SRI RAMAKRISHNA ENGINEERING COLLEGE

[Educational Service : SNR Sons Charitable Trust]
[Autonomous Institution, Reaccredited by NAAC with 'A+' Grade]
[Approved by AICTE and Permanently Affiliated to Anna University, Chennai]
[ISO 9001:2015 Certified and all eligible programmes Accredited by NBA]
VATTAMALAIPALAYAM, N.G.G.O. COLONY POST, COIMBATORE- 641 022.

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

Class: IV B.Tech AI & DS **Semester: VII**

Certified that this is the bonafide record of work done ________DHANUSH KUMAR J (71812111065)______________________in the **20AD2E32 – FULL STACK DEVELOPMENT (Theory cum Laboratory)** of this institution for VII semester during the academic year 2024-2025.

Faculty In-charge	**Head of the Department**
Mrs. P.V. Kavitha	***Dr. V. Karpagam***
AP (Sl.Gr) / AI&DS	*Professor & Head / AI&DS*

Roll No:

Submitted for the Autonomous Practical Examination held on _______________

Internal Examiner **Subject Expert**

DEPARTMENT OF

ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

VISION

To achieve excellence in the domain of Artificial Intelligence and Data Science and produce globally competent professionals to solve futuristic societal challenges and industrial needs.

MISSION

- To actively engage in the implementation of innovative intelligent solutions for interdisciplinary Artificial Intelligence based applications with ethical standards
- To promote research, innovation and entrepreneurial skills through industry and academic collaboration.

PROGRAMME EDUCATIONAL OBJECTIVES (PEOs)

PEO 1 - Exhibit proficiency in their career, higher studies and research with strong foundations in Mathematics, Computing, Artificial Intelligence and Data Science.

PEO 2 - Apply Artificial Intelligence and Data Science knowledge and skills to develop innovative solutions for multi-disciplinary problems, adhering to ethical standards.

PEO 3 - Engage in constructive research, professional development and life-long learning with skills in emerging technologies.

PROGRAMME OUTCOMES (POs)

1. **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

4. **Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, andsynthesis of the information to provide valid conclusions.
5. **Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilitiesrelevant to the professional engineering practice.
7. **Environment and sustainability**: Understand the impact of the professional engineeringsolutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics**: Apply ethical principles and commit to professional ethics and responsibilities andnorms of the engineering practice.
9. **Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend andwrite effective reports and design documentation, make effective presentations, and giveand receive clear instructions.
11. **Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning**: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAMME SPECIFIC OUTCOMES (PSOs)

PSO 1 - Analyze, design and build sustainable intelligent solutions to solve challenges imposed by industry and society.

PSO2 - Demonstrate data analysis skills to achieve effective insights and decision making to solve real-life problems.

PSO3 - Apply mathematical and statistical models to solve the computational tasks, and model real-world problems using appropriate AI / ML algorithms.

INDEX

Ex No : 1 **Date : 07/08/2024**	**WEBSITE CREATION USING HTML**

AIM:

To create a portfolio website using HTML.

ALGORITHM:

STEP 1: Open any HTML editor.

STEP 2: Insert html, head, and title tags. Specify the title for the webpage inside the title tag (e.g., "My Portfolio").

STEP 3: Open the body tag to display all your details.

STEP 4: Create a navigation bar inside the header tag, including links to different sections like Home, Profile, Career, Projects, and Contact.

STEP 5: Create a Home section that introduces yourself with a welcoming message and a call to action to explore your projects.

STEP 6: Create a Profile section that includes your photo, a brief introduction about yourself, and your personal details like date of birth, father's name, address, email, and phone number.

STEP 7: Create a Career section that displays your educational background in a table format.

STEP 8: Create a Projects section that lists your recent projects, each with a title, description, and image.

STEP 9: Create a Contact section that provides your contact information and links to your social media profiles.

STEP 10: Add a footer section to display copyright information.

STEP 11: Close all the opened tags and save the file with an .html extension.

STEP 12: Open the file using any browser to view your portfolio website.

SOURCE CODE:

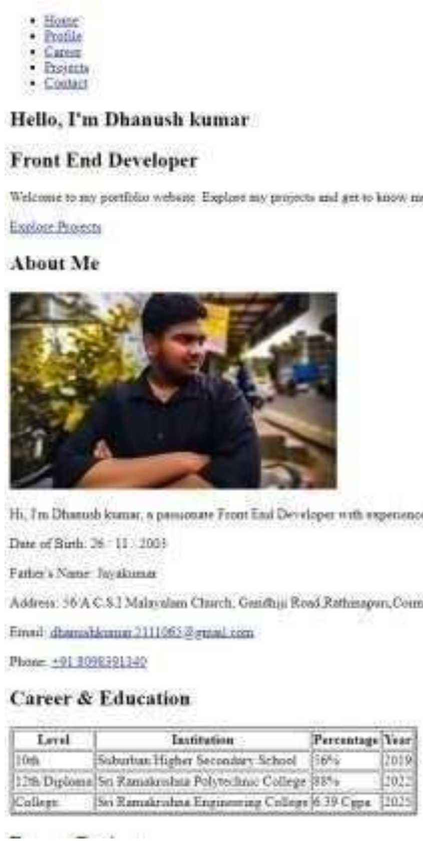
portfolio.html:

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

```
    <title>My Portfolio</title>
    <link rel="stylesheet" href="no1.css">
</head>
<body>
    <!-- Header -->
    <header>
        <nav>
            <ul>
                <li><a href="#home">Home</a></li>
                <li><a href="#profile">Profile</a></li>
                <li><a href="#career">Career</a></li>
                <li><a href="#projects">Projects</a></li>
                <li><a href="#contact">Contact</a></li>
            </ul>
        </nav>
    </header>
    <!-- Hero Section -->
    <section id="home" class="hero-section">
        <h1>Hello, I'm <span>Dhanush kumar</span></h1>
        <h2>Front End Developer</h2>
        <p>Welcome to my portfolio website. Explore my projects and get to know me better!</p>
        <a href="#projects" class="button">Explore Projects</a>
    </section>
    <!-- Profile Section -->
    <section id="profile" class="section">
        <div class="profile-container">
            <h2>About Me</h2>
            <img src="profile.jpg" alt="Profile Photo" class="profile-photo" >
            <p>Hi, I'm Dhanush kumar, a passionate Front End Developer with experience in creating
beautiful and functional websites.</p>
            <p>Date of Birth: 26 / 11 / 2003</p>
            <p>Father's Name: Jayakumar</p>
            <p>Address: 56/A C.S.I Malayalam Church, Gandhiji Road,Rathinapuri,Coimbatore-
641027</p>
            <p>Email: <a
href="mailto:your.email@example.com">dhanushkumar.2111065@gmail.com</a></p>
```

```
</div>
    </section>
    <!-- Career Section -->
    <section id="career" class="section">
      <h2>Career & Education</h2>
      <table >
        <thead><tr>
            <th>Level</th>
            <th>Institution</th>
            <th>Percentage</th>
            <th>Year</th>
          </tr>
        </thead>
        <tbody>
          <tr>
            <td>10th</td>
            <td>Suburban Higher Secondary School</td>
            <td>56%</td>
            <td>2019</td>
          </tr>
          <tr>
            <td>12th/Diploma</td>
            <td>Sri Ramakrishna Polytechnic College</td>
            <td>88%</td>
            <td>2022</td>
          </tr>
          <tr>
            <td>College</td>
            <td>Sri Ramakrishna Engineering College</td>
            <td>6.39 Cgpa</td>
            <td>2025</td>
          </tr>
        </tbody>
      </table>
    </section>
    <!-- Projects Section -->
    <section id="projects" class="projects-container">
        <h2>Recent Projects</h2>
        <div class="project-item">
            <h3>Project 1</h3>
            <p>Symptoms Based Disease Diagnosing </p>
            <img src="project1.png" alt="Project 1" >
        </div>
        <div class="project-item">
            <h3>Project 2</h3>
            <p>Network Based Facial Attendance System</p>
            <img src="project2.png" alt="Project 2" >
        </div>
        <div class="project-item">
            <p></p>
            <h3>DERMATOLOGICAL DISEASE
                DETECTION USING DEEP LEARNING</h3>
            <img src="project3.png" alt="Project 3" >
        </div>
        <div class="project-item">
            <h3>Project 4</h3>
            <p></p>
            <img src="project4.png" alt="Project 4" >
        </div>
    </section>
    <!-- Contact Section -->
    <section id="contact" class="contact-section">
        <h2>Contact Information</h2>
        <p>Email: <a
href="mailto:your.email@example.com">dhanushkumar.2111065@srec.ac.in</a></p>
        <p>Phone: <a href="tel:+1234567890">+91 8098391340</a></p>
        <p>Address: Tamil Nadu - Coimbatore</p>
        <p>Connect with me:</p>
        <ul>
            <li><a href="https://linkedin.com" target="_blank">LinkedIn</a></li>
            <li><a href="https://github.com" target="_blank">GitHub</a></li>
            <li><a href="https://twitter.com" target="_blank">Twitter</a></li>
        </ul>
    </section>
    <!-- Footer -->
    <footer>
        <p>&copy; 2024 Dhanush kumar. All rights reserved.</p>
    </footer>
</body>
</html>
```

OUTPUT:



- Home
- Profile
- Career
- Projects
- Contact

Hello, I'm Dhanush kumar

Front End Developer

Welcome to my portfolio website. Explore my projects and get to know me better!

Explore Projects

About Me

Hi, I'm Dhanush kumar, a passionate Front End Developer with experience in creating beautiful and functional websites.

Date of Birth: 26 - 11 - 2003

Father's Name: Jayakumar

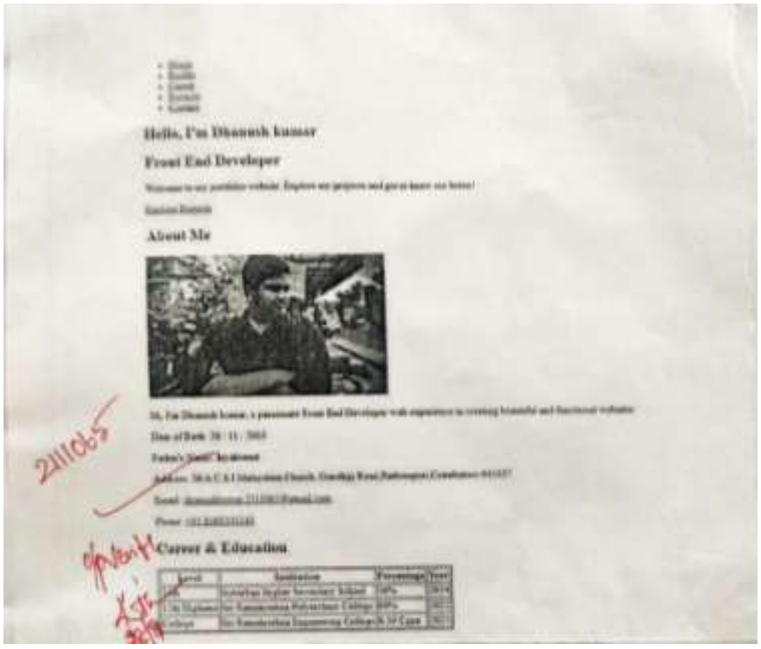
Address: 56/A C.S.I Malayalam Church, Gandhiji Road,Rathinapuri,Coimbatore-641027

Email: dhanushkumar.2111065@gmail.com

Phone: +91 8098391340

Career & Education

Level	Institution	Percentage	Year
10th	Suburban Higher Secondary School	56%	2019
12th Diploma	Sri Ramakrishna Polytechnic College	88%	2022
College	Sri Ramakrishna Engineering College	6.39 Cgpa	2025


- [illegible]
- [illegible]
- [illegible]
- [illegible]
- [illegible]

Hello, I'm Dhanush kumar

Front End Developer

[illegible]

About Me

[illegible]

Date of Birth: [illegible]

Father's Name: [illegible]

Address: [illegible]

Email: [illegible]

Phone: [illegible]

Career & Education

Level	Institution	Percentage	Year
[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]

2111065

Designed a portfolio website using HTML to showcase personal details, career background, and recent projects in a structured format. Integrated sections for Profile, Career, Projects, and Contact, along with a navigation bar for smooth user experience.

RESULT:

Thus, the website development using HTML for booking car service is implemented successfully.

Ex No : 2 **Date: 14/08/2024**	**WEBSITE CREATION USING HTML & CSS**

AIM:

To develop a website for portfolio profile using HTML & CSS.

ALGORITHM:

STEP 1: Create a new folder and create HTML files for Portfolio profile page.

STEP 2: Create a CSS file for styling the document.

STEP 3: In the Portfolio .html, create the link for the CSS file.

STEP 4: Add header, footer, navigation bar, image and description using necessary tags.

STEP 5: In the Portfolio.html, create the link for the CSS file.

STEP 6: Add header, footer, navigation bar, and links for places description using necessary tags.

STEP 7: Go to the CSS style sheet and apply necessary styles for the html elements.

STEP 8: Close the tags appropriately in HTML files.

STEP 9: Save the files and open using any browser to display the output.

