padding: 10px 10px 10px 10px;
      background-color: cornflowerblue;
      border-radius: 10px;
      margin-top: 270px;
      margin-left: 600px;
    }

    h1{
      margin-left: 92px;
    }
  </style>
</head>

<body>
  <div class="container">
    <h1>Digital Clock</h1>
    <div class="clock">
      <span id="hour"></span>
      <span>:</span>
      <span id="minute"></span>
      <span>:</span>
      <span id="second"></span>
      <span>&nbsp;</span>
      <span id="session"></span>
    </div>
  </div>
</body>
```

```

<script>
  function showTime() {
    const date = new Date();
    let hour = date.getHours();
    let minute = date.getMinutes();
    let second = date.getSeconds();
    let session = "AM";

    if (hour === 0) {
      hour = 12;
    }

    if (hour > 12) {
      hour = hour - 12;
      session = "PM";
    }

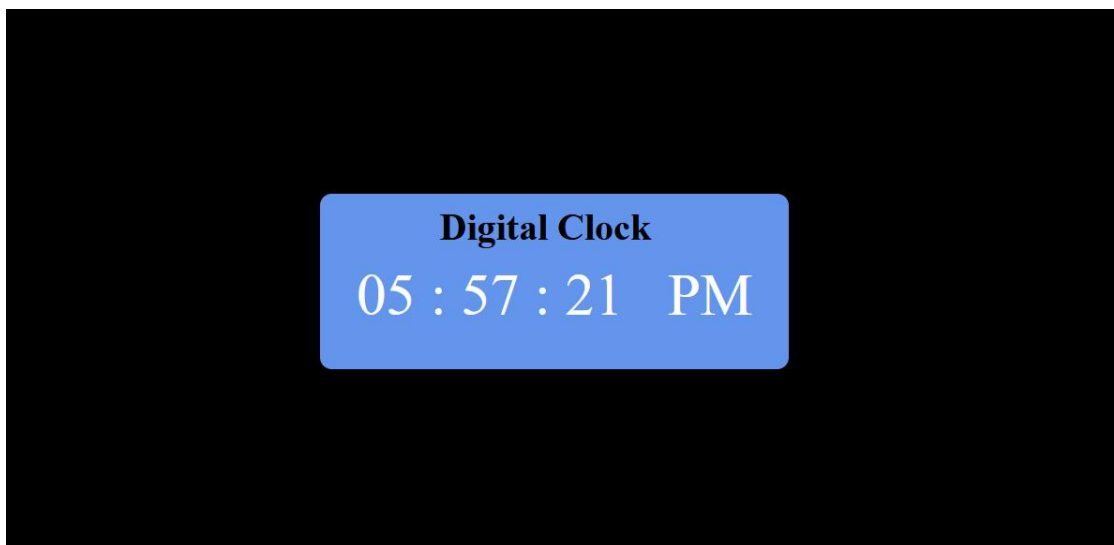
    hour = (hour < 10) ? "0" + hour : hour;
    minute = (minute < 10) ? "0" + minute : minute;
    second = (second < 10) ? "0" + second : second;

    document.getElementById("hour").textContent = hour;
    document.getElementById("minute").textContent = minute;
    document.getElementById("second").textContent = second;
    document.getElementById("session").textContent = session;
  }
  var myVar = setInterval(function () {
    showTime();
  }, 1);
</script>
</body>

</html>

```

Output



5. Develop a web application to implement online quiz system using HTML, CSS and JavaScript

Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="stylesheet" href="style.css">
  <title>Quiz</title>
</head>
<body>
  <div class="quiz-container" id="quiz">
    <div class="quiz-header">
      <h2 class="" id="question">Question Text</h2>
      <ul>
        <li>
          <input type="radio" name="answer" id="a" class="answer">
          <label for="a" id="a_text">Answer</label>
        </li>
        <li>
          <input type="radio" name="answer" id="b" class="answer">
          <label for="b" id="b_text">Answer</label>
        </li>
        <li>
          <input type="radio" name="answer" id="c" class="answer">
          <label for="c" id="c_text">Answer</label>
        </li>
        <li>
          <input type="radio" name="answer" id="d" class="answer">
          <label for="d" id="d_text">Answer</label>
        </li>
      </ul>
    </div>
    <button id="submit">Submit</button>
  </div>
  <script src="script.js"></script>
</body>
</html>
```

