KESHAV MEMORIAL INSTITUTE OF TECHNOLOGY

(AN AUTONOMOUS INSTITUTION)

Accredited by NBA & NAAC, Approved by AICTE, Affiliated to JNTUH,

Narayanguda, Hyderabad – 500029

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

LAB RECORD

WEB TECHNOLOGIES LAB

B.Tech. III YEAR I SEM (RKR21)

ACADEMIC YEAR 2024-25

KESHAV MEMORIAL INSTITUTE OF TECHNOLOGY

(AN AUTONOMOUS INSTITUTE)

Accredited by NBA & NAAC, Approved by AICTE, Affiliated to JNTUH, Hyderabad

Certificate

This is to certify that following is a Bonafide Record of the workbook task done by _____bearing Roll No_____ of _____ Branch of _____ year B.Tech Course in the_____ Subject during the Academic year _____ & _____ under our supervision.

Number of week tasks completed:

_____

Signature of Staff Member Incharge

Signature of Head of the Dept.

Signature of Internal Examiner

Signature of External Examiner

KESHAV MEMORIAL INSTITUTE OF TECHNOLOGY

(AN AUTONOMOUS INSTITUTE)

Accredited by NBA & NAAC, Approved by AICTE, Affiliated to JNTUH, Hyderabad

Daily Laboratory Assessment Sheet

Name of the Lab: Name of the Student:

Class: HT. No:

S.No.	Name of the Experiment	Page.No	Date	Observation Marks (3M)	Record Marks (4M)	Viva Voice Marks (3M)	Total Marks (10M)	Signature of Faculty

S.No.	Name of the Experiment	Page.No	Date	Observation Marks (3M)	Record Marks (4M)	Viva Voice Marks (3M)	Total Marks (10M)	Signature of Faculty
	TOTAL							

1.

Design the food blog webpage using only HTML elements

The webpage is divided into several sections

1. Hyper Link 1: "Welcome to My Food Blog" - This is the main title of the blog.
2. Hyper Link 2: "About Me" - This section introduces the blog author. Place the content about the author.
3. List 1: "Latest Posts" - This section lists the most recent blog posts.
4. List 2: "Popular Recipes" - This section lists popular recipes with links.
5. Hyper Link 3: "Contact Me" - This section provides a way for visitors to contact the author.
6. Hyper Link 4: "Subscribe to My Blog" - This section includes a HTML form for visitors to subscribe to the blog.

Requirements to implement

1: Update Content

Update the "Latest Posts" section by adding a new blog post titled "Healthy Salads for Every Day". Provide a brief description and a link to "salad-recipes.html".`

2: Add a New Section

Add a new section titled "Upcoming Events" after the "Subscribe to My Blog" section. List at least two events with their descriptions.

3: Create a Link

Create a hyperlink in the "About Me" section that links to the homepage of the blog. Use the text "Back to Home".

4: Update Form Elements

Modify the subscription form so that the email field is required and includes basic validation for email format.

Program :

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>My Food Blog</title>
</head>
<body>
  <!-- Navbar Section -->
    <h1>Welcome to My Food Blog</h1>
  <nav>
    <ul>
      <li><a href="#">Home</a></li>
      <li><a href="#about-me">About Me</a></li>
      <li><a href="#latest-posts">Latest Posts</a></li>
      <li><a href="#popular-recipes">Popular Recipes</a></li>
      <li><a href="#contact-me">Contact Me</a></li>
```

```
      <li><a href="#subscribe">Subscribe</a></li>
    </ul>
</nav>
<!-- About Me Section -->
<section id="about-me">
    <h2>About Me</h2>
    <p>Hello! I'm the author of this blog. I love cooking and sharing my favorite recipes with the world.</p>
    <p><a href="#">Back to Home</a></p>
</section>
<!-- Latest Posts Section -->
<section id="latest-posts">
    <h2>Latest Posts</h2>
    <ul>
        <li>
            <a href="salad-recipes.html">Healthy Salads for Every Day</a> - Discover quick and easy salad recipes for a healthy lifestyle.
        </li>
        <li>
            <a href="soup-recipes.html">Comforting Soups for Cold Days</a> - Warm up with these hearty soup recipes.
        </li>
        <li>
            <a href="dessert-recipes.html">Sweet Treats to Savor</a> - Indulge in these delightful dessert recipes.
        </li>
    </ul>
    <p><a href="#">Back to Home</a></p>
</section>
<!-- Popular Recipes Section -->
<section id="popular-recipes">
    <h2>Popular Recipes</h2>
    <ul>
        <li><a href="#">Spaghetti Carbonara</a></li>
        <li><a href="#">Chicken Rolls</a></li>
        <li><a href="#">Vegetarian Tacos</a></li>
    </ul>
    <div>
        <div class="recipe-card">
            <img src="recipe1.jpg" alt="Spaghetti Carbonara">
            <h4>RECIPE 1: Spaghetti Carbonara</h4>
```

```
            <p><strong>Description:</strong> A classic and custom recipe. Enjoy a creamy and delicious plate of carbonara that satisfies every craving.</p>
            <a href="#">Link to Recipe</a>
        </div>
        <div class="recipe-card">
            <img src="recipe2.jpg" alt="Chicken Rolls">
            <h4>RECIPE 2: Chicken Rolls</h4>
            <p><strong>Description:</strong> A tasty and satisfying chicken recipe that’s perfect for every occasion.</p>
            <a href="#">Link to Recipe</a>
        </div>
        <div class="recipe-card">
            <img src="recipe3.jpg" alt="Vegetarian Tacos">
            <h4>RECIPE 3: Vegetarian Tacos</h4>
            <p><strong>Description:</strong> A vibrant and healthy taco option for vegetarians and taco lovers alike.</p>
            <a href="#">Link to Recipe</a>
        </div>
    </div>
    <p><a href="#">Back to Home</a></p>
  </section>
  <!-- Contact Me Section -->
  <section id="contact-me">
    <h2>Contact Me</h2>
    <p>If you have any questions or suggestions, feel free to reach out!</p>
    <p><a href="#">Back to Home</a></p>
  </section>
  <!-- Subscribe to My Blog Section -->
   <section id="contact-subscribe">
    <h2 >Subscribe to my Blog!</h2>
    <form novalidate>
        <label for="email">Email Address:</label>
        <input type="email" id="email" required >
        <br>
        <br>
        <label for="name">Name:</label>
        <input type="text" id="name" required >
        <br>
        <br>
        <label for="password">Password:</label>
```

```
            <input type="password" id="password" required >
            <br>
            <br>
            <button type="submit" >Subscribe</button>
        </form>
        <p><a href="#">Back to Home</a></p>
    </section>
    <!-- Upcoming Events Section (After Subscribe) -->
    <section id="upcoming-events">
        <h2>Upcoming Events</h2>
        <ul>
            <li><strong>Cooking Workshop:</strong> Learn to bake desserts on January 15th.</li>
            <li><strong>Live Q&A Session:</strong> Join me on February 10th to ask your cooking-related questions.</li>
        </ul>
        <p><a href="#">Back to Home</a></p>
    </section
</body>
</html>
```

2.

Upgrade the food blog website using HTML5. The website should contain multiple sections, including an introduction about the blog author, a list of the latest blog posts, a collection of popular recipes, and a contact form for users to subscribe to the blog. The website should be user-friendly and accessible, with clear navigation and properly structured content.

Requirements to implement:

1. **Header Section:**
    1. Create a header with a title for the blog and a navigation menu that links to different sections of the page.
    2. The navigation menu should include links to the "About Me," "Latest Posts," "Popular Recipes," "Contact Me," and "Subscribe" sections.
2. **About Me Section:**
    1. Include a section that provides a brief introduction to the blog author, explaining their passion for food and cooking.
3. **Latest Posts Section:**
    1. Display the latest blog posts with a title, short description, and a "Read more" link that leads to the full post.
    2. Include at least two blog posts in this section.
4. **Popular Recipes Section:**
    1. Create a list of popular recipes with links to individual recipe pages. Each recipe should be displayed as a list item.
5. **Contact and Subscription Section:**
    1. Provide a section where users can contact the blog author via email. Include a mail to link for easy access.
    2. Include a subscription form where users can enter their name and email address to subscribe to the blog. The form should validate that both fields are filled out before submission.
6. **Footer Section:**
    1. Add a footer that contains a copyright notice with the current year.

Additional Information:

1. Use semantic HTML5 elements such as <header>, <section>, <article>, and <footer> to improve the structure and accessibility of the content.

Progarm

```
<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

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

  <title>My Food Blog</title>

</head>

<body>

  <!-- Header Section -->

  <header>

    <h1>Welcome to My Food Blog</h1>
```

```
    <nav>
      <ul>
        <li><a href="#about-me">About Me</a></li>
        <li><a href="#latest-posts">Latest Posts</a></li>
        <li><a href="#popular-recipes">Popular Recipes</a></li>
        <li><a href="#contact-me">Contact Me</a></li>
        <li><a href="#subscribe">Subscribe</a></li>
      </ul>
    </nav>
  </header>
  <!-- About Me Section -->
  <section id="about-me">
    <h2>About Me</h2>
    <p>Hello! I'm the author of this blog. I love cooking and sharing my favorite recipes with the world. Join me as I explore delightful culinary adventures!</p>
  </section>
  <!-- Latest Posts Section -->
  <section id="latest-posts">
    <h2>Latest Posts</h2>
    <ul>
      <li>
        <a href="salad-recipes.html">Healthy Salads for Every Day</a> - Discover quick and easy salad recipes for a healthy lifestyle.
      </li>
      <li>
        <a href="soup-recipes.html">Comforting Soups for Cold Days</a> - Warm up with these hearty soup recipes.
      </li>
      <li>
        <a href="dessert-recipes.html">Sweet Treats to Savor</a> - Indulge in these delightful dessert recipes.
      </li>
    </ul>
  </section>
  <!-- Popular Recipes Section -->
  <section id="popular-recipes">
    <h2>Popular Recipes</h2>
    <ul>
      <li><a href="#">Spaghetti Carbonara</a></li>
      <li><a href="#">Chicken Rolls</a></li>
      <li><a href="#">Vegetarian Tacos</a></li>
```

```
    </ul>
    <div>
      <div class="recipe-card">
        <img src="recipe1.jpg" alt="Spaghetti Carbonara">
        <h4>RECIPE 1: Spaghetti Carbonara</h4>
        <p><strong>Description:</strong> A classic and custom recipe. Enjoy a creamy and delicious plate of carbonara that satisfies every craving.</p>
        <a href="#">Link to Recipe</a>
      </div>
      <div class="recipe-card">
        <img src="recipe2.jpg" alt="Chicken Rolls">
        <h4>RECIPE 2: Chicken Rolls</h4>
        <p><strong>Description:</strong> A tasty and satisfying chicken recipe that’s perfect for every occasion.</p>
        <a href="#">Link to Recipe</a>
      </div>
      <div class="recipe-card">
        <img src="recipe3.jpg" alt="Vegetarian Tacos">
        <h4>RECIPE 3: Vegetarian Tacos</h4>
        <p><strong>Description:</strong> A vibrant and healthy taco option for vegetarians and taco lovers alike.</p>
        <a href="#">Link to Recipe</a>
      </div>
    </div>
  </section>

  <!-- Contact Me Section -->
  <section id="contact-me">
    <h2>Contact Me</h2>
    <p>If you have any questions or suggestions, feel free to <a href="mailto:author@myfoodblog.com">email me</a>.</p>
  </section>
  <!-- Subscribe Section -->
  <section id="subscribe">
    <h2>Subscribe to My Blog!</h2>
    <form novalidate>
      <label for="name">Name:</label>
      <input type="text" id="name" required><br><br>
      <label for="email">Email Address:</label>
      <input type="email" id="email" required><br><br>
      <label for="password">Password:</label>
```