SOURCE CODE:

Portfolio.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>My Portfolio</title>
    <link rel="stylesheet" href="no1.css">
</head>
<body>
    <!-- Header -->
    <header>
        <nav>
            <ul>
                <li><a href="#home">Home</a></li>
                <li><a href="#profile">Profile</a></li>
                <li><a href="#career">Career</a></li>
                <li><a href="#projects">Projects</a></li>
            <li><a href="#contact">Contact</a></li>
          </ul>
        </nav>
      </header>
      <!-- Hero Section -->
      <section id="home" class="hero-section">
        <h1>Hello, I'm <span>Dhanush kumar</span></h1>
        <h2>Front End Developer</h2>
        <p>Welcome to my portfolio website. Explore my projects and get to know me
better!</p>
        <a href="#projects" class="button">Explore Projects</a>
      </section>

      <!-- Profile Section -->
      <section id="profile" class="section">
        <div class="profile-container">
          <h2>About Me</h2>
          <img src="profile.jpg" alt="Profile Photo" class="profile-photo" >
          <p>Hi, I'm Dhanush kumar, a passionate Front End Developer with experience in
creating beautiful and functional websites.</p>
          <p>Date of Birth: 26 / 11 / 2003</p>
          <p>Father's Name: Jayakumar</p>
          <p>Address: 56/A C.S.I Malayalam Church, Gandhiji Road,Rathinapuri,Coimbatore-
641027</p>
          <p>Email: <a
href="mailto:your.email@example.com">dhanushkumar.2111065@gmail.com</a></p>
          <p>Phone: <a href="tel:+1234567890">+91 8098391340</a></p>
        </div>
      </section>

      <!-- Career Section -->
      <section id="career" class="section">
        <h2>Career & Education</h2>
        <table >
          <thead>
            <tr>
              <th>Level</th>
              <th>Institution</th>
              <th>Percentage</th>
              <th>Year</th>
            </tr>
          </thead>
          <tbody>
```

```
            <tr>
                <td>10th</td>
                <td>Suburban Higher Secondary School</td>
                <td>56%</td>
                <td>2019</td>
            </tr>
            <tr>
                <td>12th/Diploma</td>
                <td>Sri Ramakrishna Polytechnic College</td>
                <td>88%</td>
                <td>2022</td>
            </tr>
            <tr>
                <td>College</td>
                <td>Sri Ramakrishna Engineering College</td>
                <td>6.39 Cgpa</td>
                <td>2025</td>
            </tr>
        </tbody>
    </table>
</section>

<!-- Projects Section -->
<section id="projects" class="projects-container">
    <h2>Recent Projects</h2>
    <div class="project-item">
        <h3>Project 1</h3>
        <p>Symptoms Based Disease Diagnosing </p>
        <img src="project1.png" alt="Project 1" >
    </div>
    <div class="project-item">
        <h3>Project 2</h3>
        <p>Network Based Facial Attendance System</p>
        <img src="project2.png" alt="Project 2" >
    </div>
    <div class="project-item">
        <p></p>
        <h3>DERMATOLOGICAL DISEASE
            DETECTION USING DEEP LEARNING</h3>
        <img src="project3.png" alt="Project 3" >
    </div>
    <div class="project-item">
        <h3>Project 4</h3>
```

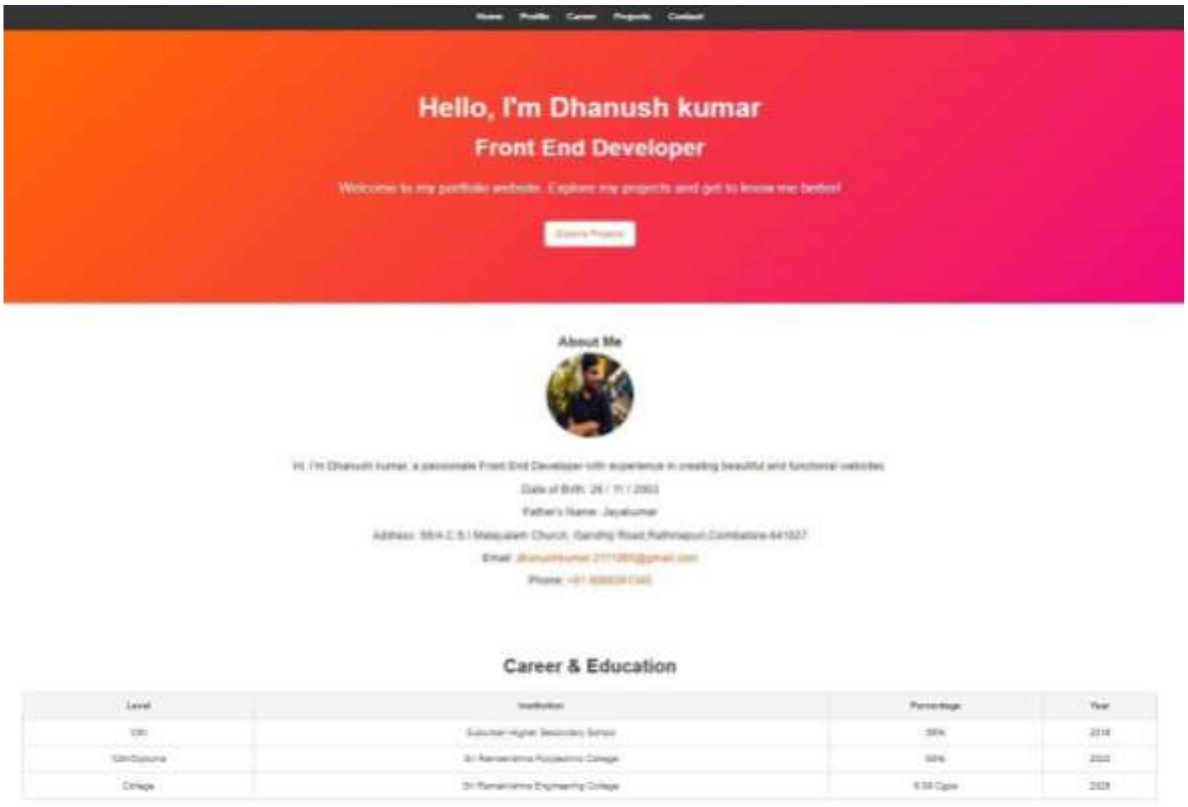
```
            <p></p>
            <img src="project4.png" alt="Project 4" >
        </div>
    </section>

    <!-- Contact Section -->
    <section id="contact" class="contact-section">
        <h2>Contact Information</h2>
        <p>Email: <a
href="mailto:your.email@example.com">dhanushkumar.2111065@srec.ac.in</a></p>
        <p>Phone: <a href="tel:+1234567890">+91 8098391340</a></p>
        <p>Address: Tamil Nadu - Coimbatore</p>
        <p>Connect with me:</p>
        <ul>
            <li><a href="https://linkedin.com" target="_blank">LinkedIn</a></li>
            <li><a href="https://github.com" target="_blank">GitHub</a></li>
            <li><a href="https://twitter.com" target="_blank">Twitter</a></li>
        </ul>
    </section>

    <!-- Footer -->
    <footer>
        <p>&copy; 2024 Dhanush kumar. All rights reserved.</p>
    </footer>
</body>

</html>
```

Portfolio.css

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

body {
    font-family: 'Arial', sans-serif;
    background-color: #f4f4f4;
    color: #333;
    line-height: 1.6;
}

header {
    background-color: #333;
    padding: 10px 0;
    text-align: center;
}

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

nav ul li {
    margin: 0 15px;
}

nav ul li a {
    color: #fff;
    text-decoration: none;
    font-weight: bold;
}

/* Hero Section */
.hero-section {
    background: linear-gradient(135deg, #ff6a00, #ee0979);
    color: white;
    padding: 100px 0;
    text-align: center;
}

.hero-section h1 {
    font-size: 3rem;
}

.hero-section h2 {
    font-size: 2.5rem;
}

.hero-section h2 span {
    color: #ffeb3b;
}

.hero-section p {
    font-size: 1.5rem;
    margin: 20px 0;
}

.button {
    background-color: #fff;
    color: #ff6a00;
    padding: 10px 20px;
    border-radius: 5px;
    text-decoration: none;
    margin-top: 20px;
    display: inline-block;
}

.button:hover {
    background-color: #ff6a00;
    color: #fff;
}

/* Profile Section */
.profile-container {
    padding: 50px;
    background-color: #fff;
    text-align: center;
}

.profile-container img.profile-photo {
    width: 150px;
    height: 150px;
    border-radius: 50%;
    margin-bottom: 20px;
}

.profile-container p {
    font-size: 1.2rem;
    margin: 10px 0;
}

.profile-container a {
    color: #ff6a00;
    text-decoration: none;
}
.profile-container a:hover {
    text-decoration: underline;
```

```
}

/* Career Section */
#career {
    padding: 50px;
    background-color: #fff;
}

#career h2 {
    text-align: center;
    font-size: 2rem;
    margin-bottom: 20px;
}

table {
    width: 100%;
    border-collapse: collapse;
    margin: 0 auto;
}

table th, table td {
    padding: 10px;
    border: 1px solid #ddd;
    text-align: center;
}

table th {
    background-color: #f4f4f4;
}

/* Projects Section */
.projects-container {
    padding: 50px;
    background-color: #fff;
}

.projects-container h2 {
    text-align: center;
    font-size: 2rem;
    margin-bottom: 20px;
}

.project-item {
    background-color: #fff;
    padding: 20px;
    border-radius: 8px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
    text-align: center;
}
.project-item img {
    width: 35%;
    height: 35%;
    border-radius: 8px;
}
/* Contact Section */
.contact-section {
    background-color: #333;
    color: white;
    padding: 50px;
    text-align: center;
}
.contact-section p {
    margin: 10px 0;
    font-size: 1.2rem;
}

.contact-section a {
    color: #ff6a00;
    text-decoration: none;
}

.contact-section a:hover {
    text-decoration: underline;
}

.contact-section ul {
    list-style: none;
    padding: 0;
}

.contact-section ul li {
    margin: 10px 0;
}

.contact-section ul li a {
    color: #ff6a00;
    text-decoration: none;
}

.contact-section ul li a:hover {
    text-decoration: underline;
}

/* Footer */
footer {
    background-color: #222;
    color: #fff;
    text-align: center;
    padding: 15px 0;
```

OUTPUT:



Hello, I'm Dhanush kumar

Front End Developer

Welcome to my portfolio website. Explore my projects and get to know me better!

About Me

Hi, I'm Dhanush kumar, a passionate Front End Developer with experience in creating beautiful and functional websites.

Date of Birth: 26 / 11 / 2003

Father's Name: Jayakumar

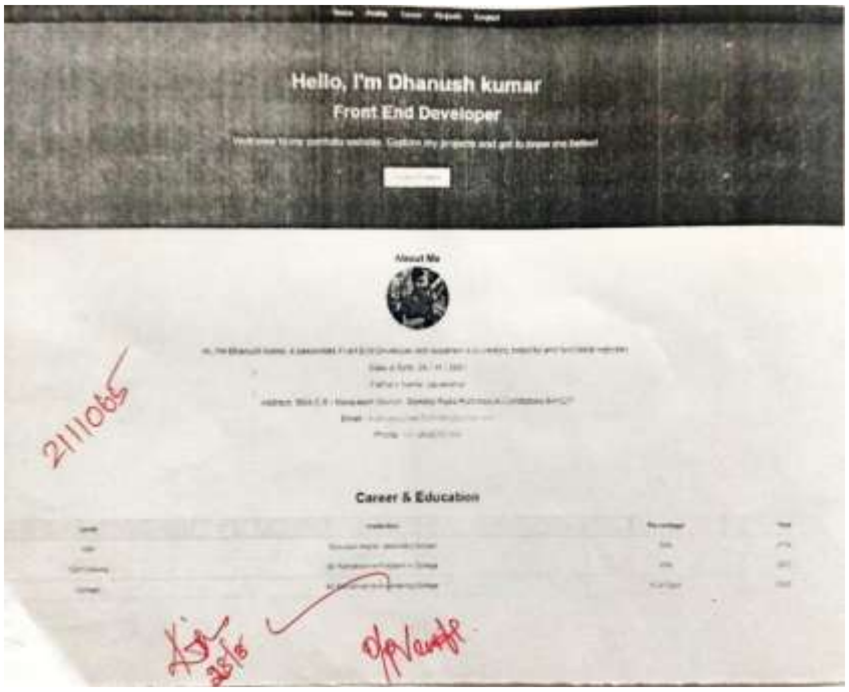
Address: 56/4 C.S.I Malayalam Church, Gandhiji Road,Rathinapuri,Coimbatore-641027

Email: [illegible]@gmail.com

Phone: +91 [illegible]

Career & Education

Level	Institution	Percentage	Year
10th	Suburban Higher Secondary School	[illegible]	2018
12th/Diploma	Sri Ramakrishna Polytechnic College	[illegible]	2022
College	Sri Ramakrishna Engineering College	[illegible] Cgpa	2025



Hello, I'm Dhanush kumar

Front End Developer

[illegible]

About Me

[illegible]

Career & Education

[illegible]

2111065

[illegible] 25/6 [illegible]

Developed a portfolio website using HTML and CSS to create a visually appealing profile with sections for personal details, projects, and contact information. Styled the layout using CSS to enhance navigation, headers, footers, and image displays for a cohesive user experience.

Result:

Thus the website for portfolio profile has been developed using HTML and CSS successfully.

Ex No : 3 **Date: 21/08/2024**	**PALINDROME OF A NUMBER USING JAVASCRIPT**

AIM:

To write a JavaScript program to check whether the given string is palindrome or not.

ALGORITHM:

STEP 1: Open the Visual Studio code.

STEP 2: Give the title as Palindrome.

STEP 3: Get the strings or number from the user.

STEP 4: Take the temporary variable that holds a number.

STEP 5: Reverse the given string.

STEP 6: Compare the original numbers with the reversed number.