Style.css

```
@import
url('https://fonts.googleapis.com/css2?family=Montserrat:wght@500&family=Poppins:wght@200;300;400;500&display=swap');

*{
  box-sizing: border-box;
}

body{
  background-color: #6c6c6c;
  background-image: linear-gradient(315 deg,#b8c6db 0%, #f5f7f7 100%);
  font-family: 'Poppins','sans-serif';
```

```

display: flex;
align-items: center;
justify-content: center;
height: 100vh;
overflow: hidden;
margin: 0;
}

.quiz-container{
background-color: #fff;
border-radius: 10px;
box-shadow: 0 0 10px 2px rgba(100, 100, 100, 0.1);
width: 600px;
overflow: hidden;
}

.quiz-header{
padding: 4rem;
}

h2{
padding: 1rem;
text-align: center;
margin: 0;
}

ul{
list-style-type: none;
padding: 0;
}

ul li{
font-size: 1.2rem;
margin: 0;
}

ul li label{
cursor: pointer;
}

button{
background-color: #03cae4;
color: #fff;
border: none;
display: block;
width: 100%;
cursor: pointer;
font-size: 1.1rem;
font-family: inherit;
padding: 1.3rem;
}

button:hover{
background-color: #04adc4;
}

button:focus{
outline: none;
background-color: #44b929;
}

```

Script.js

```
const quizData = [
  {
    question: "which language gives structure to a web page?",
    a: "Java",
    b: "C",
    c: "Html",
    d: "JavaScript",
    correct: "c"
  },
  {
    question: "Full form for HTML?",
    a: "Hyper text marked language",
    b: "Hyper term magnificent language",
    c: "High throughput magical language",
    d: "Hyper text markup language",
    correct: "d"
  },
  {
    question: "which is the fastest language in terms of execution speed?",
    a: "C",
    b: "Java",
    c: "Python",
    d: "JavaScript",
    correct: "a"
  },
  {
    question: "which language gives functionality to a web page?",
    a: "Java",
    b: "C",
    c: "Html",
    d: "JavaScript",
    correct: "d"
  },
  {
    question: "who developed Java?",
    a: "Sun Microsystems",
    b: "Microsoft",
    c: "Google",
    d: "Oracle",
    correct: "a"
  }
]
```

```
const quiz = document.getElementById('quiz')
const answerEls = document.querySelectorAll('.answer')
const questionEl = document.getElementById('question')
const a_text = document.getElementById('a_text')
const b_text = document.getElementById('b_text')
const c_text = document.getElementById('c_text')
const d_text = document.getElementById('d_text')
const submitBtn = document.getElementById('submit')
```

```
let currentQuiz = 0
let score = 0
```

```
loadQuiz()
```

```

function loadQuiz(){
  deselectAnswers()
  const currentQuizData = quizData[currentQuiz]
  questionEl.innerText = currentQuizData.question
  a_text.innerText = currentQuizData.a
  b_text.innerText = currentQuizData.b
  c_text.innerText = currentQuizData.c
  d_text.innerText = currentQuizData.d
}

function deselectAnswers(){
  answerEls.forEach(answerEl => answerEl.checked = false)
}

function getSelected(){
  let answer
  answerEls.forEach(answerEl=>{
    if(answerEl.checked){
      answer = answerEl.id
    }
  })
  return answer
}

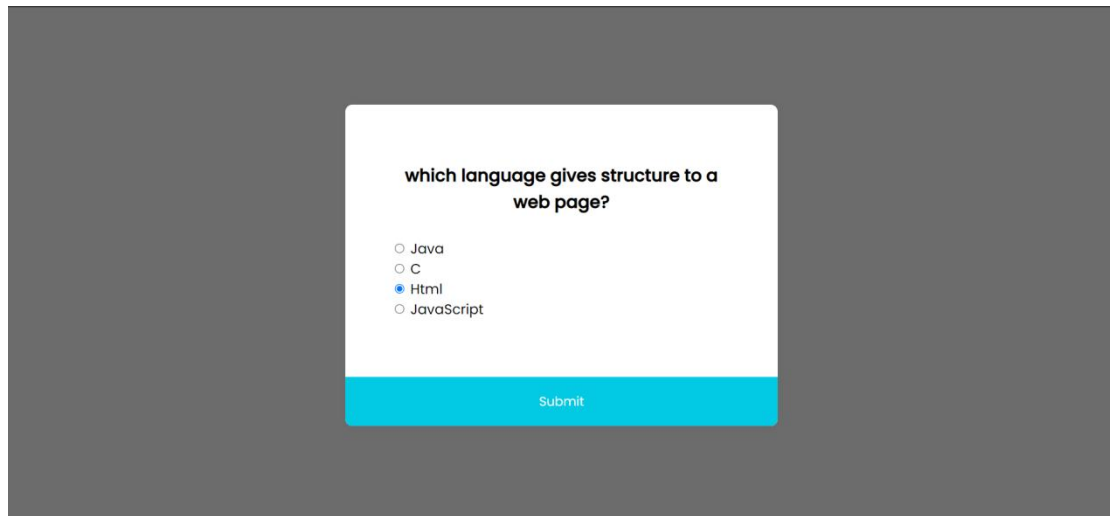
submitBtn.addEventListener('click',()=>{

  const answer = getSelected()
  if(answer){
    if(answer===quizData[currentQuiz].correct){
      score++
    }
    currentQuiz++

    if(currentQuiz<quizData.length){
      loadQuiz()
    }
    else{
      quiz.innerHTML =
        `<h2>You answered ${score}/${quizData.length} questions correctly</h2>
        <button onclick ="location.reload()">Reload</button>`
    }
  }
})