```
            <input type="password" id="password" required><br><br>
            <button type="submit">Subscribe</button>
        </form>
    </section>
    <!-- Footer Section -->
    <footer>
        <p>&copy; My Food Blog. All rights reserved.</p>
    </footer>
</body>
</html>
```

3.

Upgrade the food blog website designed in experiment 2 with CSS. Apply external styling approach.

Requirements to apply styles:

· Header:

The header includes the blog's title and a navigation bar. The navigation links allow users to jump to different sections of the page. Set the background colour to light blue., keep on scrolling the header content left to right.

· Main Content:

The main content is divided into sections (<section>) for "About the Author," "Latest Blog Posts," "Popular Recipes," and "Contact." Each section is styled with padding, a white background, rounded corners, apply element and id, class selectors.

· **Responsive Design**:

Apply media queries on popular recipes. On larger screens, the content is arranged in a grid layout, with two columns for better use of space. The grid adjusts as the screen width increases, maintaining readability and usability on various devices.

· Footer:

The footer is simple, centered, and includes copyright information. Beautify the content with setting the margin, border style.

Program

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

    html {
      scroll-padding-top: 6em;
      scroll-behavior: smooth;
    }
    body {
      font-family: Arial, sans-serif;
      margin: 0;
      padding: 0;
      background-color: #f9f9f9;
    }
    header {
      background: lightblue;
      color: #333;
      padding: 0.5em;
      text-align: center;
```

```
    position: fixed;
    width: 100%;
    top: 0;
    z-index: 1000;
}
header h1 {
    animation: scroll 10s linear infinite;
}
@keyframes scroll {
    0% { transform: translateX(100%); }
    100% { transform: translateX(-100%); }
}
nav ul {
    list-style: none;
    padding: 0;
    margin: 0;
    text-align: center;
}
nav ul li {
    display: inline;
    margin: 0 10px;
}
nav ul li a {
    color: #333;
    text-decoration: none;
}

#about-me {
    padding-top: 3em;
    margin-top: 10em;
}
section {
    padding: 2em 1.5em;
    margin: 6em auto 2em auto;
    background: white;
    border-radius: 5px;
    max-width: 90%;
    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
```

```
}
section h2 {
    font-size: 1.8em;
    color: #333;
    border-bottom: 2px solid lightblue;
    padding-bottom: 0.3em;
    margin-bottom: 1em;
    text-align: center;
}
section p, section ul {
    font-size: 1em;
    line-height: 1.6;
    color: #555;
    text-align: justify;
}
.recipe-container {
    display: grid;
    grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));
    gap: 15px;
}
.recipe-card {
    border: 1px solid #ddd;
    border-radius: 5px;
    background: #fff;
    text-align: center;
    padding: 10px;
    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
}
.recipe-card img {
    width: 100%;
    border-bottom: 1px solid #ddd;
    margin-bottom: 10px;
}
.recipe-card h4 {
    font-size: 1.1em;
}
.recipe-card a {
    color: white;
```

```
            background: lightblue;
            padding: 5px 10px;
            text-decoration: none;
            border-radius: 3px;
            display: inline-block;
            margin-top: 10px;
        }
        footer {
            text-align: center;
            padding: 1em;
            background: #333;
            color: white;
        }
        @media (min-width: 768px) {
            .recipe-container {
                grid-template-columns: repeat(2, 1fr);
            }
        }
    </style>
</head>
<body>
    <!-- Header Section -->
    <header>
        <h1>Welcome to My Food Blog</h1>
        <nav>
            <ul>
                <li><a href="#about-me">About Me</a></li>
                <li><a href="#latest-posts">Latest Posts</a></li>
                <li><a href="#popular-recipes">Popular Recipes</a></li>
                <li><a href="#contact-me">Contact Me</a></li>
                <li><a href="#subscribe">Subscribe</a></li>
            </ul>
        </nav>
    </header>
    <!-- About Me Section -->
    <section id="about-me">
        <h2>About Me</h2>
```

```
    <p>Hello! I’m the author of this blog. I love cooking and sharing my favorite recipes with the world. Join me as I explore delightful culinary adventures!</p>
  </section>
  <!-- Latest Posts Section -->
  <section id="latest-posts">
    <h2>Latest Posts</h2>
    <ul>
      <li>
        <a href="salad-recipes.html">Healthy Salads for Every Day</a> - Discover quick and easy salad recipes for a healthy lifestyle.
      </li>
      <li>
        <a href="soup-recipes.html">Comforting Soups for Cold Days</a> - Warm up with these hearty soup recipes.
      </li>
      <li>
        <a href="dessert-recipes.html">Sweet Treats to Savor</a> - Indulge in these delightful dessert recipes.
      </li>
    </ul>
  </section>
  <!-- Popular Recipes Section -->
  <section id="popular-recipes">
    <h2>Popular Recipes</h2>
    <div class="recipe-container">
      <div class="recipe-card">
        <img src="recipe1.jpg" alt="Spaghetti Carbonara">
        <h4>RECIPE 1: Spaghetti Carbonara</h4>
        <p>A classic and creamy dish to satisfy cravings.</p>
        <a href="#">Link to Recipe</a>
      </div>
      <div class="recipe-card">
        <img src="recipe2.jpg" alt="Chicken Rolls">
        <h4>RECIPE 2: Chicken Rolls</h4>
        <p>A tasty and satisfying recipe for all occasions.</p>
        <a href="#">Link to Recipe</a>
      </div>
      <div class="recipe-card">
        <img src="recipe3.jpg" alt="Vegetarian Tacos">
        <h4>RECIPE 3: Vegetarian Tacos</h4>
        <p>Healthy and vibrant tacos for vegetarians.</p>
```

```
            <a href="#">Link to Recipe</a>
          </div>
        </div>
      </section>
      <!-- Contact Me Section -->
      <section id="contact-me">
        <h2>Contact Me</h2>
        <p>If you have any questions or suggestions, feel free to <a href="mailto:author@myfoodblog.com">email me</a>.</p>
      </section>
      <!-- Subscribe Section -->
      <section id="subscribe">
        <h2>Subscribe to My Blog!</h2>
        <form novalidate>
          <label for="name">Name:</label>
          <input type="text" id="name" required><br><br>
          <label for="email">Email Address:</label>
          <input type="email" id="email" required><br><br>
          <label for="password">Password:</label>
          <input type="password" id="password" required><br><br>
          <button type="submit">Subscribe</button>
        </form>
      </section>
      <!-- Footer Section -->
      <footer>
        <p>&copy; <script>document.write(new Date().getFullYear());</script> My Food Blog. All rights reserved.</p>
      </footer>
    </body>
    </html>
```

4.

Upgrade food blog website structure using HTML5 and Bootstrap.

Requirements:

· Header:

Uses a Bootstrap navbar for responsive navigation. The navbar toggles into a hamburger menu on smaller screens.

· **Main Content**:

About Section: Introduces the blog author using Bootstrap's grid system.

Latest Blog Posts: A row of cards (card class) to display recent posts. Bootstrap's grid system with three columns on medium and larger screens and a stacked layout on smaller screens.

Popular Recipes: Similar to the blog posts section, it uses cards for displaying popular recipes.

Contact Section: A simple form with validation using Bootstrap's form controls (form-group, form-control, and btn classes). The validation feedback is customized with Bootstrap's was-validated class.

· Footer:

Simple and centered, providing basic copyright information.

Program

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>My Food Blog</title>
  <!-- Bootstrap CSS -->
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/css/bootstrap.min.css" rel="stylesheet">
  <style>
    html {
      scroll-padding-top: 6em;
      scroll-behavior: smooth;
    }
    body {
      font-family: Arial, sans-serif;
      margin: 0;
      padding: 0;
      background-color: #f9f9f9;
    }
    header {
      background: lightblue;
      color: #333;
      text-align: center;
```

```
    position: fixed;
    width: 100%;
    top: 0;
    z-index: 1000;
}
header h1 {
    animation: scroll 10s linear infinite;
}
nav ul {
    list-style: none;
    padding: 0;
    margin: 0;
    text-align: center;
}
nav ul li {
    display: inline;
    margin: 0 10px;
}
nav ul li a {
    color: #333;
    text-decoration: none;
}
@keyframes scroll {
    0% { transform: translateX(100%); }
    100% { transform: translateX(-100%); }
}
section {
    padding: 2em 1.5em;
    margin: 6em auto 2em auto;
    background: white;
    border-radius: 5px;
    max-width: 90%;
    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
}
footer {
    text-align: center;
    padding: 1em;
    background: #333;
```

```
        color: white;
      }
      .recipe-card img {
        width: 100%;
        border-bottom: 1px solid #ddd;
        margin-bottom: 10px;
      }
    </style>
</head>
<body>
  <!-- Header Section -->
  <header>
    <h1>Welcome to My Food Blog</h1>
    <nav>
      <ul>
        <li><a href="#about-me">About Me</a></li>
        <li><a href="#latest-posts">Latest Posts</a></li>
        <li><a href="#popular-recipes">Popular Recipes</a></li>
        <li><a href="#contact-me">Contact Me</a></li>
        <li><a href="#subscribe">Subscribe</a></li>
      </ul>
    </nav>
  </header>
  <!-- About Me Section -->
  <section id="about-me" class="container">
    <h2 class="text-center">About Me</h2>
    <p>Hello! I'm the author of this blog. I love cooking and sharing my favorite recipes with the world. Join me as I explore delightful culinary adventures!</p>
  </section>
  <!-- Latest Posts Section -->
  <section id="latest-posts" class="container">
    <h2 class="text-center">Latest Posts</h2>
    <div class="row">
      <div class="col-md-4">
        <div class="card">
          <div class="card-body">
            <h5 class="card-title">Healthy Salads for Every Day</h5>
            <p class="card-text">Discover quick and easy salad recipes for a healthy lifestyle.</p>
```

```
            <a href="salad-recipes.html" class="btn btn-primary">Read More</a>
          </div>
        </div>
      </div>
      <div class="col-md-4">
        <div class="card">
          <div class="card-body">
            <h5 class="card-title">Comforting Soups for Cold Days</h5>
            <p class="card-text">Warm up with these hearty soup recipes.</p>
            <a href="soup-recipes.html" class="btn btn-primary">Read More</a>
          </div>
        </div>
      </div>
      <div class="col-md-4">
        <div class="card">
          <div class="card-body">
            <h5 class="card-title">Sweet Treats to Savor</h5>
            <p class="card-text">Indulge in these delightful dessert recipes.</p>
            <a href="dessert-recipes.html" class="btn btn-primary">Read More</a>
          </div>
        </div>
      </div>
    </div>
  </section>
  <!-- Popular Recipes Section -->
  <section id="popular-recipes" class="container">
    <h2 class="text-center">Popular Recipes</h2>
    <div class="row">
      <div class="col-md-4">
        <div class="card">
          <img src="recipe1.jpg" class="card-img-top" alt="Spaghetti Carbonara">
          <div class="card-body">
            <h5 class="card-title">Spaghetti Carbonara</h5>
            <p class="card-text">A classic and creamy dish to satisfy cravings.</p>
            <a href="#" class="btn btn-primary">View Recipe</a>
          </div>
        </div>
      </div>
```

```
        <div class="col-md-4">
          <div class="card">
            <img src="recipe2.jpg" class="card-img-top" alt="Chicken Rolls">
            <div class="card-body">
              <h5 class="card-title">Chicken Rolls</h5>
              <p class="card-text">A tasty and satisfying recipe for all occasions.</p>
              <a href="#" class="btn btn-primary">View Recipe</a>
            </div>
          </div>
        </div>
        <div class="col-md-4">
          <div class="card">
            <img src="recipe3.jpg" class="card-img-top" alt="Vegetarian Tacos">
            <div class="card-body">
              <h5 class="card-title">Vegetarian Tacos</h5>
              <p class="card-text">Healthy and vibrant tacos for vegetarians.</p>
              <a href="#" class="btn btn-primary">View Recipe</a>
            </div>
          </div>
        </div>
      </div>
    </section>
    <!-- Contact Me Section -->
    <section id="contact-me">
      <h2>Contact Me</h2>
      <p>If you have any questions or suggestions, feel free to <a href="mailto:author@myfoodblog.com">email me</a>.</p>
    </section>
    <!-- Subscribe Section -->
      <!-- Subscribe Section -->
      <section id="subscribe" class="container">
        <h2 class="text-center">Subscribe to My Blog!</h2>
        <p class="text-center">Stay updated with the latest recipes and culinary tips. Subscribe now!</p>
        <form class="row g-3 needs-validation" novalidate>
          <div class="col-md-6 mx-auto">
            <label for="subName" class="form-label">Name:</label>
            <input type="text" id="subName" class="form-control" required>
          </div>
          <div class="col-md-6 mx-auto">
```

```
            <label for="subEmail" class="form-label">Email Address:</label>
            <input type="email" id="subEmail" class="form-control" required>
          </div>
          <div class="col-md-6 mx-auto">
            <label for="subPassword" class="form-label">Password:</label>
            <input type="password" id="subPassword" class="form-control" required>
          </div>
          <div class="col-12 text-center">
            <button type="submit" class="btn btn-primary">Subscribe</button>
          </div>
        </form>
      </section>
  <!-- Footer Section -->
  <footer>
    <p>&copy;  My Food Blog. All rights reserved.</p>
  </footer>
  <!-- Bootstrap JS -->
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/js/bootstrap.bundle.min.js"></script>
</body>
</html>
```

5a .

Write a JavaScript program which accepts a string as input and swap the case of each character. For example, if you input 'The Quick Brown Fox' the output should be 'tHEqUICKbROWNfOX'.

Progarm :