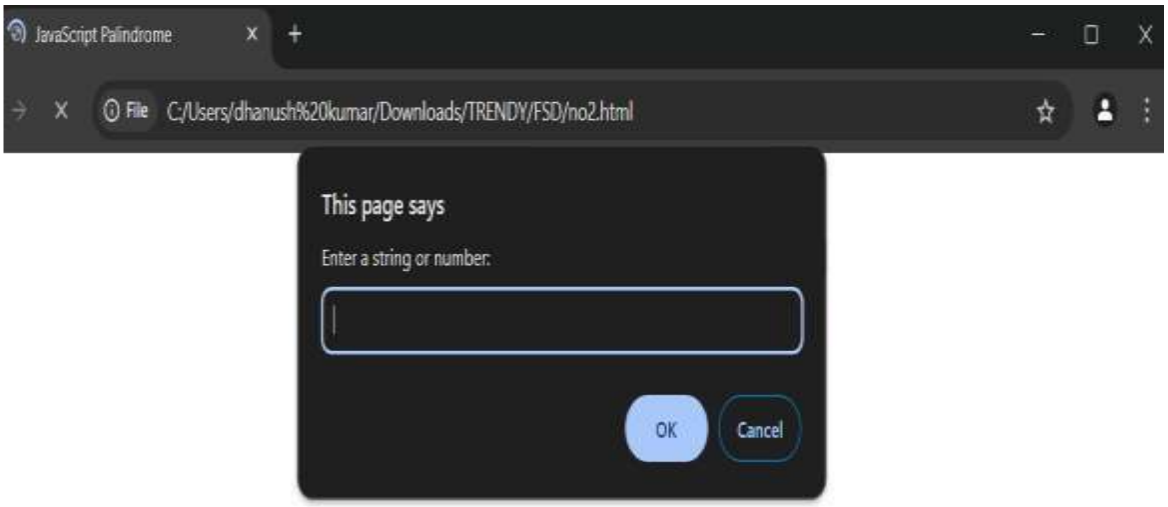
STEP 7: If the temporary and original number are the same, the number or string is apalindrome and else the given string is not a palindrome.

STEP 8: Save and compile the program.

SOURCE CODE:

```
    <html>
    <head> <title> JavaScript Palindrome </title>
    </head>
    <body>
    <script>
function validatePalin(str) { const
   len = string.length;
   for (let i = 0; i < len / 2; i++) {
          if (string[i] !== string[len - 1 - i]) {
             alert( 'It is not a palindrome');
          }}
   alert( 'It is a palindrome');}
    const string = prompt('Enter a string or number: ');
    const value = validatePalin(string);
    console.log(value);
   </script></body> <html>
```

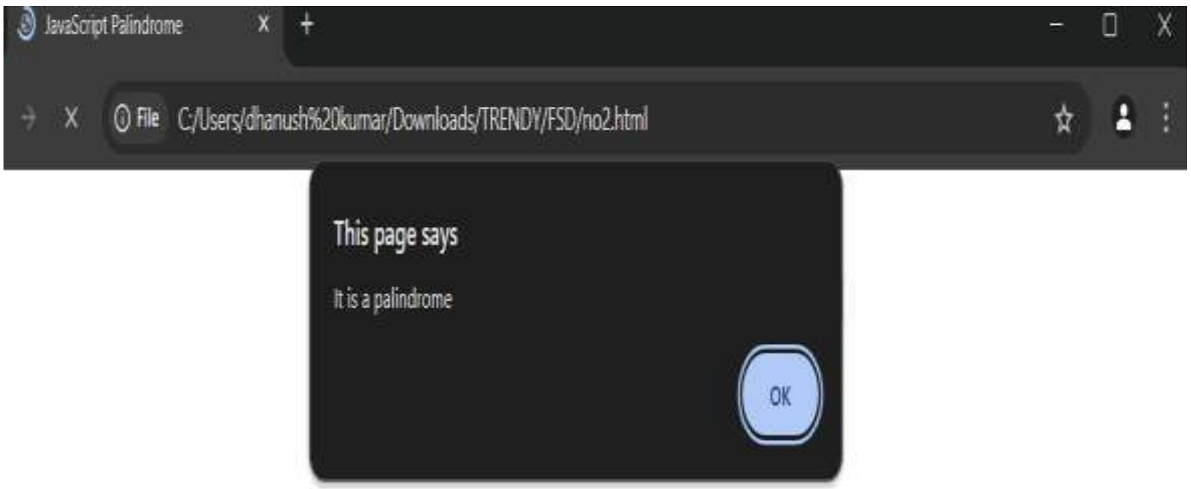
OUTPUT:


JavaScript Palindrome

C:/Users/dhanush%20kumar/Downloads/TRENDY/FSD/no2.html

This page says

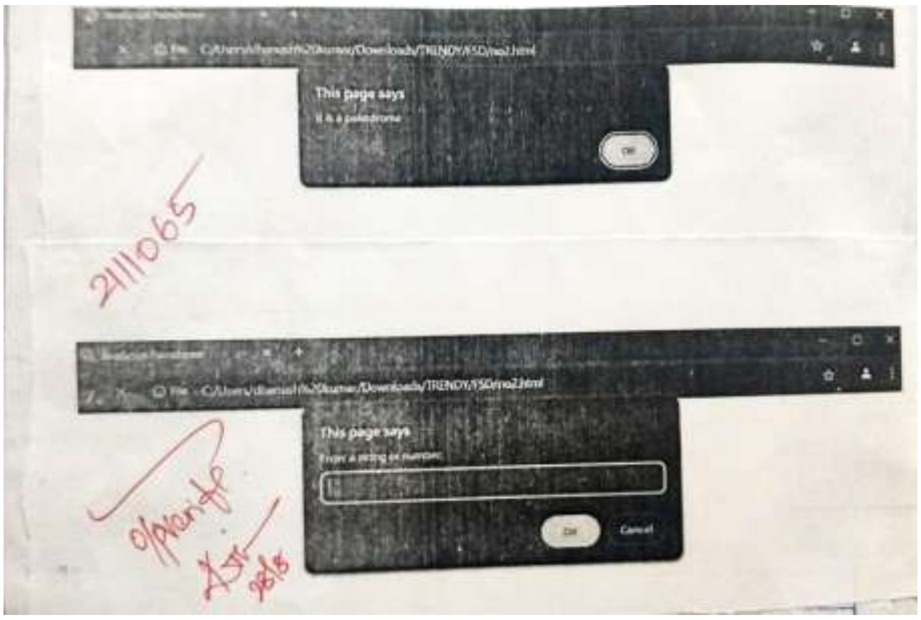
Enter a string or number:


JavaScript Palindrome

C:/Users/dhanush%20kumar/Downloads/TRENDY/FSD/no2.html

This page says

It is a palindrome



This page says

It is a palindrome

OK

This page says

Enter a string or number:

OK Cancel

2111065

o/p verified

[illegible]

Created a JavaScript program to determine if a given string or number is a palindrome by reversing the input and comparing it to the original. Implemented user input handling and logic to display whether the input is a palindrome or not.

RESULT:

Thus the JavaScript program to check whether the given string is palindrome or not is executed successfully.

Ex No : 4 **Date : 14/08/2024**	**WEB APPLICATION USING JAVASCRIPT**

AIM:

To create a International workshop registration web page using Javascript.

ALGORITHM:

STEP 1: Write a JavaScript program inside the <script> tag.
STEP 2: Use document.getElementById to get the value of the input fields (name, email, phone, etc.).
STEP 3: If any of the required fields are empty, display an alert asking to fill the text fields.
STEP 4: Use a regular expression to validate the email format. If incorrect, display an alert asking for a valid email.
STEP 5: If all fields are correctly filled, display a modal box indicating successful registration.
STEP 6: Apply CSS styles for a visually appealing form and modal box.

SOURCE CODE:

Registration.html

```
 <!DOCTYPE html>
 <html lang="en">
 <head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>International Workshop Registration</title>
<link rel="stylesheet" href="styles.css">
    <link href="https://fonts.googleapis.com/css2?family=Poppins:wght@300;400;600&display=swap"
 rel="stylesheet">
 </head>
 <body>
<header>
        <div class="hero">
            <h1>International Workshop</h1>
            <p>Join experts and enthusiasts from around the world for insightful sessions and hands-on
 workshops.</p>
            <a href="#registrationForm" class="cta-button">Register Now</a>
        </div>
    </header>
<section class="registration-section">
        <div class="container">
            <h2>Register for the Workshop</h2>
            <form id="registrationForm">
                <label for="name">Full Name:</label>
                <input type="text" id="name" name="name" required>
<label for="email">Email Address:</label>
```

```
            <input type="email" id="email" name="email" required>

            <label for="phone">Phone Number:</label>
            <input type="tel" id="phone" name="phone" required>

            <label for="country">Country:</label>
            <select id="Country" name="country" required>
                <option value="">-- Select --</option>
                <option value="India">India</option>
                <option value="America">America</option>
                <option value="China">China</option>
            </select>

            <label for="workshop">Select Workshop:</label>
            <select id="workshop" name="workshop" required>
                <option value="">-- Select --</option>
                <option value="AI Workshop">AI Workshop</option>
                <option value="Blockchain Seminar">Blockchain Seminar</option>
                <option value="Cybersecurity Bootcamp">Cybersecurity Bootcamp</option>
            </select>
            <button type="submit">Register</button>
        </form>
    </div>

    <h3>Registration Summary</h3>
    <div id="registrationDetails"></div>
</section>

<!-- Modal (Dialog Box) -->
<div id="successModal" class="modal">
    <div class="modal-content">
        <span class="close-btn">&times;</span>
        <h2>Registration Successful!</h2>
        <p>Your registration for the workshop has been successfully submitted. We will contact you
 via email.</p>
    </div>
</div>

<footer>
    <p>© 2024 International Workshop | All rights reserved</p>
</footer>

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

Styles.css

```
  /* General Styles */
  * {
 box-sizing: border-box; margin:
 0;
 padding: 0;
  }
  body {
 font-family: 'Poppins', sans-serif;
 background-color: #f4f4f9; color:
 #333;
 line-height: 1.6;
  }
.container { width:
 80%; margin:
 auto;
 max-width: 1200px;
  }
  header {
 background-image: url('https://source.unsplash.com/1600x900/?conference,workshop');
 background-size: cover;
 background-position: center;
 height: 100vh;
 display: flex;
 justify-content: center;
 align-items: center; color:
 white;
 text-align: center;}
  .hero h1 {
 font-size: 3rem; margin-
 bottom: 20px; font-
 weight: 600;
 color: rgb(255, 0, 0);}
  .hero p {
 font-size: 1.2rem;
 margin-bottom: 30px;
 font-weight: 300;
 color: rgb(0, 0, 0);}
  .cta-button {
 background-color: #ff6347;
 color: white;
 padding: 15px 30px;
 font-size: 1rem; border:
 none;  cursor: pointer;
 border-radius: 5px;
 text-decoration: none;
 transition: background-color 0.3s ease;
  }
.cta-button:hover { background-
 color: #e5533b;
```

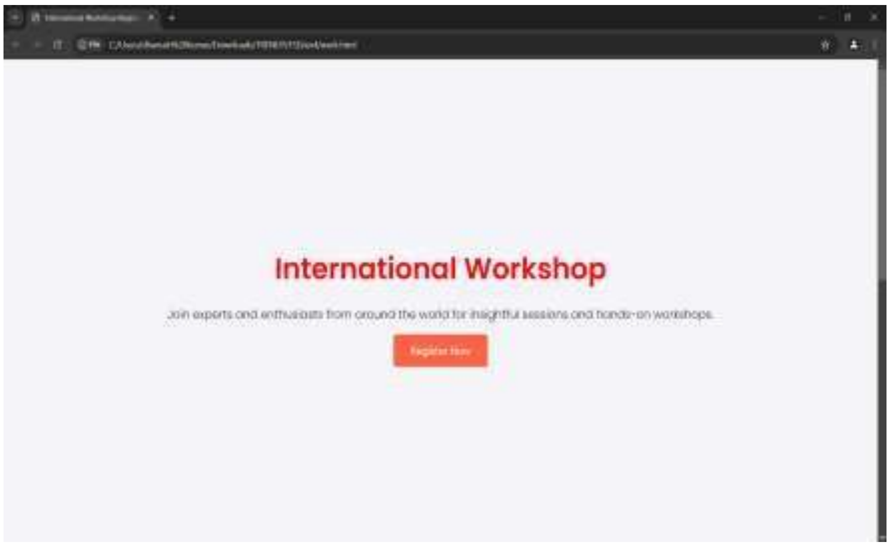
```
    }
.registration-section {  padding:
   50px 0; background-color:
   #f9f9f9;}
    form {
   display: flex;
   flex-direction: column;
   gap: 15px;
   background-color: white;
   padding: 30px;
   border-radius: 10px;
   box-shadow: 0 0 15px rgba(0, 0, 0, 0.1);
   max-width: 600px;
   margin: 20px auto;
    }label {
   font-size: 1rem;
   font-weight: 400;
    }
input, select {
   padding: 12px;
   font-size: 1rem;
   border: 1px solid #ddd;
   border-radius: 5px;
   outline: none;}
input:focus, select:focus {
   border-color: #ff6347;
    }
    button {
   padding: 15px;
   background-color: #ff6347;
   color: white;
   font-size: 1rem;
   border: none; cursor:
   pointer; border-radius:
   5px;
   transition: background-color 0.3s ease;
    }
    button:hover {
   background-color: #e5533b;
    }
    h3 {
   text-align: center;
   font-size: 2rem;
   margin-top: 30px;
   color: #333;}
#registrationDetails {
   background-color: #fff;
   padding: 20px;
   margin: 20px auto;
   max-width: 600px;
```