```

Output



A screenshot of a quiz question interface. The background is a solid dark gray. In the center, there is a white rectangular box with rounded corners. Inside this box, the text "which language gives structure to a web page?" is displayed in a bold, black, sans-serif font. Below the question, there are four radio button options: "Java", "C", "Html", and "JavaScript". The "Html" option is selected, indicated by a small blue dot next to the radio button. At the bottom of the white box, there is a solid cyan-colored button with the word "Submit" written in a small, white, sans-serif font.

which language gives structure to a web page?

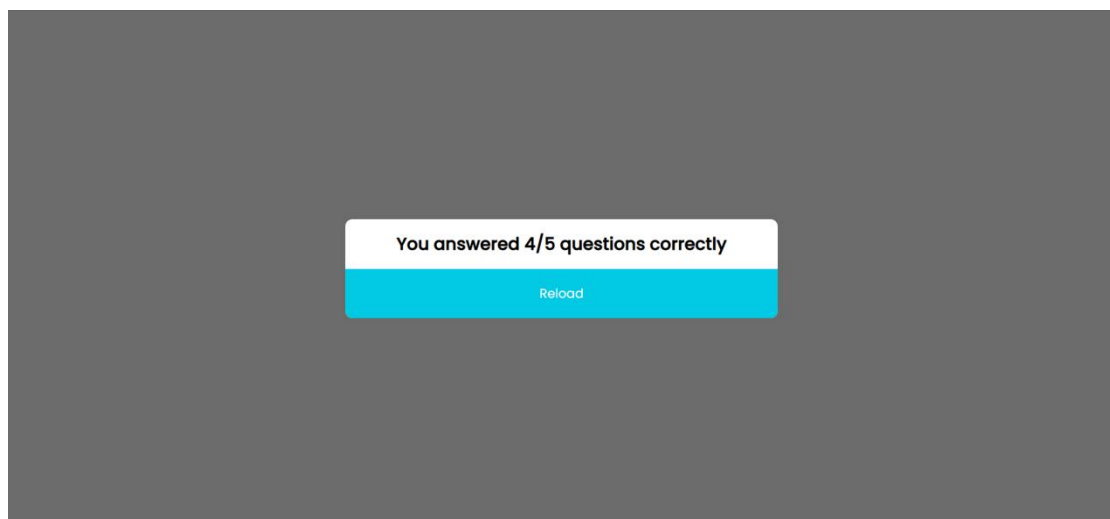
☐ Java

☐ C

☒ Html

☐ JavaScript

Submit



A screenshot of a quiz result interface. The background is a solid dark gray. In the center, there is a white rectangular box with rounded corners. Inside this box, the text "You answered 4/5 questions correctly" is displayed in a bold, black, sans-serif font. Below the text, there is a solid cyan-colored button with the word "Reload" written in a small, white, sans-serif font.