```
function swapCase(inputString) {
   let swappedString = '';
   for (let i = 0; i < inputString.length; i++) {
      let char = inputString[i];
      // Check if the character is uppercase
      if (char === char.toUpperCase()) {
         swappedString += char.toLowerCase();
      } else {
         swappedString += char.toUpperCase();
      }
   }
   return swappedString;
}
const input = "The Quick Brown Fox";
const result = swapCase(input);
console.log("Input String: ", input);
console.log("Swapped Case String: ", result);
```

5b.

Write a JavaScript program to find the most frequent item of an array.

Progarm :

```
function findMostFrequent(arr) {
   const frequencyMap = {};
   let maxFrequency = 0;
   let mostFrequentItem = null;

   // Count the frequency of each item in the array
   for (let item of arr) {
      frequencyMap[item] = (frequencyMap[item] || 0) + 1;
      if (frequencyMap[item] > maxFrequency) {
         maxFrequency = frequencyMap[item];
         mostFrequentItem = item;
      }
   }
   return {
      item: mostFrequentItem,
      frequency: maxFrequency
   };
}
const array = [1, 2, 3, 2, 4, 5, 2, 3, 3, 3];
const result = findMostFrequent(array);
console.log("Array: ", array);
console.log("Most Frequent Item: ", result.item);
console.log("Frequency: ", result.frequency);
```

5c.

Write a JavaScript program to remove duplicate items from an array

Program:

```
//option 1


function removeDuplicates(arr) {
   return arr.filter((value, index, self) => {
      return self.indexOf(value) === index;
   });
}


// Example usage:
const array = [1, 2, 2, 3, 4, 4, 5];
const uniqueArray = removeDuplicates(array);


console.log(uniqueArray); // Output: [1, 2, 3, 4, 5]


// option 2


function removeDuplicates(arr) {
   // Use a Set to store unique values, as it automatically removes duplicates
   return [...new Set(arr)];
}


// Example usage:
const array = [1, 2, 2, 3, 4, 4, 5];
const uniqueArray = removeDuplicates(array);


console.log(uniqueArray); // Output: [1, 2, 3, 4, 5]
```

6a.

Write a JavaScript program to perform a binary search.

Program :

```
function binarySearch(arr, target) {
   let start = 0;
   let end = arr.length - 1;

   while (start <= end) {
      let mid = Math.floor((start + end) / 2);

      if (arr[mid] === target) {
         return mid;
      } else if (arr[mid] < target) {
         start = mid + 1;
      } else {
         end = mid - 1;
      }
   }
   return -1;
}

let numbers = [1, 3, 5, 7, 9, 11];
console.log("Binary Search : " + binarySearch(numbers, 7));
```

6b.

Write a JavaScript program to list the properties of a JavaScript object.

Program:

```
function listProperties(obj) {
   return Object.keys(obj);
}
let sampleObject = { name: "John", age: 30, city: "New York" };
console.log("The list properties are : " + listProperties(sampleObject));
```

6c.

Write a JavaScript function to check whether an object contains given property.

Program:

```
function hasProperty(obj, property) {
   return obj.hasOwnProperty(property);
}
let person = { name: "Alice", age: 25 };
console.log(hasProperty(person, "name"));
console.log(hasProperty(person, "gender"));
```

6d.

Write a JavaScript program to sort a list of elements using Quick sort.

Program:

```
function quickSort(arr) {
   if (arr.length <= 1) return arr;
   const pivot = arr[0]; // Choose the first element as the pivot
   const left = [];
   const right = [];
   // Manually partition the array
   for (let i = 1; i < arr.length; i++) {
      if (arr[i] < pivot) {
         left.push(arr[i]);
      } else {
         right.push(arr[i]);
      }
   }
   return [...quickSort(left), pivot, ...quickSort(right)];
}
let unsortedArray = [3, 6, 8, 10, 1, 2, 1];
console.log("Quick Sort : " + quickSort(unsortedArray));
```

7a.

Implement a basic user authentication system using Map. Each user has a unique username (as key) and a password (as value). Write functions to:

1. Register a new user.
2. Delete a user.
3. Update a user's password.
4. Authenticate a user by checking if a username-password combination exists.
5. List all registered usernames using an iterator.

Program:

```
class UserAuth {
  constructor() {
    this.users = new Map();
  }
  // a) Register a new user
  register(username, password) {
    if (this.users.has(username)) {
      console.log("Username already exists.");
      return false;
    }
    this.users.set(username, password);
    console.log(`User ${username} registered successfully.`);
    return true;
  }
  // b) Delete a user
  deleteUser(username) {
    if (!this.users.has(username)) {
      console.log("User not found.");
      return false;
    }
    this.users.delete(username);
    console.log(`User ${username} deleted successfully.`);
    return true;
  }
```

```
  // c) Update a user's password
  updatePassword(username, newPassword) {
    if (!this.users.has(username)) {
      console.log("User not found.");
      return false;
    }
    this.users.set(username, newPassword);
    console.log(`Password for ${username} updated successfully.`);
    return true;
  }
  // d) Authenticate a user
  authenticate(username, password) {
    if (this.users.get(username) === password) {
      console.log("Authentication successful.");
      return true;
    } else {
      console.log("Authentication failed.");
      return false;
    }
  }
  // e) List all registered usernames using an iterator
  listUsers() {
    console.log("Registered users:");
    for (let username of this.users.keys()) {
      console.log(username);
    }
  }
}
const auth = new UserAuth();
// Registering users
auth.register("user1", "password1");
auth.register("user2", "password2");
// Deleting a user
```

```
auth.deleteUser("user1");
// Updating a user's password
auth.updatePassword("user2", "newpassword2");
// Authenticating users
auth.authenticate("user2", "newpassword2");
// Listing all registered users
auth.listUsers();
```

7b.

Write a JavaScript program to implement Bubble Sort.

Program:

```
function bubbleSort(arr) {
   let n = arr.length;
   let swapped;
   // Loop through the array
   for (let i = 0; i < n - 1; i++) {
    swapped = false;

    // Inner loop for comparing adjacent elements
    for (let j = 0; j < n - i - 1; j++) {
     // Swap if the current element is greater than the next element
     if (arr[j] > arr[j + 1]) {
       let temp = arr[j];
       arr[j] = arr[j + 1];
       arr[j + 1] = temp;
       swapped = true;
     }
    }
    // If no elements were swapped, the array is already sorted
    if (!swapped) {
     break;
    }
  }
  return arr;
// Example usage
let array = [64, 34, 25, 12, 22, 11, 90];
console.log("Original array:", array);
let sortedArray = bubbleSort(array);
console.log("Sorted array:", sortedArray);
```

8a.

Write a JS program to read from a JSON object and display the data in a table (HTML page).

Program:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Display JSON Data in Table</title>
   <style>
      table {
         width: 100%;
         border-collapse: collapse;
      }
      table, th, td {
         border: 1px solid black;
      }
      th, td {
         padding: 10px;
         text-align: left;
      }
      th {
         background-color: #f2f2f2;
      }
   </style>
</head>
<body>
   <h1>JSON Data Table</h1>
   <table id="data-table">
      <thead>
         <tr id="table-header"></tr>
      </thead>
      <tbody id="table-body"></tbody>
```

```
    </table>
    <script>
        const jsonData = [
            { "id": 1, "name": "John Doe", "age": 28, "email": "john.doe@example.com" },
            { "id": 2, "name": "Jane Smith", "age": 34, "email": "jane.smith@example.com" },
            { "id": 3, "name": "Samuel Johnson", "age": 25, "email": "samuel.johnson@example.com" }
        ];   // Function to generate the table
        function generateTable(data) {  // Get the table header and body elements
            const tableHeader = document.getElementById('table-header');
            const tableBody = document.getElementById('table-body');           // If there is data, generate the header
          if (data.length > 0) {   // Generate table headers dynamically based on the keys in first object of data array
                const headers = Object.keys(data[0]);
                headers.forEach(header => {
                    const th = document.createElement('th');
                    th.textContent = header.charAt(0).toUpperCase() + header.slice(1); // Capitalize first letter
                    tableHeader.appendChild(th);
                });    // Generate table rows dynamically based on the data
                data.forEach(item => {
                    const tr = document.createElement('tr');
                    headers.forEach(header => {
                        const td = document.createElement('td');
                        td.textContent = item[header]; // Get value for each column
                        tr.appendChild(td);
                    });
                    tableBody.appendChild(tr);
                });
            }
        }  // Call the function to populate the table directly
        generateTable(jsonData);
    </script>
</body>
</html>
```

8b.

Implement a user registration system where, after a user registers, a confirmation email is sent. Write functions to:

1. Register a user using a Promise that simulates an API call.
2. Send a confirmation email after successful registration using another Promise.
3. Handle errors if the registration fails or if sending the email fails.
4. Use Promise. All to register multiple users concurrently and send confirmation emails for each.

Program:

```
function registerUser(username) {
  return new Promise((resolve, reject) => {
    // Simulating API call with a timeout
    setTimeout(() => {
      // Simulate successful registration with 80% probability
      if (Math.random() < 0.8) {
        console.log(`User ${username} registered successfully.`);
        resolve(username);
      } else {
        reject(`Failed to register user ${username}.`);
      }
    }, 1000);
  });
}
// Simulate an API call to send a confirmation email
function sendConfirmationEmail(username) {
  return new Promise((resolve, reject) => {
    // Simulating API call with a timeout
    setTimeout(() => {
      // Simulate successful email sending with 90% probability
      if (Math.random() < 0.9) {
        console.log(`Confirmation email sent to ${username}.`);
        resolve(`Email sent to ${username}`);
      } else {
        reject(`Failed to send email to ${username}.`);
```

```
            }
        }, 500);
    });
}
// Register a user and send confirmation email
function registerAndSendEmail(username) {
    return registerUser(username)
        .then(sendConfirmationEmail)
        .catch(error => {
            console.error(error);
        });
} // Register multiple users concurrently
function registerMultipleUsers(usernames) {
    const promises = usernames.map(username => {
        return registerAndSendEmail(username)
            .then(() => {
                console.log(`Process completed for ${username}`);
            })
            .catch(error => {
                console.error(`Error in process for ${username}: ${error}`);
            });
    });
    return Promise.all(promises)
        .then(() => {
            console.log('All users processed.');
        })
        .catch(error => {
            console.error('Error in processing users:', error);
        });
}
const users = ['user1', 'user2', 'user3', 'user4'];
registerMultipleUsers(users);
```

9a.

Write a Node JS program that accepts a port from the user and runs a node server at that port.

Program:

```
const http = require('http');
const readline = require('readline');
// Create an interface for user input
const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout
}); // Function to start the server
function startServer(port) {
  const server = http.createServer((req, res) => {
    res.writeHead(200, { 'Content-Type': 'text/plain' });
    res.end('Hello, World!\n');
  });
  server.listen(port, () => {
    console.log(`Server is running at http://localhost:${port}/`);
  });
  server.on('error', (err) => {
    console.error(`Error starting server: ${err.message}`);
  });
} // Prompt user for port number
rl.question('Enter the port number to run the server: ', (port) => {
  const portNumber = parseInt(port, 10);
  if (isNaN(portNumber) || portNumber <= 0 || portNumber > 65535) {
    console.log('Invalid port number. Please enter a number between 1 and 65535.');
    rl.close();
  } else {
    startServer(portNumber);
    rl.close();
  }
});
```

9b.

Write a NodeJS program to read from a file and display the content on screen.