```
    border-radius: 10px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
    font-size: 1.1rem;
      }
      /* Modal Styles */
      .modal {
    display: none; /* Hidden by default */
    position: fixed;
    z-index: 1000; /* On top */ left:
    0;
    top: 0;
    width: 100%;
    height: 100%;
    background-color: rgba(0, 0, 0, 0.5); /* Black background with opacity */
    justify-content: center;
    align-items: center;
      }
.modal-content { background-
    color: white; padding: 20px;
    border-radius: 10px;
    width: 80%;
    max-width: 500px;
    text-align: center;
    box-shadow: 0 0 20px rgba(0, 0, 0, 0.2);
      }
.close-btn { float:
    right;
    font-size: 1.5rem;
    cursor: pointer; color:
    #888;
      }
.close-btn:hover { color:
    #333;
      }
.modal-content h2 { font-
    size: 2rem; margin-
    bottom: 10px;
      }
.modal-content p {
    font-size: 1.2rem;
    margin-top: 0;
      }
      /* Footer */
      footer {
    background-color: #333; color:
    white;
    text-align: center;
    padding: 20px 0;
    margin-top: 40px;
      }
```

Scripts.js

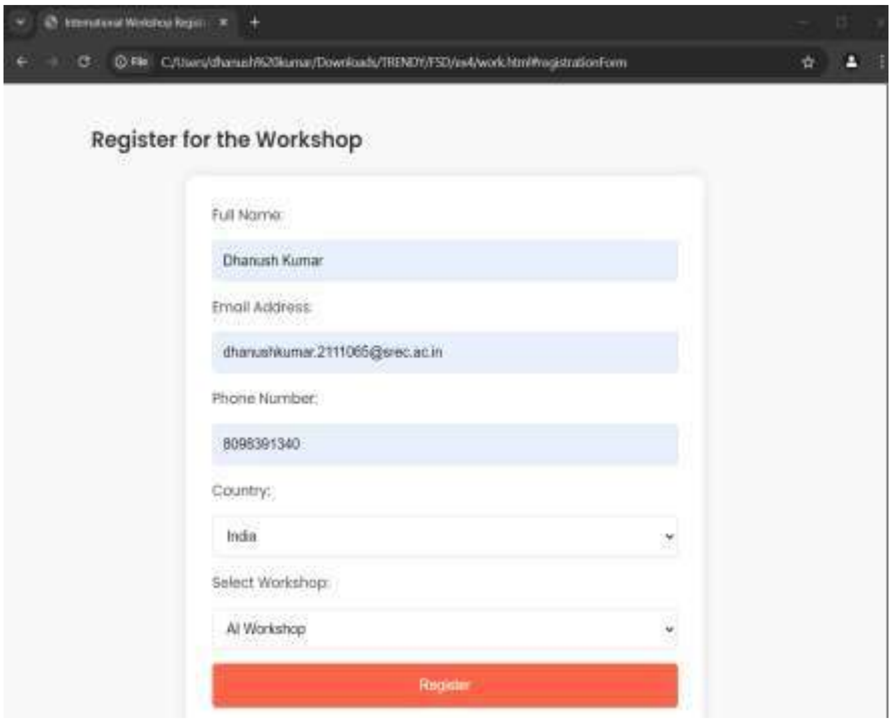
```
document.getElementById("registrationForm").addEventListener("submit", function(event) {
   event.preventDefault(); // Prevent form submission
   // Get form data
   const name = document.getElementById("name").value; const
   email = document.getElementById("email").value; const phone
   = document.getElementById("phone").value; const country =
   document.getElementById("country").value;
   const workshop = document.getElementById("workshop").value;
   // Check if all fields are filled
   if (name && email && phone && workshop) {
          // Save registration details to display
          const registrationDetails = `
             <p><strong>Name:</strong> ${name}</p>
             <p><strong>Email:</strong> ${email}</p>
             <p><strong>Phone:</strong> ${phone}</p>
             <p><strong>country:</strong> ${country}</p>
             <p><strong>Workshop:</strong> ${workshop}</p>
          `;
          // Display registration details
          document.getElementById("registrationDetails").innerHTML = registrationDetails;
          // Show the modal dialog
          const modal = document.getElementById("successModal");
          modal.style.display = "flex";
          // Close modal when 'X' button is clicked
          document.querySelector(".close-btn").addEventListener("click", function() {
             modal.style.display = "none";
          });
          // Close modal when clicked outside of it
          window.addEventListener("click", function(event) {
             if (event.target === modal) {
                modal.style.display = "none";}
           });
          // Clear the form
          document.getElementById("registrationForm").reset();
   } else {
          alert("Please fill out all the fields.");
        }
     });
```

OUTPUT:



International Workshop

Join experts and enthusiasts from around the world for insightful sessions and hands-on workshops.



International Workshop Regist
C:/Users/dhanush%20kumar/Downloads/TRENDY/FSD/ex4/work.html#registrationForm

Register for the Workshop

Full Name:

Dhanush Kumar

Email Address:

dhanushkumar.2111065@srec.ac.in

Phone Number:

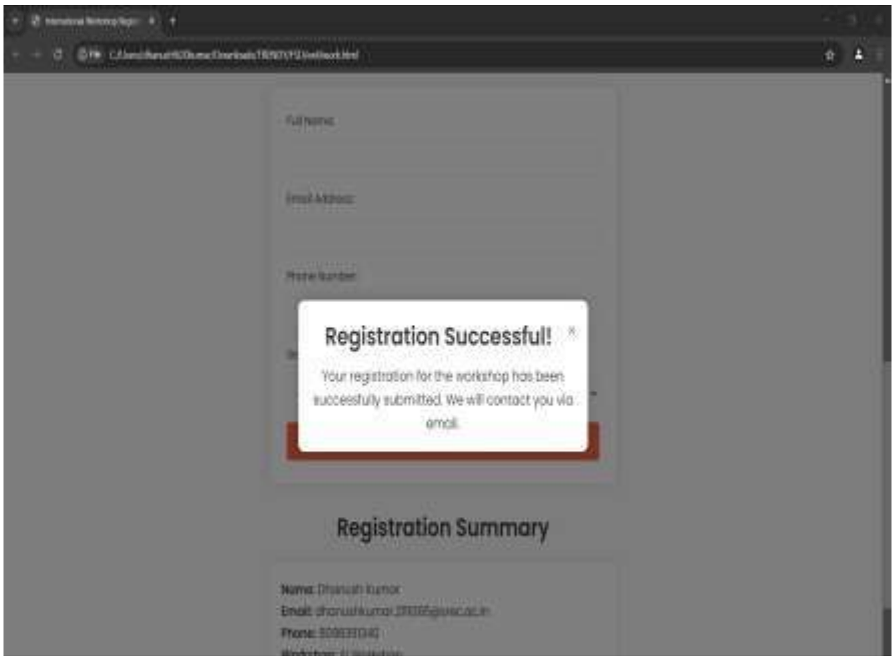
8098391340

Country:

India

Select Workshop:

AI Workshop

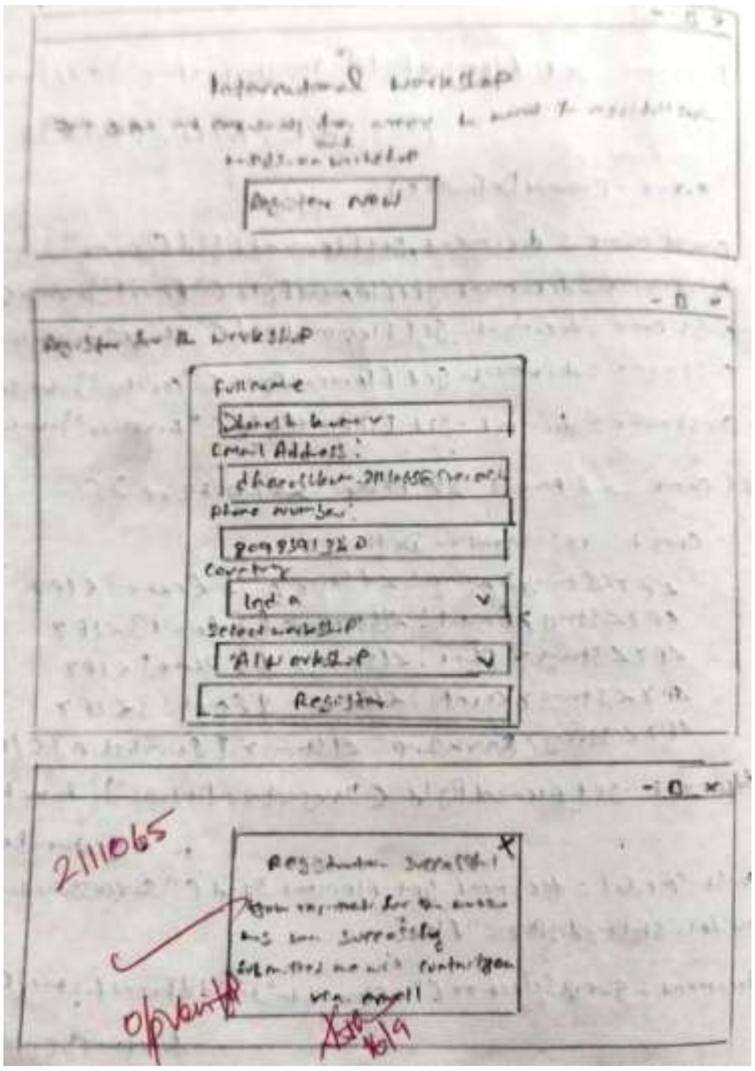


Registration Successful!

Your registration for the workshop has been successfully submitted. We will contact you via email.

Registration Summary

Name: Dhanush Kumar
Email: dhanushkumar.[illegible]@[illegible].ac.in
Phone: [illegible]



Developed a registration web page for an international workshop using JavaScript to validate user input for name, email, and phone number. Implemented form validation with alerts for empty fields and incorrect email formats, and displayed a modal box for successful registration.

RESULT:

Thus the International workshop registration webpage using JavaScript is executed successfully.

Ex No : 5 **Date: 28/08/2024**	**SIMPLE FRONT-END REACT JS APPLICATION**

AIM

To develop a simple front-end web application using ReactJS to add two numbers.

ALGORITHM

STEP 1: Open the terminal and create a new application in React.

STEP 2: Create a new React application using the command: npx create-react-app add-two-numbers-app.

STEP 3: Open the project in VS Code and open a new terminal window.

STEP 4: In the src folder, create a new file named AddNumbers.js.

STEP 5: In AddNumbers.js, create a simple component that includes two input fields for entering numbers and a button to calculate the sum.

STEP 6: Use the useState hook to handle input values and display the result when the button is clicked.

STEP 7: In the App.js file, import and render the AddNumbers component.

STEP 8: Save and run the command using npm start to start the development server.

STEP 9: Observe the output to ensure the application allows users to add two numbers and displays the result..

SOURCE CODE

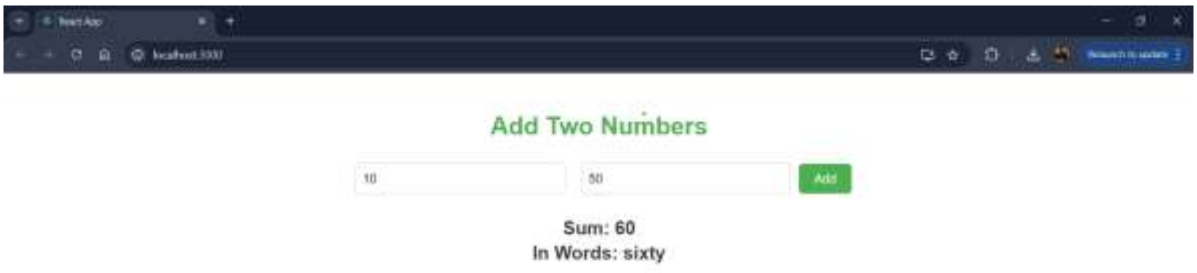
Numberadder.js:

```
import React, { useState } from 'react';
import './NumberAdder.css'; // Import CSS file

function NumberAdder() {
  const [num1, setNum1] = useState('');
  const [num2, setNum2] = useState('');
  const [sum, setSum] = useState(null);
  const [sumInWords, setSumInWords] = useState('');
  // Function to convert numbers to words
  const convertNumberToWords = (number) => {
    const words = [
      'zero', 'one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight', 'nine', 'ten',
      'eleven', 'twelve', 'thirteen', 'fourteen', 'fifteen', 'sixteen', 'seventeen', 'eighteen',
'nineteen',
      'twenty', 'thirty', 'forty', 'fifty', 'sixty', 'seventy', 'eighty', 'ninety'
    ];

    if (number < 21) return words[number];
    if (number < 100) return words[18 + Math.floor(number / 10)] + (number % 10 === 0
? '' : '-' + words[number % 10]);
    return 'Number too large to convert';
  };

  // Handler to calculate sum
  const handleAddition = () => {
    const sumValue = parseInt(num1) + parseInt(num2);
    setSum(sumValue);
    setSumInWords(convertNumberToWords(sumValue));
  };

  return (
    <div className="container">
      <h1>Add Two Numbers</h1>
      <input
        type="number"
        value={num1}
        onChange={(e) => setNum1(e.target.value)}
        placeholder="Enter first number"
        className="input-field"
      />
      <input
        type="number"
        value={num2}
        onChange={(e) => setNum2(e.target.value)}
        placeholder="Enter second number"
        className="input-field"
      />
      <button onClick={handleAddition} className="add-button">Add</button>

      {sum !== null && (
        <div className="result">
          <h2>Sum: {sum}</h2>
          <h2>In Words: {sumInWords}</h2>
        </div>
      )}
    </div>
  );
}

export default NumberAdder;
```

Index.js

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);

reportWebVitals();
```

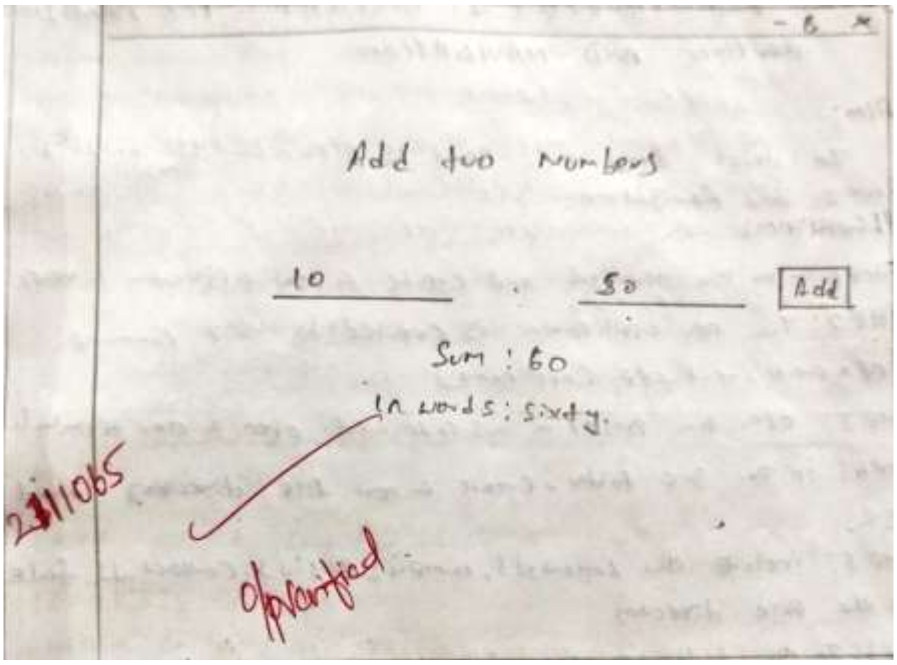
OUTPUT:



Add Two Numbers

10 50 Add

Sum: 60
In Words: sixty



Developed a ReactJS front-end application for adding two numbers, utilizing the useState hook for efficient state management. Created an interactive UI with input fields and a result display for real-time calculations.