You answered 4/5 questions correctly

Reload

6. Create a <TodoItem> component in React and reuse it inside a <TodoList> component.

App.js

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

function TodoItem({ task, onDelete }) {
  return (
    <div className="todo-item">
      <span>{task}</span>
      <button onClick={onDelete} className="delete-button">Delete</button>
    </div>
  );
}

function TodoList() {
  const [tasks, setTasks] = useState([]);
  const [newTask, setNewTask] = useState("");

  const handleTaskChange = (event) => {
    setNewTask(event.target.value);
  };

  const handleAddTask = () => {
    if (newTask.trim() !== "") {
      setTasks([...tasks, newTask]);
      setNewTask("");
    }
  };

  const handleDeleteTask = (index) => {
    const updatedTasks = tasks.filter((_, i) => i !== index);
    setTasks(updatedTasks);
  };

  return (
    <div className="container">
      <h1>Todo List</h1>
      <div className="input-container">
        <input
          type="text"
          value={newTask}
          onChange={handleTaskChange}
          placeholder="Enter a task"
        />
        <button onClick={handleAddTask}>Add Task</button>
      </div>

      <ul className="task-list">
        {tasks.map((task, index) => (
          <li key={index}>
            <TodoItem task={task} onDelete={() => handleDeleteTask(index)} />
          </li>
        ))}
      </ul>

      <style jsx>{
        .container {
```

```

    display: flex;
    flex-direction: column;
    align-items: center;
    justify-content: center;
    min-height: 100vh;
    background-color: #f2f2f2;
  }

  h1 {
    margin-bottom: 1rem;
  }

  .input-container {
    display: flex;
    align-items: center;
    margin-bottom: 1rem;
  }

  input {
    padding: 0.5rem;
    margin-right: 0.5rem;
    border: none;
    border-radius: 4px;
  }

  button {
    padding: 0.5rem 1rem;
    background-color: #4caf50;
    color: #fff;
    border: none;
    border-radius: 4px;
    cursor: pointer;
  }

  ul {
    list-style: none;
    padding: 0;
  }

  li {
    margin-bottom: 0.5rem;
  }

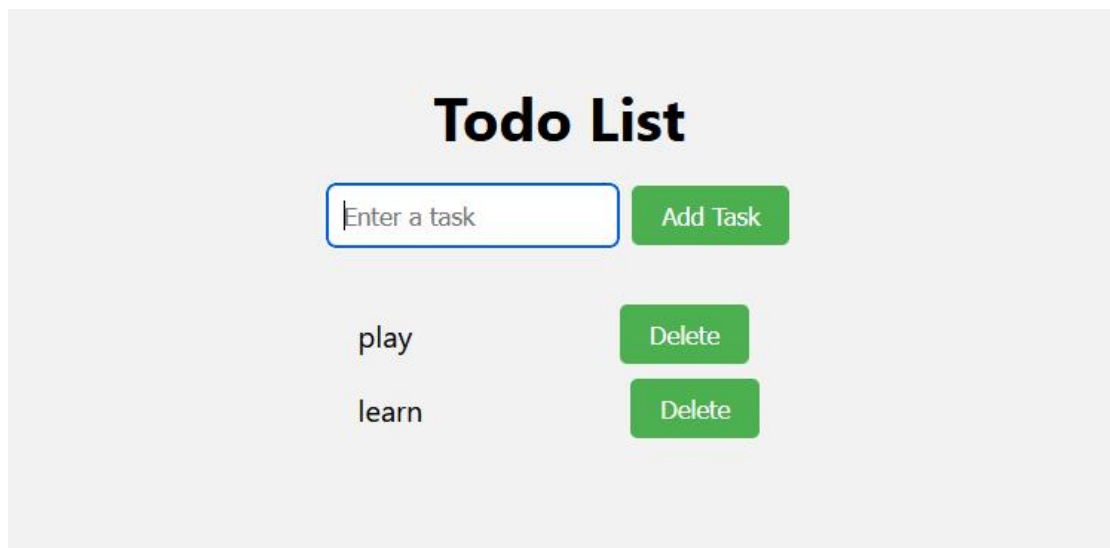
  .todo-item {
    display: flex;
    align-items: center;
  }