Program:

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

// Define the file path
const filePath = path.join(__dirname, 'sample.txt');

// Read the file content asynchronously
fs.readFile(filePath, 'utf8', (err, data) => {
   if (err) {
      console.error('Error reading file:', err.message);
      return;
   }

   // Display the content on the screen
   console.log('File Content:');
   console.log(data);
});
```

9c.

Write a NodeJS program to accept a file name from user, text from user, if file exists append the text to the file. If not create a new file and add the text to it.

Program:

```
const fs = require('fs');
const readline = require('readline');
const path = require('path');
// Create an interface for user input
const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout
});
// Prompt user for file name
rl.question('Enter the file name: ', (fileName) => {
  const filePath = path.join(__dirname, fileName);
  // Prompt user for text to add
  rl.question('Enter the text to add: ', (text) => {
    // Check if file exists
    fs.access(filePath, fs.constants.F_OK, (err) => {
      if (err) {
        // File does not exist, create new file and write text
        fs.writeFile(filePath, text, (err) => {
          if (err) {
            console.error('Error creating file:', err.message);
          } else {
            console.log('File created and text added successfully.');
          }
          rl.close();
        });
      } else {
        // File exists, append text
        fs.appendFile(filePath, text, (err) => {
          if (err) {
            console.error('Error appending to file:', err.message);
```

```
        } else {
          console.log('Text appended to file successfully.');
        }
        rl.close();
      });
    }
  });
});
});
```

10a.

Create a student database in Mongo DB with all the details of students of a class.

Queries:

```
use school;
//switched to db school and created collection students
db.students.insertMany([
 {
  name: 'John Doe',
  age: 20,
  gender: 'Male',
  rollNumber: '101',
  subjects: ['Math', 'Science', 'English'],
  address: {
   street: '123 Main St',
   city: 'Anytown',
   zip: '12345'
  }
 },
 {
  name: 'Jane Smith',
  age: 19,
  gender: 'Female',
  rollNumber: '102',
  subjects: ['Math', 'History', 'Art'],
  address: {
   street: '456 Maple Ave',
   city: 'Othertown',
   zip: '67890'
  }
 },
 {
  name: 'Alice Johnson',
  age: 21,
```

```
    gender: 'Female',
    rollNumber: '103',
    subjects: ['Biology', 'Chemistry', 'Physics'],
    address: {
      street: '789 Oak Dr',
      city: 'Sometown',
      zip: '54321'
    }
  }
])
```

10b.

Create a form such that, based on student roll number provided by user, the student details should be fetched (using ExpressJS)

Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <title>Student Details</title>
 <script>
  function fetchStudentDetails() {
   // Get roll number from input
   const rollNumber = document.getElementById('rollNumber').value;
   const resultDiv = document.getElementById('result');

   if (rollNumber.trim() === '') {
    resultDiv.innerHTML = 'Please enter a roll number.';
    return;
   }
   // Send a GET request to fetch student details
   fetch(`http://localhost:5000/api/student/${rollNumber}`)
    .then(response => response.json())
    .then(data => {
     if (data.message) {
      resultDiv.innerHTML = data.message;
     } else {
      // Display student details
      resultDiv.innerHTML = `
       <h2>Student Details</h2>
       <p><strong>Name:</strong> ${data.name}</p>
       <p><strong>Age:</strong> ${data.age}</p>
       <p><strong>Gender:</strong> ${data.gender}</p>
```

```
            <p><strong>Roll Number:</strong> ${data.rollNumber}</p>
            <p><strong>Subjects:</strong> ${data.subjects.join(', ')}</p>
            <p><strong>Address:</strong> ${data.address.street}, ${data.address.city}, ${data.address.zip}</p>
          `;
        }
      })
      .catch(err => {
        resultDiv.innerHTML = 'Error fetching student details. Please try again later.';
      });
  }
 </script>
</head>
<body>
 <h1>Fetch Student Details</h1>
 <form onsubmit="event.preventDefault(); fetchStudentDetails();">
  <label for="rollNumber">Enter Roll Number:</label>
  <input type="text" id="rollNumber" name="rollNumber" required>
  <button type="submit">Submit</button>
 </form>
 <div id="result" style="margin-top: 20px;"></div>
</body>
</html>
```

Server.js

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

// Initialize Express
const app = express();
const port = 5000;

// Middleware
```

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

// Connect to MongoDB
mongoose.connect('mongodb://localhost:27017/school', {
  useNewUrlParser: true,
  useUnifiedTopology: true
}).then(() => {
  console.log('MongoDB connected successfully');
}).catch(err => {
  console.log('MongoDB connection error:', err);
});

// Define the student schema
const studentSchema = new mongoose.Schema({
  name: String,
  age: Number,
  gender: String,
  rollNumber: String,
  subjects: [String],
  address: {
    street: String,
    city: String,
    zip: String
  }
});

// Create the student model
const Student = mongoose.model('Student', studentSchema);

// API endpoint to fetch student details based on roll number
app.get('/api/student/:rollNumber', async (req, res) => {
  const { rollNumber } = req.params;
```

```
  try {
    const student = await Student.findOne({ rollNumber });
    if (student) {
      res.json(student);
    } else {
      res.status(404).json({ message: 'Student not found' });
    }
  } catch (error) {
    res.status(500).json({ message: 'Error fetching student details' });
  }
});

// Start the server
app.listen(port, () => {
  console.log(`Backend running at http://localhost:${port}`);
});
```

11.

Create a form such that CRUD operations can be performed on the student DB using ExpressJS

Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <title>Student CRUD</title>
 <script>
  async function createStudent() {
   const studentData = {
    name: document.getElementById('name').value,
    age: document.getElementById('age').value,
    gender: document.getElementById('gender').value,
    rollNumber: document.getElementById('rollNumber').value,
    subjects: document.getElementById('subjects').value.split(','),
    address: {
     street: document.getElementById('street').value,
     city: document.getElementById('city').value,
     zip: document.getElementById('zip').value
    }
   };

   const response = await fetch('http://localhost:5000/api/students', {
    method: 'POST',
    headers: {
     'Content-Type': 'application/json'
    },
    body: JSON.stringify(studentData)
   });

   const result = await response.json();
```

```
  alert(result.message);
}

async function getStudent() {
  const rollNumber = document.getElementById('getRollNumber').value;
  const response = await fetch(`http://localhost:5000/api/students/${rollNumber}`);
  const result = await response.json();

  if (result.message) {
    alert(result.message);
  } else {
    document.getElementById('studentDetails').innerHTML = `
      <h2>Student Details</h2>
      <p><strong>Name:</strong> ${result.name}</p>
      <p><strong>Age:</strong> ${result.age}</p>
      <p><strong>Gender:</strong> ${result.gender}</p>
      <p><strong>Roll Number:</strong> ${result.rollNumber}</p>
      <p><strong>Subjects:</strong> ${result.subjects.join(', ')}</p>
      <p><strong>Address:</strong> ${result.address.street}, ${result.address.city}, ${result.address.zip}</p>
    `;
  }
}

async function updateStudent() {
  const rollNumber = document.getElementById('updateRollNumber').value;
  const studentData = {
    name: document.getElementById('updateName').value,
    age: document.getElementById('updateAge').value,
    gender: document.getElementById('updateGender').value,
    subjects: document.getElementById('updateSubjects').value.split(','),
    address: {
      street: document.getElementById('updateStreet').value,
      city: document.getElementById('updateCity').value,
```

```
        zip: document.getElementById('updateZip').value
      }
    };

    const response = await fetch(`http://localhost:5000/api/students/${rollNumber}`, {
      method: 'PUT',
      headers: {
        'Content-Type': 'application/json'
      },
      body: JSON.stringify(studentData)
    });

    const result = await response.json();
    alert(result.message);
  }

  async function deleteStudent() {
    const rollNumber = document.getElementById('deleteRollNumber').value;
    const response = await fetch(`http://localhost:5000/api/students/${rollNumber}`, {
      method: 'DELETE'
    });

    const result = await response.json();
    alert(result.message);
  }
 </script>
</head>
<body>
 <h1>Student CRUD Operations</h1>

 <h2>Create Student</h2>
 <form onsubmit="event.preventDefault(); createStudent();">
  <label for="name">Name:</label><br>
```

```
  <input type="text" id="name" required><br>

  <label for="age">Age:</label><br>
  <input type="number" id="age" required><br>

  <label for="gender">Gender:</label><br>
  <input type="text" id="gender" required><br>

  <label for="rollNumber">Roll Number:</label><br>
  <input type="text" id="rollNumber" required><br>

  <label for="subjects">Subjects (comma separated):</label><br>
  <input type="text" id="subjects" required><br>

  <label for="street">Street:</label><br>
  <input type="text" id="street" required><br>

  <label for="city">City:</label><br>
  <input type="text" id="city" required><br>

  <label for="zip">Zip:</label><br>
  <input type="text" id="zip" required><br>

  <button type="submit">Create Student</button>
</form>

<h2>Get Student Details</h2>
<form onsubmit="event.preventDefault(); getStudent();">
  <label for="getRollNumber">Roll Number:</label><br>
  <input type="text" id="getRollNumber" required><br>
  <button type="submit">Get Details</button>
</form>
```

```
<div id="studentDetails"></div>

<h2>Update Student</h2>
<form onsubmit="event.preventDefault(); updateStudent();">
 <label for="updateRollNumber">Roll Number:</label><br>
 <input type="text" id="updateRollNumber" required><br>

 <label for="updateName">Name:</label><br>
 <input type="text" id="updateName" required><br>

 <label for="updateAge">Age:</label><br>
 <input type="number" id="updateAge" required><br>

 <label for="updateGender">Gender:</label><br>
 <input type="text" id="updateGender" required><br>

 <label for="updateSubjects">Subjects (comma separated):</label><br>
 <input type="text" id="updateSubjects" required><br>

 <label for="updateStreet">Street:</label><br>
 <input type="text" id="updateStreet" required><br>

 <label for="updateCity">City:</label><br>
 <input type="text" id="updateCity" required><br>

 <label for="updateZip">Zip:</label><br>
 <input type="text" id="updateZip" required><br>

 <button type="submit">Update Student</button>
</form>

<h2>Delete Student</h2>
<form onsubmit="event.preventDefault(); deleteStudent();">
```

```
    <label for="deleteRollNumber">Roll Number:</label><br>
    <input type="text" id="deleteRollNumber" required><br>
    <button type="submit">Delete Student</button>
  </form>
</body>
</html>
```

Server.js

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

// Initialize the app
const app = express();
const port = 5000;

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

// Connect to MongoDB
mongoose.connect('mongodb://localhost:27017/school', {
  useNewUrlParser: true,
  useUnifiedTopology: true
}).then(() => {
  console.log('MongoDB connected successfully');
}).catch(err => {
  console.log('MongoDB connection error:', err);
});

// Define the Student schema
const studentSchema = new mongoose.Schema({
```

```
  name: String,
  age: Number,
  gender: String,
  rollNumber: { type: String, unique: true },
  subjects: [String],
  address: {
    street: String,
    city: String,
    zip: String
  }
});

// Create the Student model
const Student = mongoose.model('Student', studentSchema);

// API endpoints

// Create: Add a new student
app.post('/api/students', async (req, res) => {
  const { name, age, gender, rollNumber, subjects, address } = req.body;

  const student = new Student({
    name,
    age,
    gender,
    rollNumber,
    subjects,
    address
  });

  try {
    await student.save();
    res.status(201).json({ message: 'Student created successfully' });
```

```
  } catch (error) {
    res.status(500).json({ message: 'Error creating student', error });
  }
});

// Read: Get a student by roll number
app.get('/api/students/:rollNumber', async (req, res) => {
  const { rollNumber } = req.params;

  try {
    const student = await Student.findOne({ rollNumber });
    if (student) {
      res.json(student);
    } else {
      res.status(404).json({ message: 'Student not found' });
    }
  } catch (error) {
    res.status(500).json({ message: 'Error fetching student', error });
  }
});

// Update: Update student details
app.put('/api/students/:rollNumber', async (req, res) => {
  const { rollNumber } = req.params;
  const { name, age, gender, subjects, address } = req.body;

  try {
    const student = await Student.findOneAndUpdate(
      { rollNumber },
      { name, age, gender, subjects, address },
      { new: true }
    );
```

```
    if (student) {
      res.json({ message: 'Student updated successfully', student });
    } else {
      res.status(404).json({ message: 'Student not found' });
    }
  } catch (error) {
    res.status(500).json({ message: 'Error updating student', error });
  }
});

// Delete: Delete a student by roll number
app.delete('/api/students/:rollNumber', async (req, res) => {
  const { rollNumber } = req.params;

  try {
    const student = await Student.findOneAndDelete({ rollNumber });
    if (student) {
      res.json({ message: 'Student deleted successfully' });
    } else {
      res.status(404).json({ message: 'Student not found' });
    }
  } catch (error) {
    res.status(500).json({ message: 'Error deleting student', error });
  }
});

// Start the server
app.listen(port, () => {
  console.log(`Server running at http://localhost:${port}`);
});
```

12.

You are given a string str, and your task is to find the number of vowels and non-vowels in the string.

Sample Input-1:

Hello

Sample Output-1:

Vowels: 2

Non-Vowels: 3

Program :

```
console.log("Enter a string:");
const readline = require('readline');

// Set up readline interface for reading input
const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout
});
// Function to count vowels in a string
const countVowels = (str) => {
  const vowels = 'aeiouAEIOU';
  return str.split('').filter(char => vowels.includes(char)).length;
};
// Read input from user
rl.on('line', (input) => {
  const len = input.length;
  const count = countVowels(input);
  console.log(`Vowels: ${count}\nNon-Vowels: ${len - count}`);
  rl.close();
});
```

13.