RESULT:

Thus, the simple front-end web application using React.js is executed successfully.

Ex No : 6 **Date: 04/09/2024**	**FRONT-END REACT JS APPLICATION FOR HANDLING ROUTING AND NAVIGATION**

AIM

To develop blog articles front-end web application using ReactJS.

ALGORITHM

STEP 1: Open the terminal and create a new application in react.

STEP 2: The new application is created by using the command: npxcreate-react-app (app name)
.

STEP 3: Open the project in VS code and open a new terminalwindow.

STEP 4: In the terminal window,execute the command :npm i -Dreact-router-dom.

STEP 5: In the src folder, create a new file directory named pages.

STEP 6: Include the Layout.js,Home.js,Blog.js,Contact.js files in thepage directory.

STEP 7: In the index.js file include the above pages and implementpage routing.

STEP 8: Save and run the command using npm start.

STEP 9: Observe the output.

SOURCE CODE

index.js:

```
// src/index.js
import React from 'react';
import ReactDOM from 'react-dom/client';
import { BrowserRouter as Router, Route, Routes } from 'react-router-dom';
import Layout from './pages/Layout';
import Home from './pages/Home';
import Blog from './pages/Blog';
import Contact from './pages/Contact';


const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <Router>
    <Layout>
      <Routes>
```

```
            <Route path="/" element={<Home />} />
            <Route path="/blog" element={<Blog />} />
            <Route path="/contact" element={<Contact />} />
          </Routes>
        </Layout>
    </Router>
);
```

Layout.js:

```
    // src/pages/Layout.js
    import React from 'react';
    import { Link } from 'react-router-dom';
    import './Layout.css';

    const Layout = ({ children }) => {
      return (
        <div className="layout">
          <header>
            <nav>
              <ul>
                <li><Link to="/">Home</Link></li>
                <li><Link to="/blog">Blog</Link></li>
                <li><Link to="/contact">Contact</Link></li>
              </ul>
            </nav>
          </header>
          <main>{children}</main>
          <footer>
            <p>&copy; 2024 My App</p>
          </footer>
        </div>
      );
    };

    export default Layout;
```

Home.js:

```
    // src/pages/Home.js
    import React from 'react';

    const Home = () => {
      return (
        <div>
          <h1>Welcome to the Home Page</h1>
          <p>This is the homepage of the application.</p>
        </div>
      );
    };
    export default Home;
```

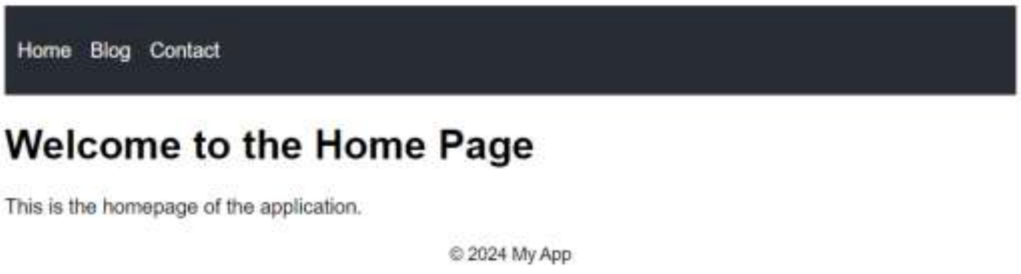
Blogs.js:

```
// src/pages/Blog.js
import React from 'react';

const Blog = () => {
  return (
    <div>
      <h1>Blog Page</h1>
      <p>Here are some interesting blog posts.</p>
    </div>
  );
};

export default Blog;
```

Contact.js:

```
// src/pages/Contact.js
import React from 'react';

const Contact = () => {
  return (
    <div>
      <h1>Contact Us</h1>
      <p>Get in touch through this page.</p>
    </div>
  );
};

export default Contact;
```

OUTPUT:

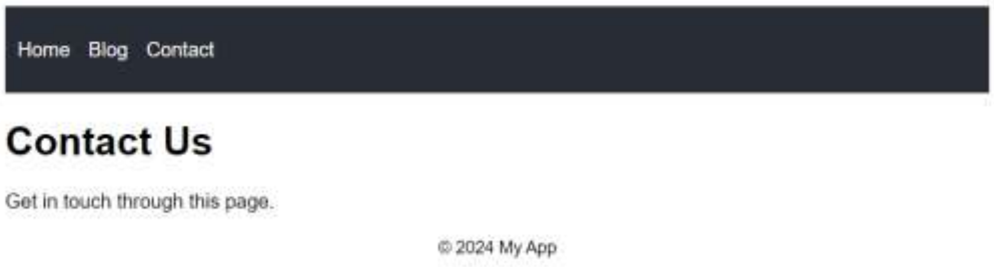


Home Blog Contact

Welcome to the Home Page

This is the homepage of the application.

© 2024 My App

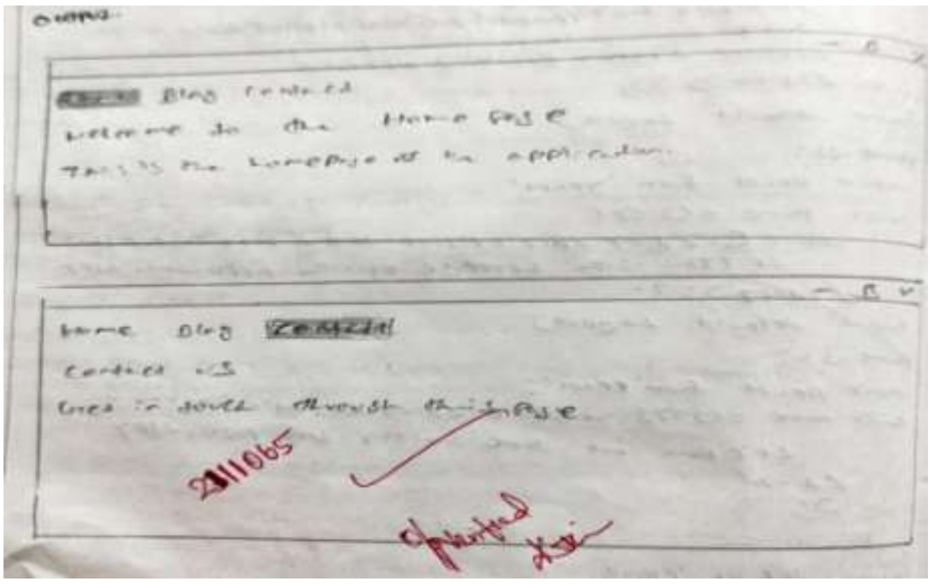


Home Blog Contact

Contact Us

Get in touch through this page.

© 2024 My App



Home Blog Contact

Welcome to the Home Page

This is the homepage of the application.

Home Blog Contact

Contact us

Get in touch through this page

2311065

[illegible]

Built a ReactJS front-end application for managing and viewing blog articles with seamless navigation using React Router. Implemented multiple pages including Home, Blog, and Contact for a complete blog experience.

RESULT:

Thus, the front-end react js application for handling routing and navigation is executed successfully.

Ex No: 7 **Date: 11/09/2024**	**Database CRUD operations using MongoDB**

Aim:

To execute CRUD operations using **MongoDB**

MONGODB

MongoDB is an object-oriented, simple, dynamic, and scalable NoSQL database. It is based on the NoSQL document store model. The data objects are stored as separate documents inside a collection - instead of storing the data into the columns and rows of a traditional relational database. The motivation of the MongoDB language is to implement a data store that provides high performance, high availability, and automatic scaling. MongoDB is extremely simple to install and implement. MongoDB uses JSON or BSON documents to store data

DATABASE

Database is a physical container for collections. Each database gets its own set of files on the file system. A single MongoDB server typically has multiple databases.

COLLECTION

Collection is a group of MongoDB documents. It is the equivalent of an RDBMS table. A collection exists within a single database. Documents within a collection can have different fields.

DOCUMENTS

A document is a set of key-value pairs. Documents have dynamic schema

Create Database:

MongoDB **use DATABASE_NAME** is used to create database. The command will create a new database if it doesn't exist, otherwise it will return the existing database

Syntax

```
use DATABASE_NAME
```

To check your currently selected database, use the command **db**

If you want to check yourdatabases list, use the command **show dbs**.

Drop Database:

MongoDB **db.dropDatabase()** command is used to drop a existing database.

Syntax

```
db.dropDatabase()
```

Create Collection

MongoDB **db.createCollection(name, options)** is used to create collection. In the command, **name** is name of collection to be created. **Options** is a document and is used to specify configuration of collection.

Syntax

```
db.createCollection(name, options)
```

Drop Collection

MongoDB's **db.collection.drop()** is used to drop a collection from the database.

Syntax

```
db.COLLECTION_NAME.drop()
```

<u>EXECUTED QUERIES:</u>

1. Create a database in MongoDB

```
>use student
```

Output : switched to db student

2. Perform an operation to check your database list

```
>show dbs
```

Output : Local 0.078 GB

3. Perform an operation to check your currently working database

```
>db
```

Output : student

4. Execute a query to drop the created database

```
>db.dropDatabase()
```

Output : {“OK”:1}

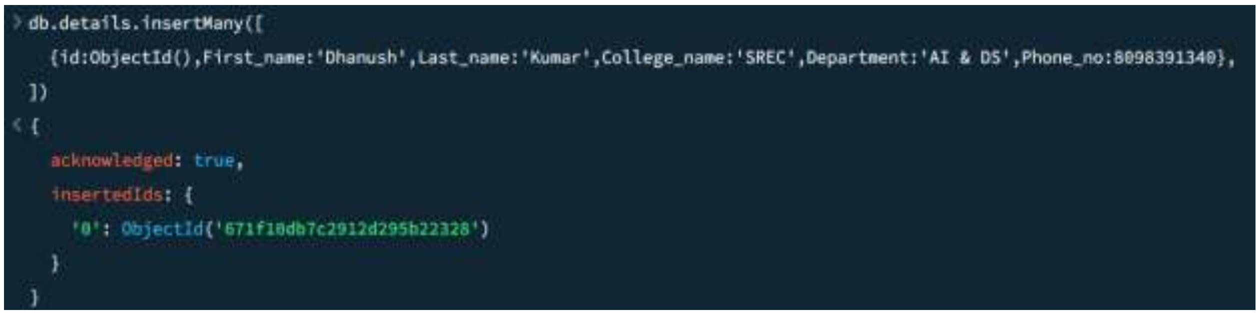
5. Create a collection

```
>db.CreateCollection(“details”)
```

Output : {“OK”:1}

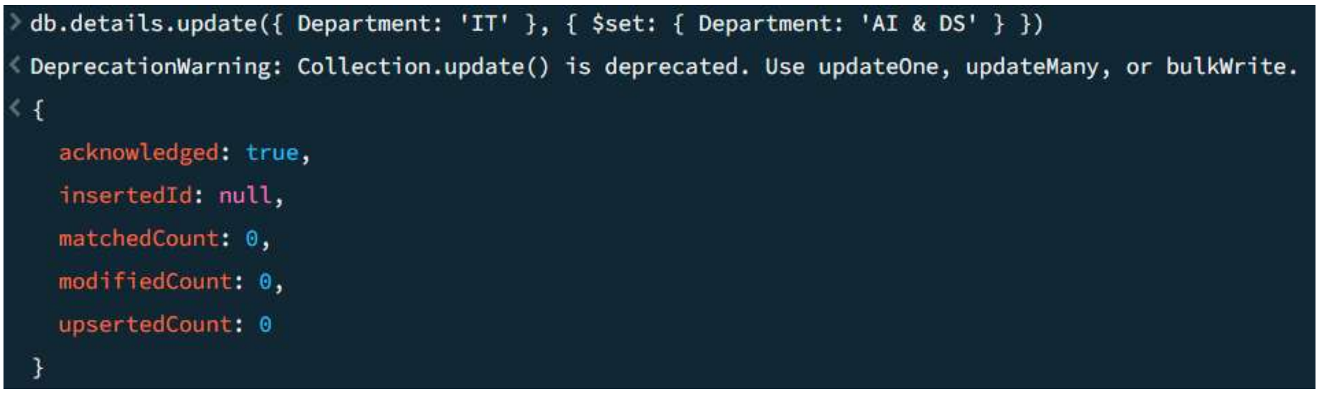
EXECUTED QUERIES:

1. Insert values into the created collection



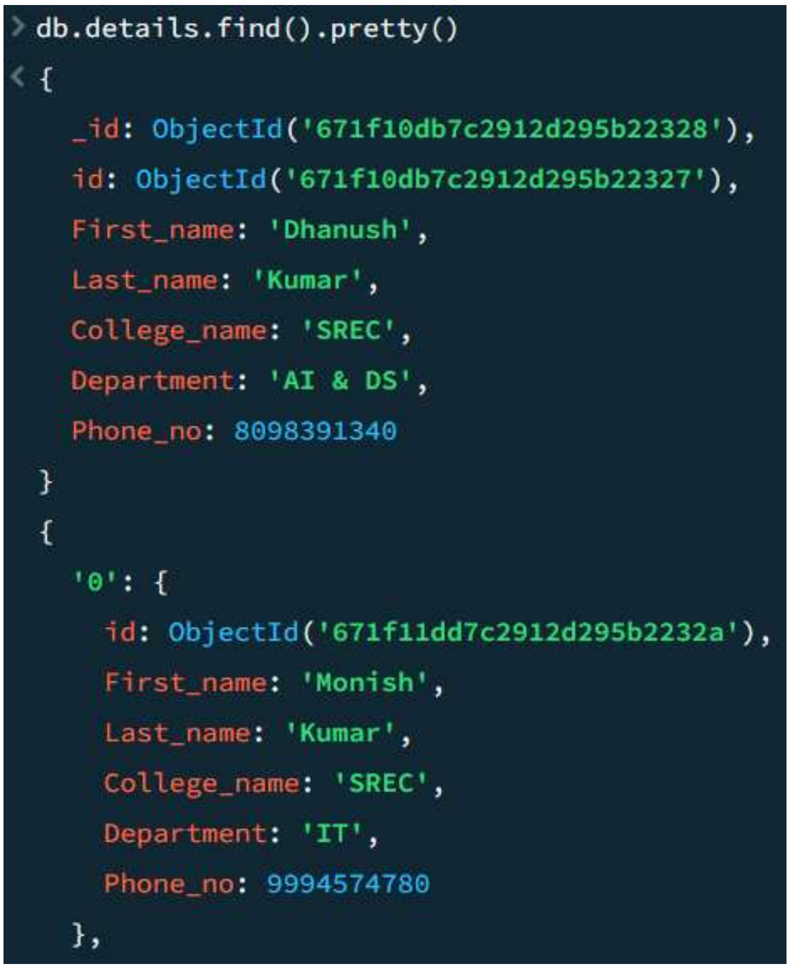
```
> db.details.insertMany([
    {id:ObjectId(),First_name:'Dhanush',Last_name:'Kumar',College_name:'SREC',Department:'AI & DS',Phone_no:8098391340},
  ])
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId('671f10db7c2912d295b22328')
    }
  }
```

2. Update the collection



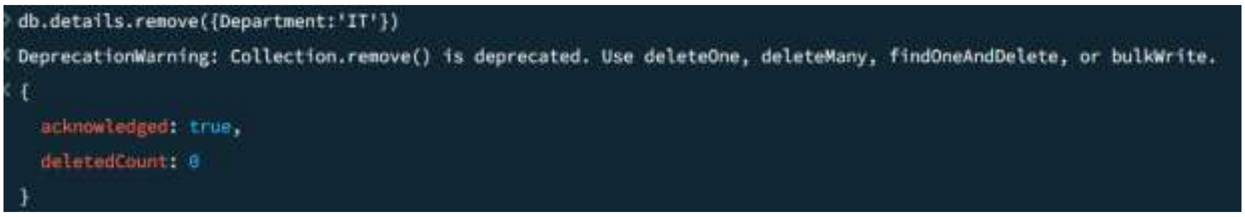
```
> db.details.update({ Department: 'IT' }, { $set: { Department: 'AI & DS' } })
< DeprecationWarning: Collection.update() is deprecated. Use updateOne, updateMany, or bulkWrite.
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 0,
    modifiedCount: 0,
    upsertedCount: 0
  }
```

3. Find the collection using Pretty()

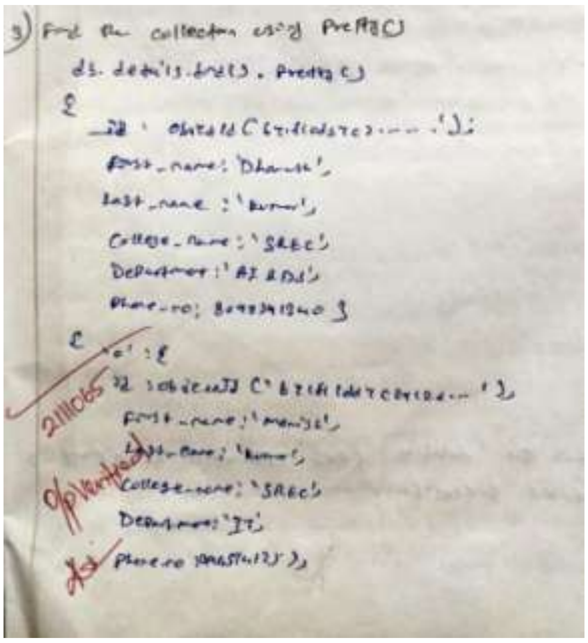


```
> db.details.find().pretty()
< {
    _id: ObjectId('671f10db7c2912d295b22328'),
    id: ObjectId('671f10db7c2912d295b22327'),
    First_name: 'Dhanush',
    Last_name: 'Kumar',
    College_name: 'SREC',
    Department: 'AI & DS',
    Phone_no: 8098391340
  }
  {
    '0': {
      id: ObjectId('671f11dd7c2912d295b2232a'),
      First_name: 'Monish',
      Last_name: 'Kumar',
      College_name: 'SREC',
      Department: 'IT',
      Phone_no: 9994574780
    },
```

4. Remove the collection



```
> db.details.remove({Department:'IT'})
< DeprecationWarning: Collection.remove() is deprecated. Use deleteOne, deleteMany, findOneAndDelete, or bulkWrite.
< {
    acknowledged: true,
    deletedCount: 0
  }
```



```
3) Find the collection using Pretty()
   db. details.find() . Pretty ()
   {
     _id : ObjectId ('[illegible]......');
     first_name: 'Dhanush',
     last_name : 'kumar',
     College_name : 'SREC',
     Department : 'AI & DS',
     Phone_no: [illegible] }
   {
     _id : ObjectId ('[illegible]......');
     first_name : 'manoj',
     last_name: 'kumar',
     College_name: 'SREC',
     Department: 'IT',
     Phone_no [illegible] });
```

21/11/05

Output Verified

[illegible]

Implemented MongoDB to handle CRUD operations in a user registration system, utilizing collections to store and manage user details. Built queries for creating, listing, and dropping databases and collections to support dynamic user data management.

RESULT:

Thus, the database CRUD operations using MongoDB is executed successfully.

Ex No: 8 **Date: 18/09/2024**	**BACK-END WEB DEVELOPMENT USING NODEJS**

AIM

To design a web application using NodeJS application to make registration by the user.

ALGORITHM

STEP 1: Install Node.js.

STEP 2: Create a Folder named Event and open it in visual studio code

STEP 3: Open terminal and type “npm init” to install package.json file

STEP 4: Install express by the command “npm install express” in the terminal

STEP 5: Create a HTML file named index.html and create a form forregistration of user by enteringhis/her details

STEP 6: Create a javascript file named example.js

STEP 7: In example.js, write an express code to connect to server and to get thedetails of formsubmitted by user by listening to the port 8000.

STEP 8: Execute the js file by the command “node example.js”

STEP 9: In the browser, paste the html file path and fill the form.

STEP 10: Browser will be redirected to server address and details submitted byuser will be displayed.

SOURCE CODE

<u>index.html</u>

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

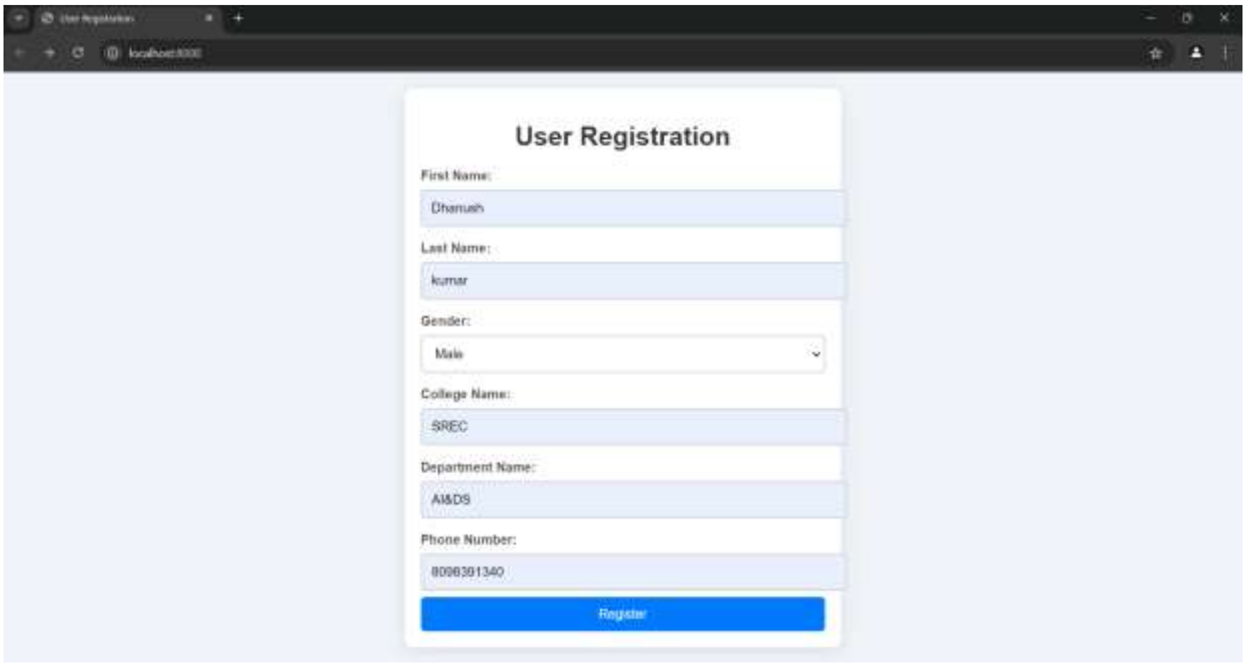
```
</head>
<body>
  <div class="container">
    <h1>User Registration</h1>
    <form action="http://localhost:8000/register" method="POST">
      <label for="firstName">First Name:</label>
      <input type="text" id="firstName" name="firstName" required>

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

      <label for="sex">Sex:</label>
      <select id="sex" name="sex" required>
        <option value="" disabled selected>Select your sex</option>
        <option value="male">Male</option>
        <option value="female">Female</option>
        <option value="other">Other</option>
      </select>

      <label for="college">College Name:</label>
      <input type="text" id="college" name="college" required>

      <label for="department">Department Name:</label>
      <input type="text" id="department" name="department" required>

      <label for="phone">Phone Number:</label>
      <input type="tel" id="phone" name="phone" required>

      <button type="submit">Register</button>
    </form>
  </div>
</body>
</html>
```