  .delete-button {
    margin-left: 7rem;
  }
`}</style>
</div>
);
}

export default TodoList;

```

Output



Todo List

Enter a task Add Task

play Delete

learn Delete

7. Design a shopping cart application using React. Your shopping webpage should have the provisions for selecting the list of items from different category, Once the items are selected on clicking the submit button the items in the cart with its price should be displayed.

app.js

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

function ShoppingCart() {
  const [selectedItems, setSelectedItems] = useState([]);
  const [cartItems, setCartItems] = useState([]);

  // Available items
  const items = [
    { id: 1, name: 'Laptop', price: 40 },
    { id: 2, name: 'Phone', price: 20 },
    { id: 3, name: 'TV', price: 30 },
    { id: 4, name: 'Speaker', price: 10 },
  ];

  // Function to handle item selection
  const handleSelectItem = (itemId) => {
    const selectedItem = items.find((item) => item.id === itemId);
    setSelectedItems([...selectedItems, selectedItem]);
  };

  // Function to handle adding selected items to the cart
  const handleAddToCart = () => {
    setCartItems([...cartItems, ...selectedItems]);
    setSelectedItems([]);
  };

  // Calculate total cost of selected items
  const totalCost = selectedItems.reduce((total, item) => total + item.price, 0);

  return (
    <div className="container">
      <h1>Shopping Cart</h1>
      <div className="item-list">
        <h2>Available Items</h2>
        <ul>
          {items.map((item) => (
            <li key={item.id}>
              {item.name} - ${item.price}
              <button onClick={() => handleSelectItem(item.id)}>Select</button>
            </li>
          ))}
        </ul>
      </div>
      <div className="selected-items">
        <h2>Selected Items</h2>
        {selectedItems.length === 0 ? (
          <p>No items selected</p>
        ) : (
          <ul>
            {selectedItems.map((item) => (
              <li key={item.id}>
                {item.name} - ${item.price}
              </li>
            ))}
          </ul>
        )}
      </div>
    </div>
  );
}
```

```

        </li>
      )})
    </ul>
  )}
</div>
<div className="total-cost">
  <h2>Total Cost</h2>
  <p>${totalCost}</p>
</div>
<div className="cart">
  <button onClick={handleAddToCart}>Add to Cart</button>
</div>
<style jsx>{`
  .container {
    display: flex;
    flex-direction: column;
    align-items: center;
    justify-content: center;
    min-height: 100vh;
    background-color: #f2f2f2;
  }

  h1 {
    margin-bottom: 1rem;
  }

  .item-list {
    margin-bottom: 1rem;
  }

  ul {
    list-style: none;
    padding: 0;
  }

  li {
    display: flex;
    align-items: center;
    justify-content: space-between;
    margin-bottom: 0.25rem;
  }

  .selected-items {
    margin-bottom: 1rem;
  }

  .total-cost {
    margin-bottom: 1rem;
  }

  .cart {
    display: flex;
    align-items: center;
  }

  button {
    padding: 0.5rem 1rem;
    background-color: #4caf50;
    color: #fff;
    border: none;
  }
`}
```

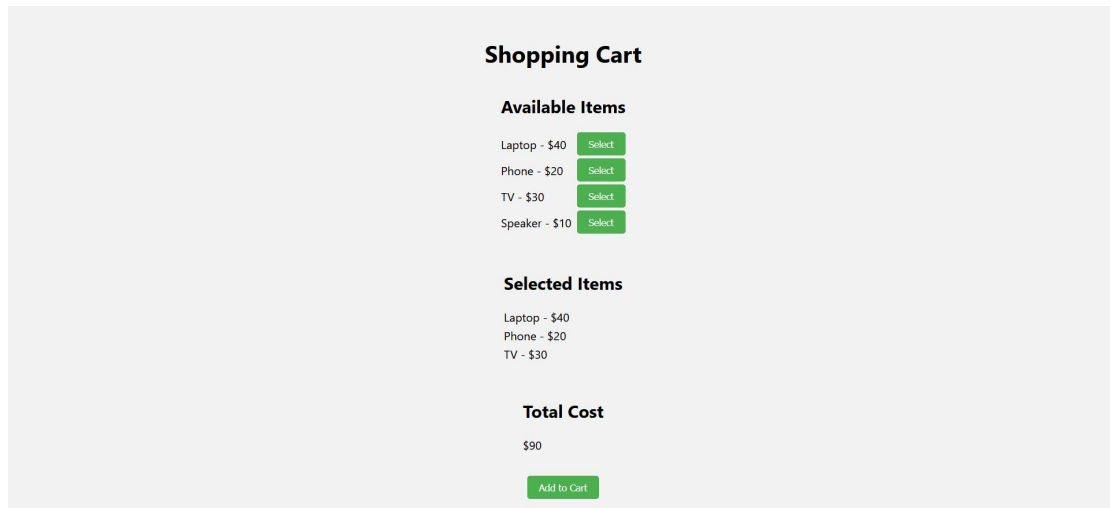
```

        border-radius: 4px;
        cursor: pointer;
      }
    }</style>
  </div>
);
}

export default ShoppingCart;