You are given two sorted arrays, arr1 and arr2. Your task is to merge these two arrays into a single sorted array. Both input arrays are already sorted in non-decreasing order. The output should also be a sorted array that contains all elements from both arr1 and arr2.

Sample Input:

1 3 5

2 4 6

Sample Output:

1 2 3 4 5 6

Program:

```
const readline = require('readline'); // Set up readline interface for reading input
const rl = readline.createInterface({
 input: process.stdin,
 output: process.stdout
}); // Function to merge two sorted arrays
 const mergeSortedArrays = (arr1, arr2) => {
 let merged = [];
 let i = 0, j = 0;  // Merge arrays while both have elements
 while (i < arr1.length && j < arr2.length) {
  if (arr1[i] < arr2[j]) {
   merged.push(arr1[i]);
   i++;
  } else {
   merged.push(arr2[j]);
   j++;
  }
 }
 while (i < arr1.length) {   // Add remaining elements from arr1 (if any)
  merged.push(arr1[i]);
  i++;
 }
 while (j < arr2.length) {   // Add remaining elements from arr2 (if any)
  merged.push(arr2[j]);
  j++;
```

```
  }
  return merged;
}; // Function to read input and process arrays
const processInput = () => {
  rl.question('', (input1) => {
    const arr1 = input1.split(' ').map(Number); // Convert input string to an array of numbers
    rl.question('', (input2) => {
      const arr2 = input2.split(' ').map(Number); // Convert input string to an array of numbers
      const mergedArray = mergeSortedArrays(arr1, arr2);
      console.log("merge sorted array : " + mergedArray.join(' '));
      rl.close();
    });
  });
};
processInput();
```

14.

You are given a collection of books objects in a library. Request the user's library ID and assist them in determining whether the book is available at the library.

Sample Input 1:

5

Sample Output 1:

Book with libraryId 5 : 'The Old Man and the Sea' by 'Ernest Hemingway' is available.

Program:

```
let library = [
  {
    title: 'The Great Gatsby',
    author: 'Francis Scott Fitzgerald',
    libraryId: 1
  },
  {
    title: 'The Catcher in the Rye',
    author: 'J. D. Salinger',
    libraryId: 2
  },
  {
    title: 'The Grapes of Wrath',
    author: 'John Steinbeck, Robert DeMott',
    libraryId: 3
  },
  {
    title: 'The Road Ahead',
    author: 'Bill Gates',
    libraryId: 4
  },
  {
    title: 'The Old Man and the Sea',
    author: 'Ernest Hemingway',
    libraryId: 5
```

```
    },
    {
        title: 'Of Mice and Men',
        author: 'John Steinbeck',
        libraryId: 6
    },
    {
        title: 'A Song of Ice and Fire',
        author: 'R. R. Martin',
        libraryId: 7
    }
];
let readline = require('readline').createInterface({ input: process.stdin, output: process.stdout });
let libraryId;
readline.question('', function (line) {
    libraryId = parseInt(line);
    solution(libraryId);
    readline.close();
});
function solution(libraryId) { // Find the book with the matching libraryId
    let book = library.find(item => item.libraryId === libraryId);
    if (book) {
        // If book exists, print the details
        console.log(`Book with libraryId ${libraryId} : '${book.title}' by '${book.author}' is available.`);
    } else {
        // If book does not exist, print not available message
        console.log(`Sorry! Book with libraryId ${libraryId} is not available.`);
    }
}
```

15.

Convert an Array of Objects to an Array of Strings Using map().

Task: You have an array of user objects, and you need to create an array of their names.

//output

['Alice', 'Bob', 'Charlie', 'David', 'Eve', 'Frank', 'Grace', 'Hank', 'Ivy']

Program:

```
const users = [
   { name: 'Alice', age: 21 },
   { name: 'Bob', age: 22 },
   { name: 'Charlie', age: 21 },
   { name: 'David', age: 22 },
   { name: 'Eve', age: 23 },
   { name: 'Frank', age: 21 },
   { name: 'Grace', age: 23 },
   { name: 'Hank', age: 24 },
   { name: 'Ivy', age: 22 }
 ];

 // Using map to create an array of user names
 const userNames = users.map(user => user.name);

 console.log(userNames);
```

16.

Find Users Who Are Adults Using filter()

Task: Given an array of users with ages, filter out those who are 18 years or older.

Output:

[

{ name: 'Bob', age: 20 },

{ name: 'David', age: 18 },

{ name: 'Eve', age: 22 },

{ name: 'Grace', age: 19 },

{ name: 'Hank', age: 21 },

{ name: 'Jack', age: 18 }

]

Program:

```
const users = [
   { name: 'Alice', age: 17 },
   { name: 'Bob', age: 20 },
   { name: 'Charlie', age: 16 },
   { name: 'David', age: 18 },
   { name: 'Eve', age: 22 },
   { name: 'Frank', age: 15 },
   { name: 'Grace', age: 19 },
   { name: 'Hank', age: 21 },
   { name: 'Ivy', age: 17 },
   { name: 'Jack', age: 18 }
 ];

 // Implement your logic here and store the result as "adults".
 const adults = users.filter(user => user.age >= 18);

 console.log(adults);
```

17.

Group People by Age Using reduce(). Task: Given an array of people, group them by their ages.

Output:

{

'21': ['Alice', 'Charlie', 'Frank'],

'22': ['Bob', 'David', 'Ivy'],

'23': ['Eve', 'Grace'],

'24': ['Hank']

}

Program:

```
const users = [
   { name: 'Alice', age: 21 },
   { name: 'Bob', age: 22 },
   { name: 'Charlie', age: 21 },
   { name: 'David', age: 22 },
   { name: 'Eve', age: 23 },
   { name: 'Frank', age: 21 },
   { name: 'Grace', age: 23 },
   { name: 'Hank', age: 24 },
   { name: 'Ivy', age: 22 }
 ];
 // Implement your logic here  and store the result as "groupedByAge".
 const groupedByAge = users.reduce((acc, user) => {
   // If the age group doesn't exist in the accumulator, create it
   if (!acc[user.age]) {
     acc[user.age] = [];
   }
   // Push the user's name into the corresponding age group
   acc[user.age].push(user.name);
   return acc;
 }, {});
 console.log(groupedByAge);
```

18.

You are tasked with simulating an ice cream production process using JavaScript.

The user will select a fruit, holder, and topping, and the program will simulate

the production process with specific delays for each step.

Input:

A fruit (e.g., "apple").

A holder (e.g., "cup").

A topping (e.g., "chocolate").

Ouptut: (i.e., the following production steps are executed with delays:)

1. Production starts immediately.

2. After 1 second: The fruit is chopped.

3. After 1 second: Liquid (water and ice) is added.

4. After 2 seconds: The machine starts.

5. After 2 seconds: Ice cream is placed in the holder.

6. After 3 seconds: Topping is added.

7. After 2 seconds: Ice cream is served.

Program:

```
const readline = require('readline');
let stocks = {
  Fruits: ["strawberry", "grapes", "banana", "apple"],
  liquid: ["water", "ice"],
  holder: ["cone", "cup", "stick"],
  toppings: ["chocolate", "peanuts"],
};
// Set up readline interface for user input
const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout
});
// Ask for fruit selection
const askFruit = () => {
  rl.question(`Choose a fruit (${stocks.Fruits.join(', ')}): `, (fruit) => {
    if (stocks.Fruits.includes(fruit.toLowerCase())) {
```

```
      askHolder(fruit);
    } else {
      console.log("Invalid fruit selection. Please try again.");
      askFruit();
    }
  });
};
// Ask for holder selection
const askHolder = (fruit) => {
  rl.question(`Choose a holder (${stocks.holder.join(', ')}): `, (holder) => {
    if (stocks.holder.includes(holder.toLowerCase())) {
      askTopping(fruit, holder);
    } else {
      console.log("Invalid holder selection. Please try again.");
      askHolder(fruit);
    }
  });
};
// Ask for topping selection
const askTopping = (fruit, holder) => {
  rl.question(`Choose a topping (${stocks.toppings.join(', ')}): `, (topping) => {
    if (stocks.toppings.includes(topping.toLowerCase())) {
      production(fruit, holder, topping);
    } else {
      console.log("Invalid topping selection. Please try again.");
      askTopping(fruit, holder);
    }
  });
};
// Ice cream production process
const production = (fruit_name, holder_name, topping_name) => {
  console.log(`${fruit_name} was selected`);
  console.log("Production has started");
```

```
  setTimeout(() => {
    console.log(`The ${fruit_name} fruit has been chopped`);
    setTimeout(() => {
      console.log("Water and ice added");
      setTimeout(() => {
        console.log("Start the machine");
        setTimeout(() => {
          console.log(`Ice cream placed on ${holder_name}`);
          setTimeout(() => {
            console.log(`${topping_name} as toppings`);
            setTimeout(() => {
              console.log("Serve Ice Cream");
              rl.close(); // Close readline after completion
            }, 2000); // Serve ice cream after 2 seconds
          }, 3000); // Add topping after 3 seconds
        }, 2000); // Place ice cream after 2 seconds
      }, 2000); // Start machine after 2 seconds
    }, 1000); // Add liquid after 1 second
  }, 1000); // Chop fruit after 1 second
};
// Start the process
askFruit();
```

19.

Write a JavaScript program that simulates a food delivery process using asynchronous functions. The program should:

1. Simulate placing an order with a delay of 1 second.
2. Simulate preparing food with a delay of 3 seconds.
3. Simulate packaging the food and delivering it concurrently, where packaging takes 1 second and delivery takes 2 seconds.
4. Log messages at each step to indicate the progress of the food delivery process.

Program:

```
const placeOrder = () => {
   return new Promise((resolve) => {
     setTimeout(() => {
       console.log("Order placed.");
       resolve();
     }, 1000); // 1 second delay
   });
 };

 const prepareFood = () => {
   return new Promise((resolve) => {
     setTimeout(() => {
       console.log("Food prepared.");
       resolve();
     }, 3000); // 3 seconds delay
   });
 };

 const deliverFood = () => {
   return new Promise((resolve, reject) => {
     setTimeout(() => {
       console.log("Food delivered.");
       resolve();
     }, 2000); // 2 seconds delay
   });
```

```
};

const packageFood = () => {
  return new Promise((resolve) => {
    setTimeout(() => {
      console.log("Food packaged.");
      resolve();
    }, 1000); // 1 second delay
  });
};

const startFoodDeliveryProcess = async () => {
  try {
    // Step 1: Place the order
    await placeOrder();

    // Step 2: Prepare the food
    await prepareFood();

    // Step 3: Start both packaging and delivery concurrently
    await Promise.all([packageFood(), deliverFood()]);

    console.log("Food delivery process complete.");
  } catch (error) {
    console.log("Error during the food delivery process:", error);
  }
};

startFoodDeliveryProcess();
```

20.

Create an html page and js file as described below.

Problem Statement: Basic Web Server with Routing in Node.js

You are tasked with creating a simple Node.js web server that serves different

pages based on the URL path.

The server should: Serve an HTML page with a dynamic message in the content.

Handle routes for:

=> Products page: Display the products as shown in the class.

Use the http module for server creation and fs module to read the HTML file.

***Index.html*(*Home Page*)**

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Home Page</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <h1>Welcome to Our Website!</h1>
    <p>This is the homepage with a dynamic message.</p>
    <a href="/products">Go to Products</a>
</body>
</html>
```

products.html (Products Page)

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

```
<body>
    <h1>Our Products</h1>
    <ul>
        <li>Product 1: Amazing Widget</li>
        <li>Product 2: Super Gadget</li>
        <li>Product 3: Cool Gizmo</li>
    </ul>
    <a href="/">Back to Home</a>
</body>
</html>
```

server.js

```
const http = require('http');
const fs = require('fs');
const path = require('path');
// Create the server
const server = http.createServer((req, res) => {
    const url = req.url; // Get the URL of the request
    if (url === '/') {
        // Serve the homepage
        fs.readFile(path.join(__dirname, 'index.html'), 'utf-8', (err, data) => {
            if (err) {
                res.writeHead(500, { 'Content-Type': 'text/plain' });
                res.end('Error reading file');
                return;
            }
            res.writeHead(200, { 'Content-Type': 'text/html' });
            res.end(data);
        });
    } else if (url === '/products') {
        // Serve the products page
        fs.readFile(path.join(__dirname, 'products.html'), 'utf-8', (err, data) => {
            if (err) {
                res.writeHead(500, { 'Content-Type': 'text/plain' });
```

```
            res.end('Error reading file');
            return;
        }
        res.writeHead(200, { 'Content-Type': 'text/html' });
        res.end(data);
    });
  } else {
    // Serve a 404 page for unknown routes
    res.writeHead(404, { 'Content-Type': 'text/plain' });
    res.end('Page not found');
  }
});
// Define the server port
const port = 3000;
server.listen(port, () => {
  console.log(`Server is running on http://localhost:${port}`);
});
```

21.

Create a simple Node.js application and serves it via an HTTP server

(without using Express).

The response of the server should display a text

"Welcome to KMIT!" inside an <h1> tag on the webpage.

Ensure that you have submitted the correct URL,

checks for the presence of the <h1> tag, and validates the text content.

server.js

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

// Create the server
const server = http.createServer((req, res) => {
   // Set the response header
   res.writeHead(200, { 'Content-Type': 'text/html' });

   // Send the HTML response with the <h1> tag
   res.end('<html><body><h1>Welcome to KMIT!</h1></body></html>');
});

// Define the server port
const port = 3000;
server.listen(port, () => {
   console.log(`Server is running at http://localhost:${port}`);
});
```

22.