example.js

```
// example.js
const express = require('express');
const bodyParser = require('body-parser');
const path = require('path');
const fs = require('fs'); // Require the file system module
const app = express();
const PORT = 8000;

// Middleware
app.use(bodyParser.urlencoded({ extended: true }));
app.use(express.static(path.join(__dirname))); // Serve static files

// Route to handle registration
app.post('/register', (req, res) => {
  const { firstName, lastName, sex, college, department, phone } = req.body;
```

```
    // Prepare the data to save in JSON format
    const userData = {
        firstName,
        lastName,
        sex,
        college,
        department,
        phone
    };

    // Read existing data from users.json, if it exists
    fs.readFile('users.json', (err, data) => {
        if (err) {
            console.error(err);
            return res.status(500).send('Error reading data.');
        }

        // Parse existing data or initialize an empty array
        const users = data.length > 0 ? JSON.parse(data) : [];

        // Add the new user to the array
        users.push(userData);

        // Write the updated array back to users.json
        fs.writeFile('users.json', JSON.stringify(users, null, 2), (err) => {
            if (err) {
                console.error(err);
                return res.status(500).send('Error saving data.');
            }

            // Respond with the user details
            res.send(`
                <h1>User Registration Details</h1>
                <p>First Name: ${firstName}</p>
                <p>Last Name: ${lastName}</p>
                <p>Sex: ${sex}</p>
                <p>College Name: ${college}</p>
                <p>Department Name: ${department}</p>
                <p>Phone Number: ${phone}</p>
            `);
        });
    });
});

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

OUTPUT



User Registration

First Name:

Dhanush

Last Name:

kumar

Gender:

Male

College Name:

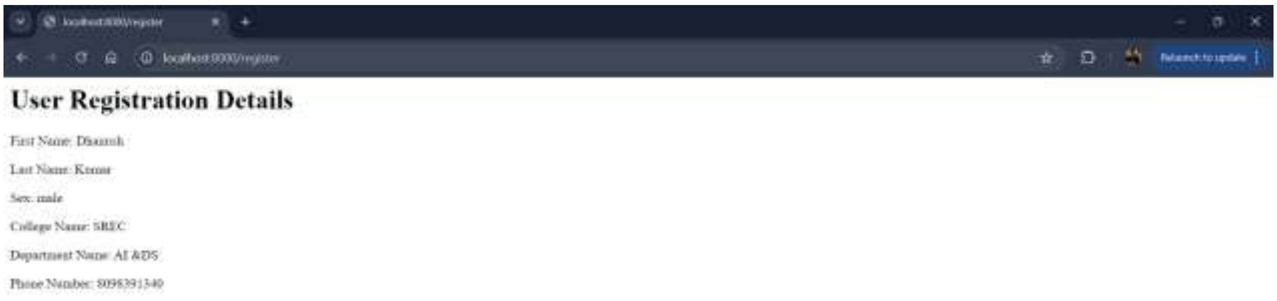
SREC

Department Name:

AI&DS

Phone Number:

8098391340



User Registration Details

First Name: Dhanush

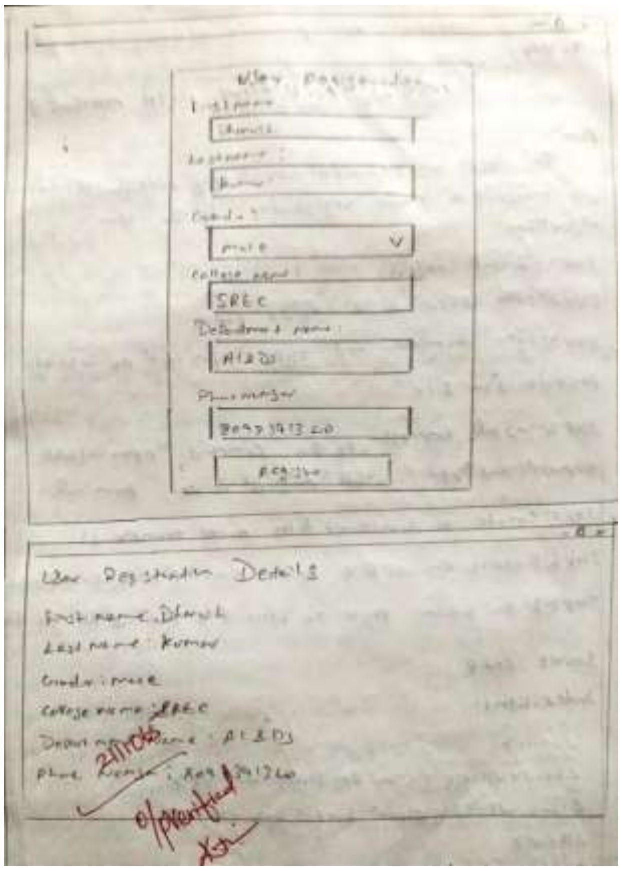
Last Name: Kumar

Sex: male

College Name: SREC

Department Name: AI &DS

Phone Number: 8098391340



User Registration

First name

[illegible]

Last name :

[illegible]

Gender :

male

College name

SREC

Department name :

AI&DS

Phone number

[illegible]

Register

User Registration Details

First name : [illegible]

Last name : Kumar

Gender : male

College name : SREC

Depart name : AI&DS

Phone Number : [illegible]

Developed a Node.js application with Express to handle user registrations via a form, enabling real-time data capture and server connectivity. Implemented HTML for the form interface and configured Express to listen on port 8000 for form submissions.

RESULT

Thus, the web application using NodeJS for registration by the user has been implemented successfully.

Ex No: 9 **Date: 25/09/2024**	NODEJS APPLICATION WITH MONGODB

AIM

To design a web application using NodeJS application with mongoDB to make registration by the user.

ALGORITHM

STEP 1: Install Node.js.

STEP 2: Create a Folder named Event and open it in visual studio code

STEP 3: Open terminal and type “npm init” to install package.json file

STEP 4: Install express by the command “npm install express” and “npm install mongodb” in the terminal

STEP 5: Create a HTML file named index.html and create a form forregistration of user by enteringhis/her details

STEP 6: Create a javascript file named example.js

STEP 7: In example.js, write an express code to connect to server and to get thedetails of formsubmitted by user by listening to the port 8000.

STEP 8: Execute the js file by the command “node example.js”

STEP 9: In the browser, paste the html file path and fill the form.

STEP 10: Browser will be redirected to server address and details submitted byuser will be displayed.

SOURCE CODE

index.html

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

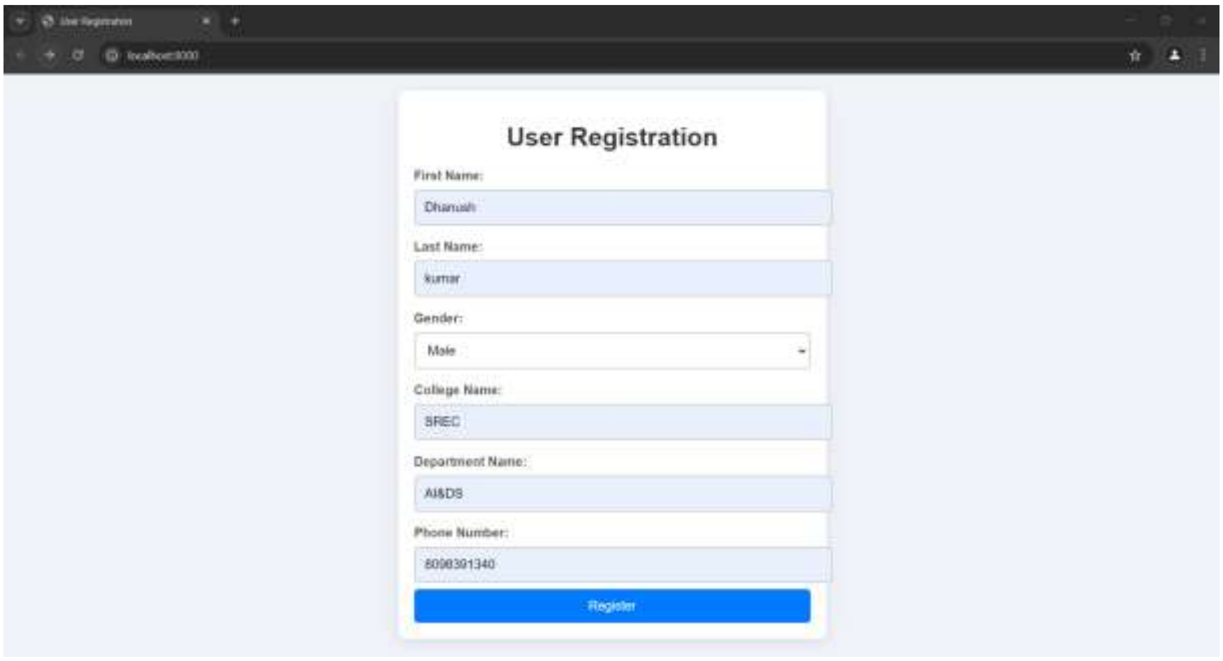
```
<body>
  <div class="container">
    <h1>User Registration</h1>
    <form action="http://localhost:8000/register" method="POST">
      <label for="firstName">First Name:</label>
      <input type="text" id="firstName" name="firstName" required>

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

      <label for="Gender">Gender:</label>
      <select id="Gender" name="Gender" required>
        <option value="" disabled selected>Select your gender</option>
        <option value="male">Male</option>
        <option value="female">Female</option>
        <option value="other">Other</option>
      </select>
      <label for="college">College Name:</label>
      <input type="text" id="college" name="college" required>

      <label for="department">Department Name:</label>
      <input type="text" id="department" name="department" required>
      <label for="phone">Phone Number:</label>
      <input type="tel" id="phone" name="phone" required>
      <button type="submit">Register</button>
    </form>
  </div>
</body>
</html>
```

example.js

```
const express = require('express');
const mongoose = require('mongoose');
const path = require('path'); // Add this to handle file paths
const app = express();

// Middleware to parse form data
app.use(express.urlencoded({ extended: true }));

// Serve static files from the current directory
app.use(express.static(path.join(__dirname))); // Serve files in the same directory as this
script

// MongoDB connection URL
const url = 'mongodb://localhost:27017/eventDB';

// Connect to MongoDB
mongoose.connect(url, { useNewUrlParser: true, useUnifiedTopology: true })
  .then(() => console.log('Connected to Database'))
  .catch(error => console.error('Database connection error:', error));
// Define a User schema
```

```
const userSchema = new mongoose.Schema({
   firstName: { type: String, required: true },
   lastName: { type: String, required: true },
   gender: String,
   college: String,
   department: String,
   phone: String
});

// Create a User model based on the schema
const User = mongoose.model('User', userSchema);

// Serve the index.html file on the root route
app.get('/', (req, res) => {
   res.sendFile(path.join(__dirname, 'index.html')); // Send index.html file
});

// Registration route
app.post('/register', async (req, res) => {
   const user = new User({
      firstName: req.body.firstName,
      lastName: req.body.lastName,
      gender: req.body.gender,
      college: req.body.college,
      department: req.body.department,
      phone: req.body.phone
   });
   try {
      await user.save();
      // Send back a detailed response with all submitted user data
      res.send(`
         <h1>Thank you, ${user.firstName}! Your registration was successful.</h1>
         <h2>Registration Details:</h2>
         <p>First Name: ${user.firstName}</p>
         <p>Last Name: ${user.lastName}</p>
         <p>Gender: ${user.gender}</p>
         <p>College Name: ${user.college}</p>
         <p>Department Name: ${user.department}</p>
         <p>Phone Number: ${user.phone}</p>
      `);
   } catch (error) {
      console.error('Error saving user:', error.message);
      res.status(500).send('<h1>Failed to register. Please try again later.</h1>'); }
});
// Start the server
app.listen(8000, () => {
   console.log('Server is running on http://localhost:8000');

});
```

OUTPUT



User Registration

First Name: Dhanush

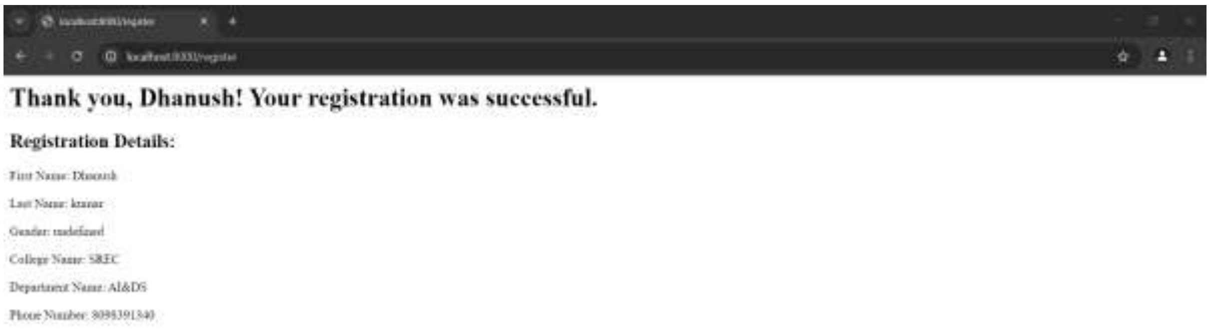
Last Name: kumar

Gender: Male

College Name: SREC

Department Name: AI&DS

Phone Number: 8098391340



Thank you, Dhanush! Your registration was successful.

Registration Details:

First Name: Dhanush

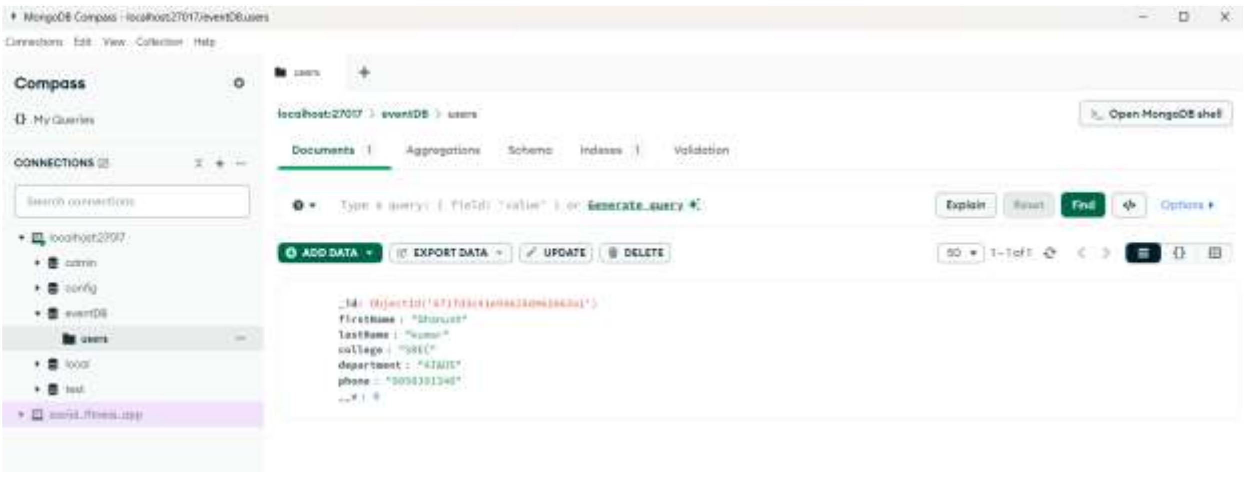
Last Name: kumar

Gender: undefined

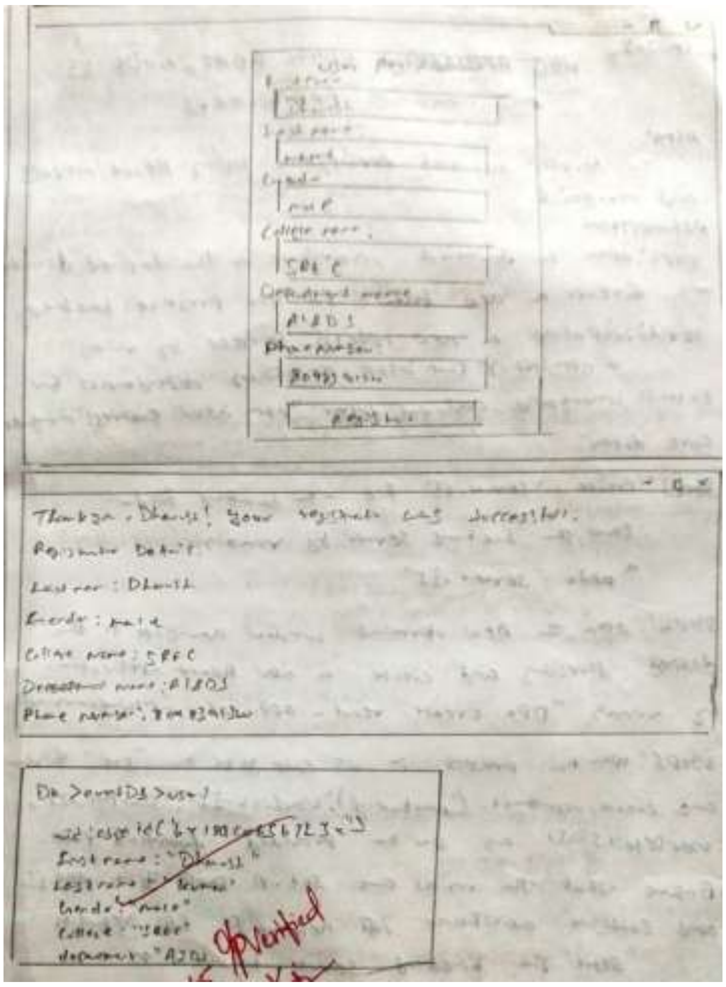
College Name: SREC

Department Name: AI&DS

Phone Number: 8098391340