```

Output



8. Develop a Command Line Application for an online super market using NodeJS & MySQL to perform: a) search based on product id or name b) On retrieving the results, display the product details of different brands in table format with the Price field in sorted order.

app.js

```
const mysql = require('mysql2');
const readline = require('readline');

// Create a connection to the MySQL database
const connection = mysql.createConnection({
  host: 'localhost',
  user: 'root',
  password: '',
  database: 'supermarket'
});

// Connect to the database
connection.connect((error) => {
  if (error) {
    console.error('Failed to connect to the database:', error);
    process.exit(1);
  }
});

// Create readline interface
const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout
});

// Function to search for products based on ID or name
function searchProducts() {
  rl.question('Enter product ID or name to search: ', (searchInput) => {
    rl.pause();
    // Query the database for products matching the search input
    const query = `
      SELECT *
      FROM products
      WHERE id = ? OR name LIKE ?
    `;
    connection.query(query, [searchInput, `%${searchInput}%`], (error, results) => {
      if (error) {
        console.error('Error executing the query:', error);
        rl.close();
        return;
      }

      if (results.length === 0) {
        console.log('No products found matching the search input.');
```

```
      } else {
        // Sort the results by price in ascending order
        results.sort((a, b) => a.price - b.price);

        // Display the results in table format
        console.log('\nProduct Details:');
        console.log('-----');
        console.log('ID\tName\tBrand\tPrice');
```

```

        console.log('-----');
        results.forEach((product) => {
            console.log(` ${product.id}\t${product.name}\t\t${product.brand}\t\t${product.price}`);
        });
    }

    promptAction();
});
});
}

// Function to display all products
function displayProducts() {
    // Query the database to retrieve all products
    const query = `
        SELECT id, name, brand, price
        FROM products
    `;
    connection.query(query, (error, results) => {
        if (error) {
            console.error('Error executing the query:', error);
            rl.close();
            return;
        }

        if (results.length === 0) {
            console.log('No products found. ');
        } else {
            // Display the results in table format
            console.log("\nProduct List:");
            console.log('-----');
            console.log('ID\tName\t\tBrand\t\tPrice');
            console.log('-----');
            results.forEach((product) => {
                console.log(` ${product.id}\t${product.name}\t\t${product.brand}\t\t${product.price}`);
            });
        }

        promptAction();
    });
}

// Function to add a new product
function addProduct() {
    rl.question('Enter product name: ', (name) => {
        rl.question('Enter product brand: ', (brand) => {
            rl.question('Enter product price: ', (price) => {
                rl.pause(); // Pause the readline interface
                // Insert the new product into the database
                const query = `
                    INSERT INTO products (name, brand, price)
                    VALUES (?, ?, ?)
                `;
                connection.query(query, [name, brand, price], (error) => {
                    if (error) {
                        console.error('Error adding the product:', error);
                    } else {
                        console.log('Product added successfully. ');
                    }
                });
            });
        });
    });
}

```

```

        promptAction();
    });
    });
    });
}

// Function to delete a product
function deleteProduct() {
    rl.question('Enter the ID of the product to delete: ', (id) => {
        rl.pause(); // Pause the readline interface
        // Delete the product from the database
        const query = `
            DELETE FROM products
            WHERE id = ?
        `;
        connection.query(query, [id], (error) => {
            if (error) {
                console.error('Error deleting the product:', error);
            } else {
                console.log('Product deleted successfully.');
```

```
});  
}
```

```
// Start the program  
promptAction();
```