Create an html page and js file as described below.

Problem Statement: Basic Web Server with Routing in Node.js. You are tasked with creating a simple Node.js web server that serves different pages based on the URL path. The server should: Serve an HTML page with a dynamic message in the content.

Handle routes for:

=> '/' or '/home': Display the home page content.

=> '/about': Display the about page content.

=> '/contact': Display the contact page content.

=> 'Any other route': Display an error message (404).

Use the http module for server creation and fs module to read the HTML file.

index.html **(Home Page)**

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Home Page</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <header>
    <h1>Welcome to Our School Website!</h1>
    <nav>
      <a href="/home">Home</a> |
      <a href="/about">About</a> |
      <a href="/contact">Contact</a>
    </nav>
  </header>

  <section>
    <h2>About Our School</h2>
    <p>We provide a world-class education for students. Our teachers are passionate and our curriculum is tailored to meet the needs of every student.</p>
```

```
    </section>

    <footer>
        <p>&copy; 2024 Our School. All rights reserved.</p>
    </footer>
</body>
</html>
```

***about.html* (About Page)**

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>About Page</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <header>
        <h1>About Us</h1>
        <nav>
            <a href="/home">Home</a> |
            <a href="/about">About</a> |
            <a href="/contact">Contact</a>
        </nav>
    </header>

    <section>
        <h2>Our Mission</h2>
        <p>Our mission is to provide high-quality education and create a nurturing environment for students to excel academically and personally.</p>

        <h2>Our Values</h2>
        <ul>
```

```
        <li>Excellence</li>
        <li>Innovation</li>
        <li>Integrity</li>
        <li>Community</li>
      </ul>
    </section>

    <footer>
      <p>&copy; 2024 Our School. All rights reserved.</p>
    </footer>
  </body>
  </html>
```

contact.html (Contact Page)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Contact Page</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <header>
        <h1>Contact Us</h1>
        <nav>
            <a href="/home">Home</a> |
            <a href="/about">About</a> |
            <a href="/contact">Contact</a>
        </nav>
    </header>
    <section>
        <h2>Contact Form</h2>
        <form action="/submit-contact" method="POST">
```

```
            <label for="name">Your Name:</label>
            <input type="text" id="name" name="name" required>
            <label for="email">Your Email:</label>
            <input type="email" id="email" name="email" required>
            <label for="message">Your Message:</label>
            <textarea id="message" name="message" rows="4" required></textarea>
            <button type="submit">Submit</button>
        </form>
        <h2>Our Location</h2>
        <p>123 School Street, City, State, 12345</p>
        <h2>Call Us</h2>
        <p>+1 (555) 123-4567</p>
        <h2>Email</h2>
        <p>info@ourschool.com</p>
    </section>
    <footer>
        <p>&copy; 2024 Our School. All rights reserved.</p>
    </footer>
</body>
</html>
```

server.js

```
const http = require('http');
const fs = require('fs');
const path = require('path');
// Create an HTTP server
const server = http.createServer((req, res) => {
    let filePath = '';
    let statusCode = 200;
    // Handle different routes
    switch (req.url) {
        case '/':
        case '/home':
            filePath = path.join(__dirname, 'home.html');
```

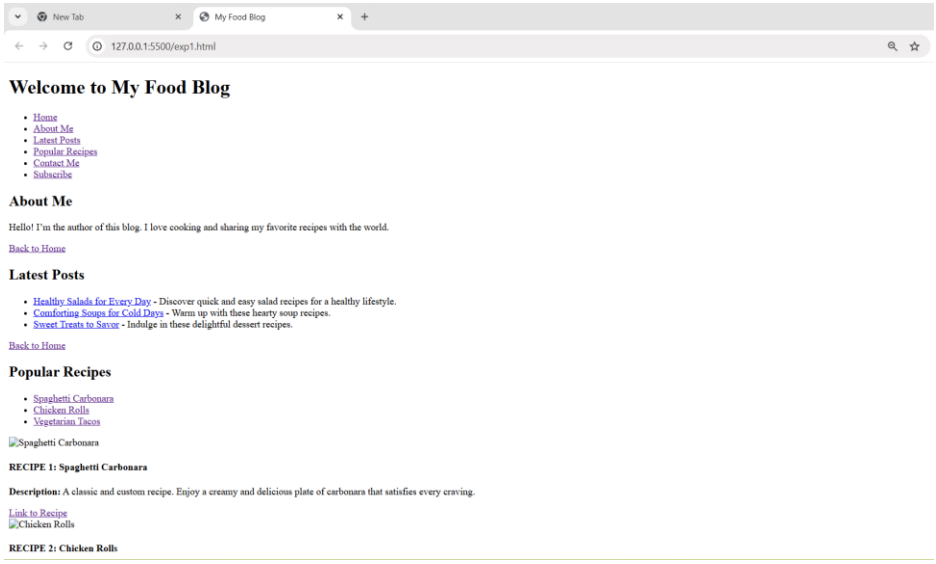
```
            break;
        case '/about':
            filePath = path.join(__dirname, 'about.html');
            break;
        case '/contact':
            filePath = path.join(__dirname, 'contact.html');
            break;
        default:
            filePath = path.join(__dirname, '404.html');
            statusCode = 404;
            break;
    }
    // Read and serve the HTML file
    fs.readFile(filePath, 'utf-8', (err, data) => {
        if (err) {
            res.writeHead(500, { 'Content-Type': 'text/html' });
            res.end('<h1>Internal Server Error</h1>');
        } else {
            res.writeHead(statusCode, { 'Content-Type': 'text/html' });
            res.end(data);
        }
    });
});
// Set the server to listen on port 5000
server.listen(5000, () => {
    console.log('Server is running at http://localhost:5000');
});
```

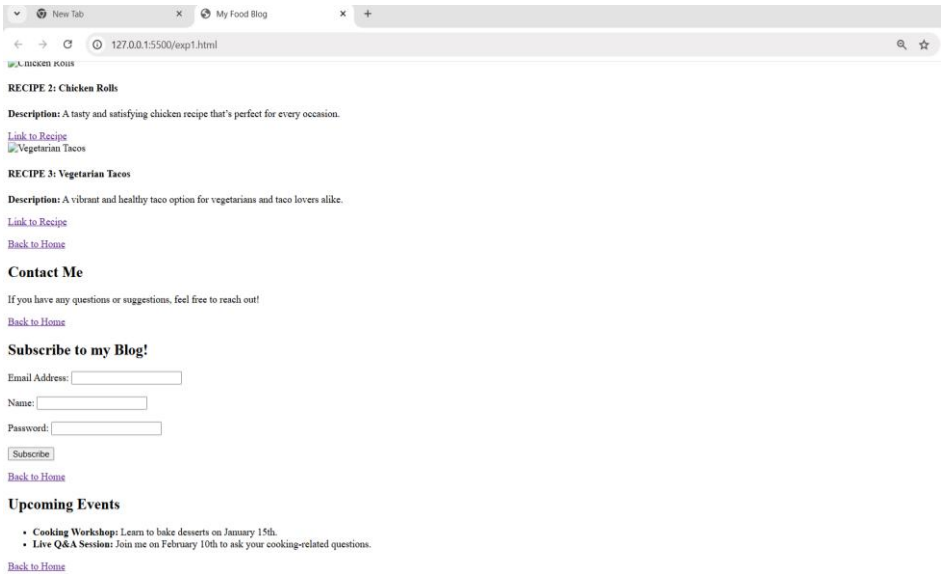
style.css