Output

```
Choose an action:  
1. Search for products  
2. Display all products  
3. Add a product  
4. Delete a product  
5. Exit  
Enter your choice: 1  
Enter product ID or name to search: 7  
7  
  
Product Details:  
-----  
ID      Name      Brand      Price  
-----  
7       Laptop     Lenovo     50000.00
```

```
Enter your choice: 2  
2  
  
Product List:  
-----  
ID      Name      Brand      Price  
-----  
6       Watch     Titan      3000.00  
7       Laptop     Lenovo     50000.00  
8       Mobile    Apple      100000.00
```

```
Enter your choice: 3  
3  
Enter product name: Speaker  
Speaker  
Enter product brand: JBL  
JBL  
Enter product price: 2000  
2000  
Product added successfully.
```

```
Enter your choice: 4  
4  
Enter the ID of the product to delete: 8  
8  
Product deleted successfully.
```

9. Create a basic CRUD operation API by following REST syntax for a given model student with the following fields [field names] using MySQL & Express JS.

Steps:

1. Initialize node inside the project folder by typing npm init.
2. Then install required node packages like mysql, express, dotenv, body-parser.
3. Command for installing -> npm i package_name.
4. Install xampp server and create a database.
5. Create a file called ".env" and store the db credentials inside it.

display.ejs

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <style>
    .container{
      padding: 20px;
      margin-top: 60px;
      margin-left: 515px;
    }
    table{
      width: 600px;
      text-align: center;
      margin-left: -85px;
    }
    table th,td{
      height: 40px;
      width: 40px;
      background-color: white;
    }
    form{
      margin-left: -72px;
    }
    input{
      height: 40px;
      width: 160px;
      margin-left: 20px;
    }
    h1{
      margin-left: 145px;
    }
    body{
      background-color: rgb(151, 150, 150);
    }
  </style>
  <title>Display</title>
</head>
<body>
  <div class="container">
    <h1>CRUD App</h1>
    <form action="/" method="post">
```



```

        <input type="text" placeholder="Name" name="name">
        <input type="text" placeholder="Email" name="email">
        <input type="submit" value="Add" >
    </form><br><br>
<table border="2">
    <tr>
        <th>ID</th>
        <th>Name</th>
        <th>Email</th>
        <th>Update</th>
        <th>Delete</th>
    </tr>
    <% if (test.length > 0) { %>
        <% for (var i = 0; i < test.length; i++) { %>
            <tr>
                <td><%= test[i].id %></td>
                <td><%= test[i].UserName %></td>
                <td><%= test[i].Email %></td>
                <td><a href="/update?id=<%= test[i].id %>">Update</a></td>
                <td><a href="/delete?id=<%= test[i].id %>">Delete</a></td>
            </tr>
            <% } %>
        <% } else { %>
            <tr>
                <td colspan="5"><h3>Data not found</h3></td>
            </tr>
            <% } %>
        </table>
    </div>
</body>
</html>

```

update.ejs

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Update</title>
    <style>
        .container{
            padding: 20px;
            margin-top: 60px;
            margin-left: 400px;
        }
        form{
            margin-left: -72px;
        }
        input{
            height: 40px;
            width: 160px;
            margin-left: 20px;
        }
        h1{

```

```

    margin-left: 205px;
  }
  body{
    background-color: rgb(151, 150, 150);
  }
</style>
</head>
<body>
  <div class="container">
    <h1>Update Record</h1>
    <form action="/updateData" method="post">
      <input type="text" placeholder="Name" name="id" value="<%= test[0].id%>" readonly>
      <input type="text" placeholder="Name" name="name" value="<%= test[0].UserName%>"
autocomplete="off">
      <input type="text" placeholder="Email" name="email" value="<%= test[0].Email%>"
autocomplete="off">
      <input type="submit" value="Update" >
    </form>
  </div>
</body>
</html>

```

.env

```

HOST = localhost
USER = your_user_name
PASSWORD =
DATABASE = your_db_name

```

connection.js

```

const mysql = require('mysql');

require('dotenv').config();

var con = mysql.createConnection({
  host: process.env.HOST,
  user: process.env.USER,
  password: process.env.PASSWORD,
  database: process.env.DATABASE
});

con.connect(function(err){
  if(err) throw err;
  console.log("DB Connected");
});

module.exports = con;

```

app.js

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

//creating a app variable to carry out express framework's functions
const app = express()

//dotenv variable for accessing .env file's variables
require('dotenv').config();

//establish connection
var con = require("./connection");

//to get data from front end
const bodyParser = require("body-parser");
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({extended:true}));

//for using js code inside html elements
app.set('view engine','ejs');

//listening app on port 3000
app.listen(3000,function(){
  console.log("Server is running");
})

//add data[CREATE]
app.post("/",function(req,res){
  var name = req.body.name;
  var email = req.body.email;
  var sql = 'INSERT INTO test (UserName, Email) VALUES (?)';
  var values = [name,email];
  con.query(sql,[values],function(err,result){
    if(err) throw err;
    res.redirect('/');
  })
})

//display data[READ]
app.get("/", function(req, res) {
  var sql = 'SELECT * FROM test;';
  con.query(sql, function(error, result) {
    if (error) throw error;
    res.render("display", { test: result });
  });
});

//display update form
app.get("/update",function(req,res){
  con.connect(function(error){
    if(error) console.log(error);
    var sql = "select * from test where id =?;";
    var id = req.query.id;
    con.query(sql,[id],function(error,result){
```

```

        if(error) console.log(error);
        res.render('update',{test : result});
    })
}
})

//update data[UPDATE]
app.post("/updateData",function(req,res){
    var name = req.body.name;
    var email = req.body.email;
    var id = req.body.id;
    var sql = "update test set UserName=?, Email=? where id=?;";
    con.query(sql,[name,email,id],function(error,result){
        if(error) console.log(error);
        res.redirect("/");
    })
})

//delete data[DELETE]
app.get("/delete",function(req,res){
    con.connect(function(err){
        if(err) console.log(err);
        var sql = "delete from test where id=?;";
        var id = req.query.id;
        con.query(sql,[id],function(error,result){
            if(error) console.log(error);
            res.redirect("/");
        })
    })
})
})

```

Output

CRUD App

ID	Name	Email	Update	Delete
15	abc	abc@gmail.com	Update	Delete

Update Record

10. To build an AJAX Application

index.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>Weather App</title>
    <link rel="stylesheet" type="text/css" href="style.css">
  </head>
  <body>
    <div class="container">
      <h1>Weather App</h1>
      <form id="searchForm">
        <input type="text" name="city" placeholder="Enter city name">
        <button type="submit">Search</button>
      </form>
      <div id="weatherInfo"></div>
    </div>
    <script src="script.js"></script>
  </body>
</html>
```

style.css

```
.container {
  max-width: 500px;
  margin: 0 auto;
  text-align: center;
}

h1 {
  font-size: 3em;
}

form input[type=text] {
  padding: 10px;
  font-size: 1.2em;
  border: none;
  border-bottom: 2px solid #ccc;
}

form button {
  padding: 10px;
  font-size: 1.2em;
  background-color: #4CAF50;
  color: white;
  border: none;
  cursor: pointer;
  transition: background-color 0.3s;
}

form button:hover {
  background-color: #3e8e41;
}
```

```
#weatherInfo {
  margin-top: 20px;
  font-size: 1.2em;
}
```

script.js

```
document.getElementById('searchForm').addEventListener('submit', function(event) {
  event.preventDefault();

  var cityInput = document.querySelector('input[name="city"]');
  var cityName = cityInput.value;

  if (cityName !== '') {
    var xhr = new XMLHttpRequest();
    xhr.open('GET', 'https://api.openweathermap.org/data/2.5/weather?q=' +
    encodeURIComponent(cityName) + '&appid=6d548e9d58b6f97ecf4c25872b547423', true);
    xhr.onreadystatechange = function() {
      if (xhr.readyState === 4) {
        if (xhr.status === 200) {
          var weatherData = JSON.parse(xhr.responseText);
          var weatherInfo = document.getElementById('weatherInfo');
          weatherInfo.innerHTML = `
            <h2>${weatherData.name}</h2>
            <p>Temperature: ${convertKelvinToCelsius(weatherData.main.temp)}°C</p>
            <p>Humidity: ${weatherData.main.humidity}%</p>
            <p>Description: ${weatherData.weather[0].description}</p>
          `;
        } else {
          var weatherInfo = document.getElementById('weatherInfo');
          weatherInfo.innerHTML = 'Unable to fetch weather information. Please try again later.';
        }
      }
    };
    xhr.send();
  }
});

function convertKelvinToCelsius(kelvin) {
  return Math.round(kelvin - 273.15);
}
```

Output

Weather App

Chennai

Search

Chennai

Temperature: 31°C

Humidity: 90%

Description: few clouds