```
body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 20px;
    background-color: #f4f4f4;
```

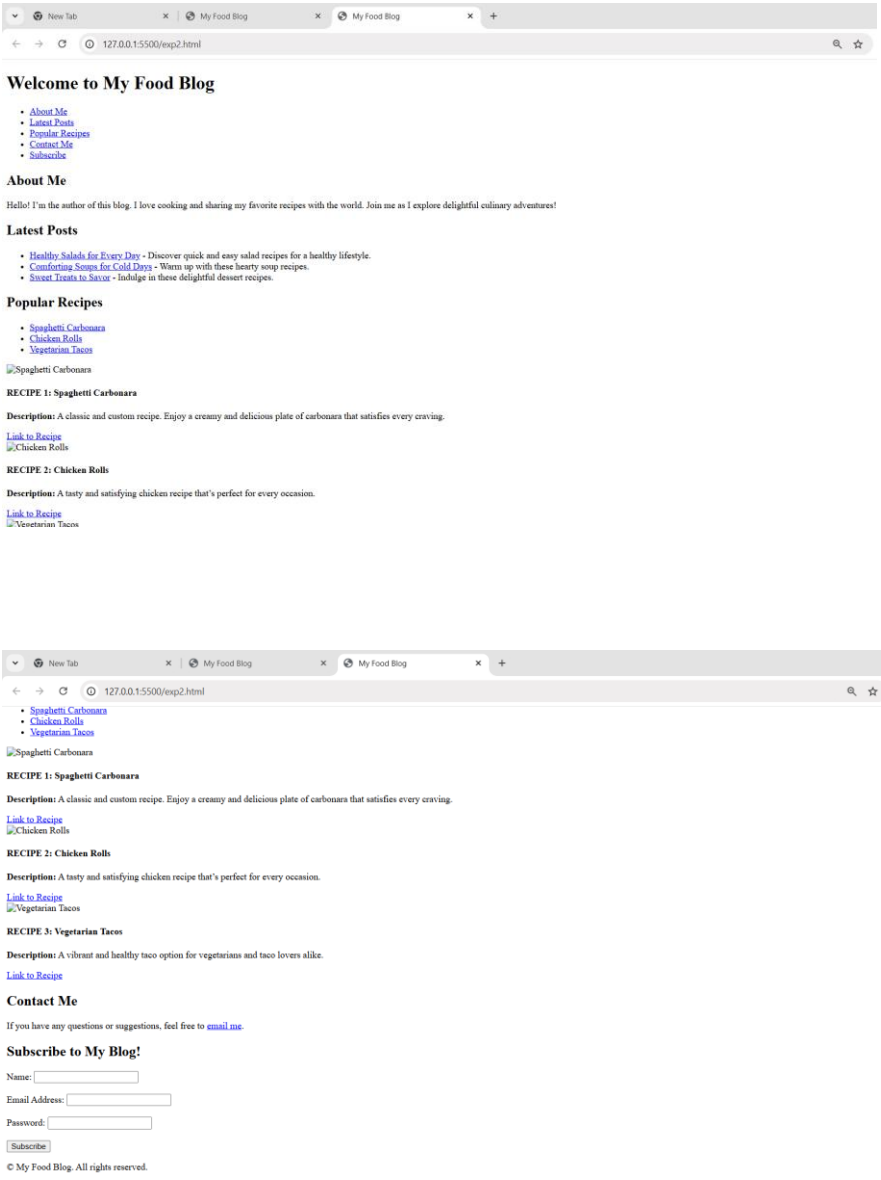
```
}
h1 {
    color: #333;
}
a {
    color: #3498db;
    text-decoration: none;
}
a:hover {
    text-decoration: underline;
}
nav {
    margin-top: 20px;
}
```

1


New Tab
My Food Blog
127.0.0.1:5500/exp1.html
Welcome to My Food Blog
Home
About Me
Latest Posts
Popular Recipes
Contact Me
Subscribe
About Me
Hello! I'm the author of this blog. I love cooking and sharing my favorite recipes with the world.
Back to Home
Latest Posts
Healthy Salads for Every Day - Discover quick and easy salad recipes for a healthy lifestyle.
Comforting Soups for Cold Days - Warm up with these hearty soup recipes.
Sweet Treats to Savor - Indulge in these delightful dessert recipes.
Back to Home
Popular Recipes
Spaghetti Carbonara
Chicken Rolls
Vegetarian Tacos
Spaghetti Carbonara
RECIPE 1: Spaghetti Carbonara
Description: A classic and custom recipe. Enjoy a creamy and delicious plate of carbonara that satisfies every craving.
Link to Recipe
Chicken Rolls
RECIPE 2: Chicken Rolls

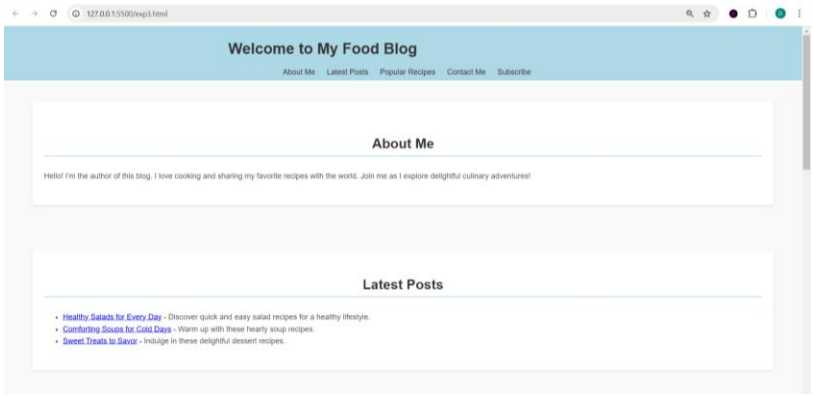

New Tab
My Food Blog
127.0.0.1:5500/exp1.html
RECIPE 2: Chicken Rolls
Description: A tasty and satisfying chicken recipe that's perfect for every occasion.
Link to Recipe
Vegetarian Tacos
RECIPE 3: Vegetarian Tacos
Description: A vibrant and healthy taco option for vegetarians and taco lovers alike.
Link to Recipe
Back to Home
Contact Me
If you have any questions or suggestions, feel free to reach out!
Back to Home
Subscribe to my Blog!
Email Address:
Name:
Password:
Subscribe
Back to Home
Upcoming Events
Cooking Workshop: Learn to bake desserts on January 15th.
Live Q&A Session: Join me on February 10th to ask your cooking-related questions.
Back to Home

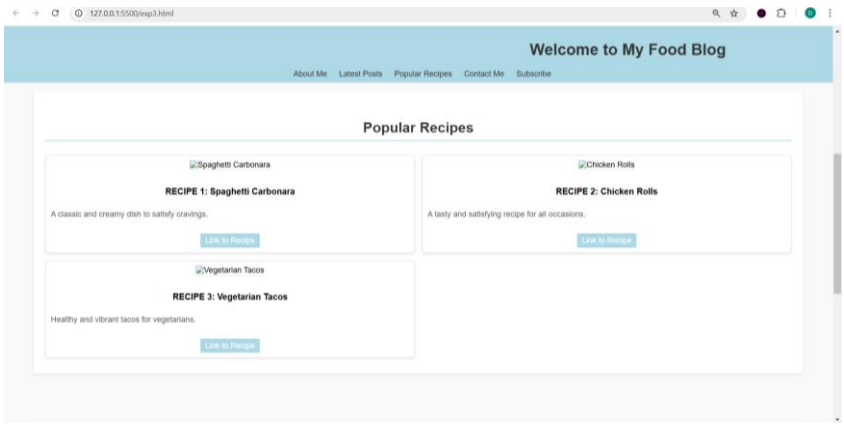
2

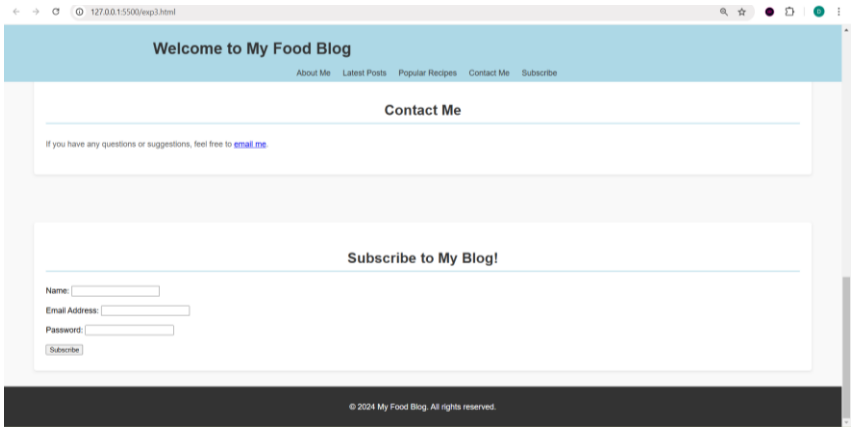

New Tab
My Food Blog
My Food Blog
127.0.0.1:5500/exp2.html
Welcome to My Food Blog
About Me
Latest Posts
Popular Recipes
Contact Me
Subscribe
About Me
Hello! I'm the author of this blog. I love cooking and sharing my favorite recipes with the world. Join me as I explore delightful culinary adventures!
Latest Posts
Healthy Salads for Every Day - Discover quick and easy salad recipes for a healthy lifestyle.
Comforting Soups for Cold Days - Warm up with these hearty soup recipes.
Sweet Treats to Savor - Indulge in these delightful dessert recipes.
Popular Recipes
Spaghetti Carbonara
Chicken Rolls
Vegetarian Tacos
Spaghetti Carbonara
RECIPE 1: Spaghetti Carbonara
Description: A classic and custom recipe. Enjoy a creamy and delicious plate of carbonara that satisfies every craving.
Link to Recipe
Chicken Rolls
RECIPE 2: Chicken Rolls
Description: A tasty and satisfying chicken recipe that's perfect for every occasion.
Link to Recipe
Vegetarian Tacos

New Tab
My Food Blog
My Food Blog
127.0.0.1:5500/exp2.html
Spaghetti Carbonara
Chicken Rolls
Vegetarian Tacos
Spaghetti Carbonara
RECIPE 1: Spaghetti Carbonara
Description: A classic and custom recipe. Enjoy a creamy and delicious plate of carbonara that satisfies every craving.
Link to Recipe
Chicken Rolls
RECIPE 2: Chicken Rolls
Description: A tasty and satisfying chicken recipe that's perfect for every occasion.
Link to Recipe
Vegetarian Tacos
RECIPE 3: Vegetarian Tacos
Description: A vibrant and healthy taco option for vegetarians and taco lovers alike.
Link to Recipe
Contact Me
If you have any questions or suggestions, feel free to email me.
Subscribe to My Blog!
Name:
Email Address:
Password:
Subscribe
© My Food Blog. All rights reserved.

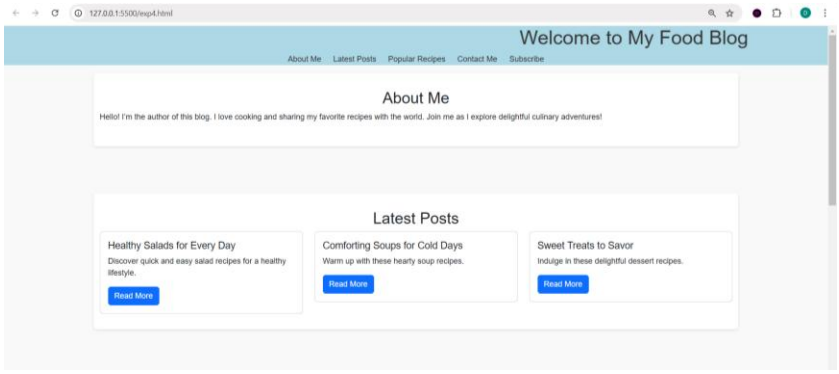
3

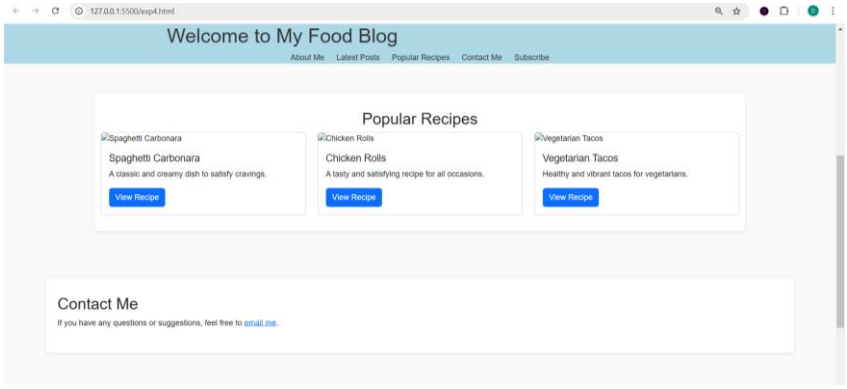

127.0.0.1:5500/exp3.html
Welcome to My Food Blog
About Me Latest Posts Popular Recipes Contact Me Subscribe
About Me
Hello! I'm the author of this blog. I love cooking and sharing my favorite recipes with the world. Join me as I explore delightful culinary adventures!
Latest Posts
Healthy Salads for Every Day - Discover quick and easy salad recipes for a healthy lifestyle.
Comforting Soups for Cold Days - Warm up with these hearty soup recipes.
Sweet Treats to Savor - Indulge in these delightful dessert recipes.

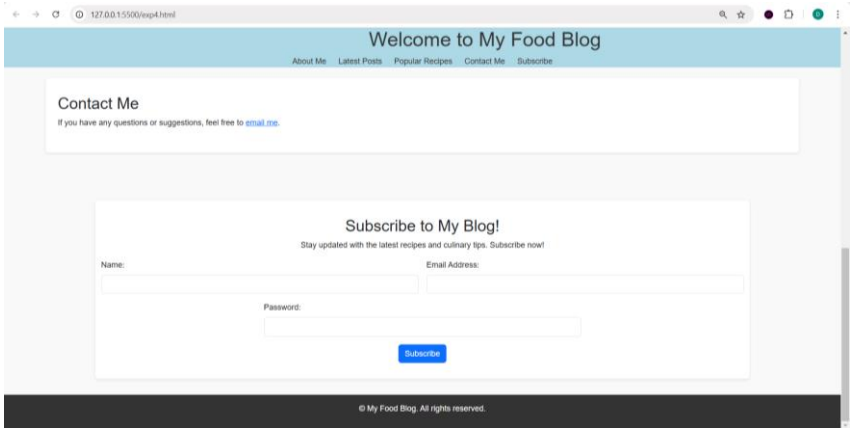

127.0.0.1:5500/exp3.html
Welcome to My Food Blog
About Me Latest Posts Popular Recipes Contact Me Subscribe
Popular Recipes
Spaghetti Carbonara
RECIPE 1: Spaghetti Carbonara
A classic and creamy dish to satisfy cravings.
Link to Recipe
Chicken Rolls
RECIPE 2: Chicken Rolls
A tasty and satisfying recipe for all occasions.
Link to Recipe
Vegetarian Tacos
RECIPE 3: Vegetarian Tacos
Healthy and vibrant tacos for vegetarians.
Link to Recipe


127.0.0.1:5500/exp3.html
Welcome to My Food Blog
About Me Latest Posts Popular Recipes Contact Me Subscribe
Contact Me
If you have any questions or suggestions, feel free to email me.
Subscribe to My Blog!
Name:
Email Address:
Password:
Subscribe
© 2024 My Food Blog. All rights reserved.

4


127.0.0.1:5500/exp4.html
Welcome to My Food Blog
About Me
Latest Posts
Popular Recipes
Contact Me
Subscribe
About Me
Hello! I'm the author of this blog. I love cooking and sharing my favorite recipes with the world. Join me as I explore delightful culinary adventures!
Latest Posts
Healthy Salads for Every Day
Discover quick and easy salad recipes for a healthy lifestyle.
Read More
Comforting Soups for Cold Days
Warm up with these hearty soup recipes.
Read More
Sweet Treats to Savor
Indulge in these delightful dessert recipes.
Read More


127.0.0.1:5500/exp4.html
Welcome to My Food Blog
About Me
Latest Posts
Popular Recipes
Contact Me
Subscribe
Popular Recipes
Spaghetti Carbonara
Spaghetti Carbonara
A classic and creamy dish to satisfy cravings.
View Recipe
Chicken Rolls
Chicken Rolls
A tasty and satisfying recipe for all occasions.
View Recipe
Vegetarian Tacos
Vegetarian Tacos
Healthy and vibrant tacos for vegetarians.
View Recipe
Contact Me
If you have any questions or suggestions, feel free to email me.


127.0.0.1:5500/exp4.html
Welcome to My Food Blog
About Me
Latest Posts
Popular Recipes
Contact Me
Subscribe
Contact Me
If you have any questions or suggestions, feel free to email me.
Subscribe to My Blog!
Stay updated with the latest recipes and culinary tips. Subscribe now!
Name:
Email Address:
Password:
Subscribe
© My Food Blog. All rights reserved.

5a

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
[Running] node "c:\Users\Dell\OneDrive\Desktop\wt_lab\exp5(a).js"
Input String:  The Quick Brown Fox
Swapped Case String:  tHE qUICK bROWN fOX

[Done] exited with code=0 in 0.541 seconds
```

5b

```
[Running] node "c:\Users\Dell\OneDrive\Desktop\wt_lab\exp5(b).js"
Array:  [
  1, 2, 3, 2, 4,
  5, 2, 3, 3, 3
]
Most Frequent Item:  3
Frequency:  4

[Done] exited with code=0 in 0.5 seconds
```

5c

```
[Running] node "c:\Users\Dell\OneDrive\Desktop\wt_lab\exp(c).js"
[ 1, 2, 3, 4, 5 ]

[Done] exited with code=0 in 0.406 seconds
```

6a

```
[Running] node "c:\Users\Dell\OneDrive\Desktop\WT\day-6.js"
Binary Search : 3

[Done] exited with code=0 in 0.543 seconds
```

6b

```
[Running] node "c:\Users\Dell\OneDrive\Desktop\WT\day-6.js"
The list properties are : name,age,city

[Done] exited with code=0 in 0.411 seconds
```

6c

```
[Running] node "c:\Users\Dell\OneDrive\Desktop\WT\day-6.js"
true
false

[Done] exited with code=0 in 0.407 seconds
```

6d.

```
[Running] node "c:\Users\Dell\OneDrive\Desktop\WT\day-6.js"
Quick Sort : 1,1,2,3,6,8,10

[Done] exited with code=0 in 0.438 seconds
```

12

```
PS C:\Users\Dell\OneDrive\Desktop\wt_lab> node exp12.js
Enter a string:
Hello
Vowels: 2
Non-Vowels: 3
```

13.

```
PS C:\Users\Dell\OneDrive\Desktop\wt_lab> node exp13.js
1 3 6
2 4 7
merge sorted array : 1 2 3 4 6 7
```

15

```
PS C:\Users\Dell\OneDrive\Desktop\wt_lab> node exp15.js
[
  'Alice',   'Bob',
  'Charlie', 'David',
  'Eve',     'Frank',
  'Grace',   'Hank',
  'Ivy'
]
```

16

```
PS C:\Users\Dell\OneDrive\Desktop\wt_lab> node exp16.js
[
  { name: 'Bob', age: 20 },
  { name: 'David', age: 18 },
  { name: 'Eve', age: 22 },
  { name: 'Grace', age: 19 },
  { name: 'Hank', age: 21 },
  { name: 'Jack', age: 18 }
]
PS C:\Users\Dell\OneDrive\Desktop\wt_lab>
```

17

```
PS C:\Users\Dell\OneDrive\Desktop\wt_lab> node exp17.js
{
  '21': [ 'Alice', 'Charlie', 'Frank' ],
  '22': [ 'Bob', 'David', 'Ivy' ],
  '23': [ 'Eve', 'Grace' ],
  '24': [ 'Hank' ]
}
PS C:\Users\Dell\OneDrive\Desktop\wt_lab>
```

18

```
PS C:\Users\Dell\OneDrive\Desktop\wt_lab> node exp18.js
Choose a fruit (strawberry, grapes, banana, apple): strawberry
Choose a holder (cone, cup, stick): cup
Choose a topping (chocolate, peanuts): chocolate
strawberry was selected
Production has started
The strawberry fruit has been chopped
Water and ice added
Start the machine
Ice cream placed on cup
chocolate as toppings
Serve Ice Cream
PS C:\Users\Dell\OneDrive\Desktop\wt_lab>
```

14

```
PS C:\Users\Dell\OneDrive\Desktop\wt_lab> node exp14.js

Sorry! Book with libraryId NaN is not available.
PS C:\Users\Dell\OneDrive\Desktop\wt_lab> node exp14.js
5
Book with libraryId 5 : 'The Old Man and the Sea' by 'Ernest Hemingway' is available.
PS C:\Users\Dell\OneDrive\Desktop\wt_lab>
```

19

```
PS C:\Users\Dell\OneDrive\Desktop\wt_lab> node exp19.js
Order placed.
Food prepared.
Food packaged.
Food delivered.
Food delivery process complete.
```

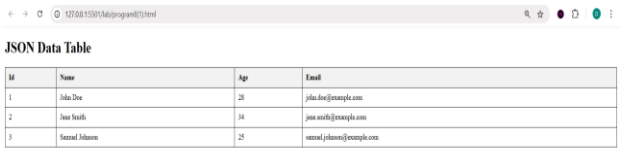
7a

```
PS C:\Users\Dell\OneDrive\Desktop\wt_lab> node exp7.js
User user1 registered successfully.
User user2 registered successfully.
User user1 deleted successfully.
Password for user2 updated successfully.
Authentication successful.
Registered users:
user2
PS C:\Users\Dell\OneDrive\Desktop\wt_lab>
```

7b

```
PS C:\Users\Dell\OneDrive\Desktop\wt_lab> node exp7b.js
Original array: [
  64, 34, 25, 12,
  22, 11, 90
]
Sorted array: [
  11, 12, 22, 25,
  34, 64, 90
]
PS C:\Users\Dell\OneDrive\Desktop\wt_lab>
```

8a



JSON Data Table

Id	Name	Age	Email
1	John Doe	28	john.doe@example.com
2	Jane Smith	34	jane.smith@example.com
3	Samuel Johnson	25	samuel.johnson@example.com

8b

```
PS C:\Users\Dell\OneDrive\Desktop\wt_lab> node exp8a.js
User user1 registered successfully.
User user2 registered successfully.
User user3 registered successfully.
User user4 registered successfully.
Confirmation email sent to user1.
Process completed for user1
Confirmation email sent to user2.
Process completed for user2
Confirmation email sent to user3.
Process completed for user3
Confirmation email sent to user4.
Process completed for user4
All users processed.
PS C:\Users\Dell\OneDrive\Desktop\wt_lab>
```

9a

```
PS C:\Users\Dell\OneDrive\Desktop\wt_lab> node exp9a.js
Enter the port number to run the server: 5000
Server is running at http://localhost:5000/
```

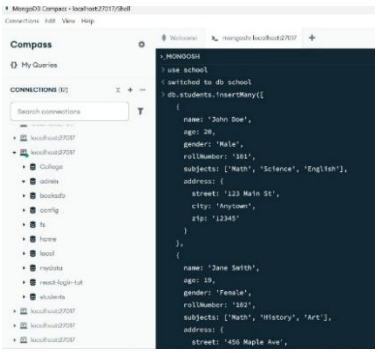
9b

```
PS C:\Users\Dell\OneDrive\Desktop\wt_lab> node exp9b.js
File Content:
Meena,23,WaterDept,2300
Sheila,29,Accounts,40000
Jim,45,Accounts,80000
Naina,78,Water,90000
Kiran,56,Housekeeping,67000

PS C:\Users\Dell\OneDrive\Desktop\wt_lab>
```

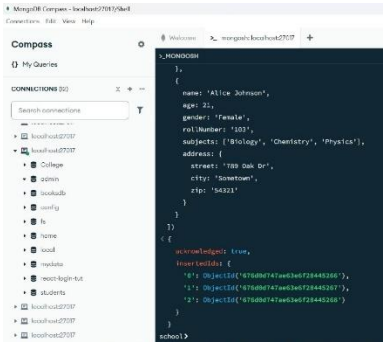
9c

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\Dell\OneDrive\Desktop\wt_lab> node exp9c.js
Enter the file name: sample
Enter the text to add: These are employee details
File created and text added successfully.
PS C:\Users\Dell\OneDrive\Desktop\wt_lab>
```

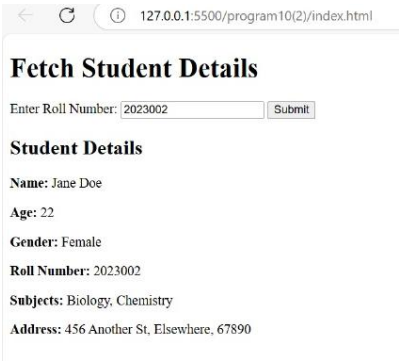
10a



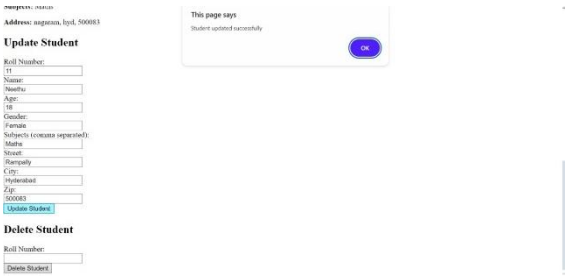
10a



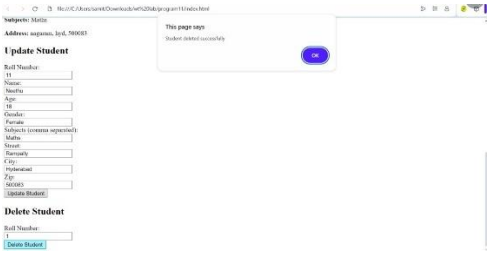
10b



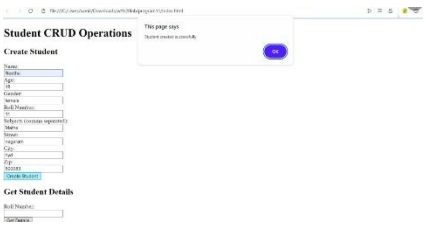
11



11

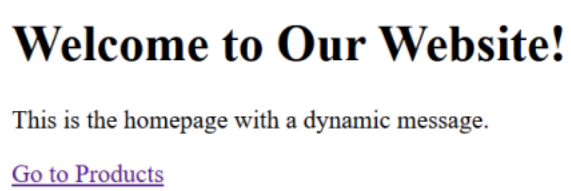


11



11

20

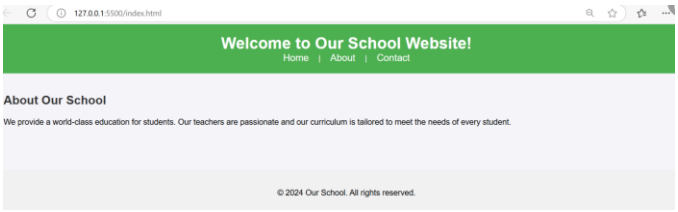

Welcome to Our Website!
This is the homepage with a dynamic message.
Go to Products

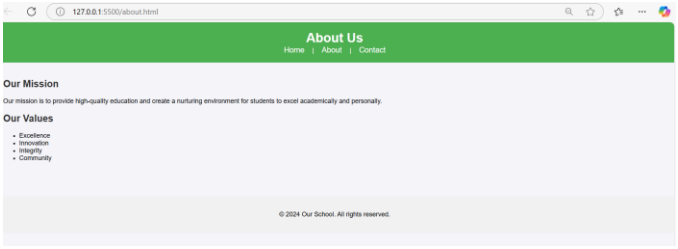

localhost:3000/products
Our Products
Product 1: Amazing Widget
Product 2: Super Gadget
Product 3: Cool Gizmo
Back to Home

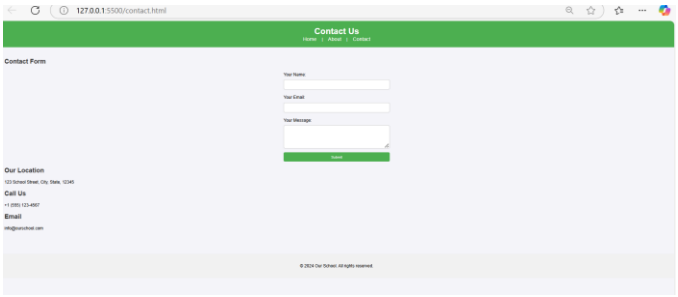
21


localhost:3000
Welcome to KMIT!

22


127.0.0.1:5500/index.html
Welcome to Our School Website!
Home | About | Contact
About Our School
We provide a world-class education for students. Our teachers are passionate and our curriculum is tailored to meet the needs of every student.
© 2024 Our School. All rights reserved.


127.0.0.1:5500/about.html
About Us
Home | About | Contact
Our Mission
Our mission is to provide high-quality education and create a nurturing environment for students to excel academically and personally.
Our Values
Excellence
Innovation
Integrity
Community
© 2024 Our School. All rights reserved.


127.0.0.1:5500/contact.html
Contact Us
Home | About | Contact
Contact Form
Your Name:
Your Email:
Your Message:
Our Location
Call Us
Email