```
_id: ObjectId('[illegible]')
firstName : "Dhanush"
lastName : "kumar"
college : "SREC"
department : "AI&DS"
phone : "8098391340"
__v : 0
```



User Registration

First name

Dhanush

Last name

Kumar

Gender

male

College name

SREC

Department name

AIDS

Phone number

80493 [illegible]

Register

Thanks, Dhanush! Your registration was successful.

Registration Details

First name : Dhanush

Gender : male

College name : SREC

Department name : AIDS

Phone number : [illegible]

```
Db >eventDB >user)
   _id : ObjectId("[illegible]")
   first name : "Dhanush"
   Last name : "Kumar"
   Gender : "male"
   College : "SREC"
   department : "AIDS"
```

Verified

Set up a Node.js web application with MongoDB to manage user registration data, storing form details like name, email, and password in a MongoDB collection. Configured Express to handle form submissions and save user information into the 'Event' database under a 'Users' collection.

RESULT

Thus, the web application using NodeJS applications with mongobd for registration by the user has been implemented successfully.

Ex No : 10 **Date: 1/10/2024**	**WEB APPLICATION USING REACT, NODEJS AND MONGODB FRAMEWORKS**

AIM

To develop a Web Application using React, Nodejs and MongoDB.

ALGORITHM

STEP 1: Open a terminal, navigate to the desired directory, and create a new folder for the project backend.

STEP 2: Initialize a new Node.js project by running:” npm init –y” Then install the necessary dependencies for Express, Mongoose, CORS, and dotenv
“ **npm install express mongoose cors dotenv”**

STEP 3: Create a `server.js` file in the backend folder, then copy the provided server code into it. Update your `.env` file with your MongoDB connection URI.

STEP 4: Start the backend server by running:
“node server.js” Ensure you see "Server is running on port 8080" and "MongoDB connected"messages in the terminal.

STEP 5: Open a new terminal window, navigate to the desired directory, and create a new React application by running:“ **npx create-react-app user-management-app**”

STEP 6: Open the project in VS Code, go to the `src` folder, and create components (`Navbar.js`, `UserList.js`, `UserForm.js`, `UserDetails.js`) as per the provided frontend code. Ensure that the routes are set up properly in `App.js`, and configure `axiosInstance` if needed for API requests.

Start the backend server by running:
node server.js

Start the frontend React application by running:
npm start

SOURCE CODE

BACKEND

Server.js:

```
            / // server.js
const express = require('express');
const mongoose = require('mongoose');
const cors = require('cors');
const dotenv = require('dotenv');
dotenv.config();
const app = express();
const PORT = process.env.PORT || 8080;
// Middleware
app.use(cors());
app.use(express.json());
// Connect to MongoDB
console.log('Connecting to MongoDB with URI:', process.env.MONGO_URI);
mongoose.connect(process.env.MONGO_URI, {
  useNewUrlParser: true,
  useUnifiedTopology: true,
})
  .then(() => console.log('MongoDB connected'))
  .catch(err => console.log(err));
// User Schema
const userSchema = new mongoose.Schema({
  name: { type: String, required: true },
  email: { type: String, required: true, unique: true },
  age: { type: Number },
});
const User = mongoose.model('User', userSchema);
// Routes
// Get all users
app.get('/api/users', async (req, res) => {
  try {
    const users = await User.find();
    res.json(users);
  } catch (err) {
    res.status(500).json({ message: err.message });
  }
});
// Get a single user
app.get('/api/users/:id', getUser, (req, res) => {
  res.json(res.user);
});
// Create a new user
app.post('/api/users', async (req, res) => {
  const user = new User({
    name: req.body.name,
    email: req.body.email,
    age: req.body.age,
  });
  try {
    const newUser = await user.save();
    res.status(201).json(newUser);
  } catch (err) {
    res.status(400).json({ message: err.message });
  }
});

// Update a user
app.put('/api/users/:id', getUser, async (req, res) => {
  if (req.body.name != null) {
    res.user.name = req.body.name;
  }
  if (req.body.email != null) {
    res.user.email = req.body.email;
  }
  if (req.body.age != null) {
    res.user.age = req.body.age;
  }
  try {
    const updatedUser = await res.user.save();
    res.json(updatedUser);
  } catch (err) {
    res.status(400).json({ message: err.message });
  }
});
// Delete a user
// Delete a user without relying on res.user.remove()
app.delete('/api/users/:id', async (req, res) => {
  try {
    const deletedUser = await User.findByIdAndDelete(req.params.id);
    if (deletedUser == null) {
      return res.status(404).json({ message: 'Cannot find user' });
    }
    res.json({ message: 'Deleted User' });
  } catch (err) {
    res.status(500).json({ message: err.message });
  }
});
// Middleware function to get user by ID
async function getUser(req, res, next) {
  let user;
  try {
    user = await User.findById(req.params.id);
    if (user == null) {
      return res.status(404).json({ message: 'Cannot find user' });
    }
  } catch (err) {
    return res.status(500).json({ message: err.message });
  }

  res.user = user;
  next();
}
// Start the server
app.listen(PORT, () => {
  console.log(`Server is running on port ${PORT}`);
});
```

FRONTEND

Navbar.js:

```
// src/components/Navbar.js
import React from 'react';
import { Link } from 'react-router-dom';

const Navbar = () => {
  return (
    <nav style={navStyle}>
      <h2>User Management</h2>
      <ul style={ulStyle}>
        <li><Link to="/">Home</Link></li>
        <li><Link to="/add">Add User</Link></li>
      </ul>
    </nav>
  );
};
const navStyle = {
  display: 'flex',
  justifyContent: 'space-between',
  background: '#333',
  color: '#fff',
  padding: '10px',
};

const ulStyle = {
  listStyle: 'none',
  display: 'flex',
};

export default Navbar;
```

UserDetails.js:

```
// src/components/UserDetails.js
import React, { useEffect, useState } from 'react';
import { Card, CardContent, Typography, Button, Box } from '@mui/material';
import { Link as RouterLink } from 'react-router-dom';
import axiosInstance from '../axiosInstance';
import { motion } from 'framer-motion';
import { FaEdit, FaArrowLeft } from 'react-icons/fa';
import { useParams } from 'react-router-dom';

const UserDetails = () => {
  const [user, setUser] = useState(null);
  const { id } = useParams();
  useEffect(() => {
    axiosInstance.get(`/users/${id}`)
      .then(res => setUser(res.data))
      .catch(err => console.error(err));
  }, [id]);

  if (!user) return <Typography>Loading...</Typography>;

  return (
    <Box sx={{ display: 'flex', justifyContent: 'center', p: 2 }}>
      <motion.div
        initial={{ y: 50, opacity: 0 }}
        animate={{ y: 0, opacity: 1 }}
        transition={{ duration: 0.8, type: 'spring', stiffness: 100 }}
      >
        <Card sx={{ minWidth: 275, maxWidth: 500 }}>
          <CardContent>
            <Typography variant="h5" component="div" gutterBottom>
              {user.name}
            </Typography>
            <Typography variant="body1" color="text.secondary">
              <strong>Email:</strong> {user.email}
            </Typography>
            <Typography variant="body1" color="text.secondary">
              <strong>Age:</strong> {user.age}
            </Typography>
          </CardContent>
          <Box sx={{ display: 'flex', justifyContent: 'flex-end', p: 2 }}>
            <Button
              component={RouterLink}
              to={`/edit/${user._id}`}
              variant="contained"
              color="primary"
              startIcon={<FaEdit />}
              sx={{ mr: 1 }}

              whileHover={{ scale: 1.05 }}
              whileTap={{ scale: 0.95 }}
            >
              Edit
            </Button>
            <Button
              component={RouterLink}
              to="/"
              variant="outlined"
              color="secondary"
              startIcon={<FaArrowLeft />}

              whileHover={{ scale: 1.05 }}
              whileTap={{ scale: 0.95 }}
           >
              Back
            </Button>
          </Box>
          </Card>
        </motion.div>
      </Box>
    );
  };

  export default UserDetails;
```

UserForm.js:

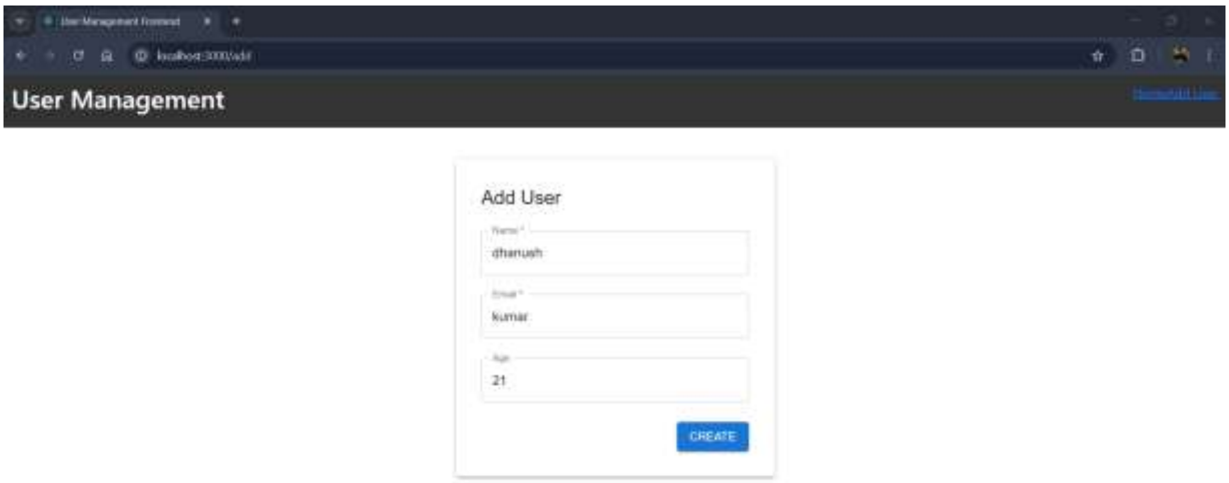
```
// src/components/UserForm.js
import React, { useState, useEffect } from 'react';
import { TextField, Button, Box, Typography, Paper } from '@mui/material';
import { useNavigate, useParams } from 'react-router-dom';
import axiosInstance from '../axiosInstance';
import { motion } from 'framer-motion';

const UserForm = () => {
  const [user, setUser] = useState({ name: '', email: '', age: '' });
  const navigate = useNavigate();
  const { id } = useParams();

  const isEdit = Boolean(id);

  useEffect(() => {
    if (isEdit) {
      axiosInstance.get(`/users/${id}`)
        .then(res => setUser(res.data))
        .catch(err => console.error(err));
    }
  }, [id, isEdit]);

  const handleChange = (e) => {
    const { name, value } = e.target;
    setUser(prevState => ({
      ...prevState,
      [name]: value
    }));
  };

  const handleSubmit = async (e) => {
    e.preventDefault();
    try {
      if (isEdit) {
        await axiosInstance.put(`/users/${id}`, user);
      } else {
        await axiosInstance.post('/users', user);
      }
      navigate('/');
    } catch (err) {
      console.error(err);
      alert('An error occurred while saving the user.');
    }
  };

  return (
    <Box sx={{ display: 'flex', justifyContent: 'center', p: 2 }}>
      <motion.div
        initial={{ scale: 0.8, opacity: 0 }}
        animate={{ scale: 1, opacity: 1 }}
        transition={{ type: 'spring', stiffness: 100 }}
      >
        <Paper elevation={3} sx={{ p: 4, width: 400 }}>
          <Typography variant="h5" gutterBottom>
            {isEdit ? 'Edit User' : 'Add User'}
          </Typography>
          <form onSubmit={handleSubmit}>
            <TextField
              label="Name"
              name="name"
              value={user.name}
              onChange={handleChange}
              required
              fullWidth
              margin="normal"
              variant="outlined"
              component={motion.div}
              whileHover={{ scale: 1.02 }}
            />
            <TextField
              label="Email"
              name="email"
              type="email"
              value={user.email}
              onChange={handleChange}
              required
              fullWidth
              margin="normal"
              variant="outlined"
              component={motion.div}
              whileHover={{ scale: 1.02 }}
            />
            <TextField
              label="Age"
              name="age"
              type="number"
              value={user.age}
              onChange={handleChange}
              fullWidth
              margin="normal"
              variant="outlined"
              component={motion.div}
              whileHover={{ scale: 1.02 }}
            />
            <Box sx={{ display: 'flex', justifyContent: 'flex-end', mt: 2 }}>
              <Button
                type="submit"
                variant="contained"
                color="primary"
                component={motion.button}
                whileHover={{ scale: 1.05 }}
                whileTap={{ scale: 0.95 }}
              >
                {isEdit ? 'Update' : 'Create'}
              </Button>
            </Box>
          </form>
        </Paper>
      </motion.div>
    </Box>);};

  export default UserForm;
```

UserList.js:

```
// src/components/UserList.js
import React, { useEffect, useState } from 'react';
import { Link } from 'react-router-dom';
import axios from 'axios';
const UserList = () => {
  const [users, setUsers] = useState([]);

  const fetchUsers = async () => {
    try {
      const res = await axios.get('http://localhost:8080/api/users');
      setUsers(res.data);
    } catch (err) {
      console.error(err);
    } };
  const deleteUser = async (id) => {
    try {
      await axios.delete(`http://localhost:8080/api/users/${id}`);
      setUsers(users.filter(user => user._id !== id));
    } catch (err) {
      console.error(err);
    }};
  useEffect(() => {
    fetchUsers();
  }, []);
  return (
    <div style={{ padding: '20px' }}>
      <h2>User List</h2>
      <table border="1" cellPadding="10" cellSpacing="0">
```

```
        <thead>
          <tr>
            <th>Name</th>
            <th>Email</th>
            <th>Age</th>
            <th>Actions</th>
          </tr>
        </thead>
        <tbody>
          {users.map(user => (
            <tr key={user._id}>
              <td><Link to={`/users/${user._id}`}>{user.name}</Link></td>
              <td>{user.email}</td>
              <td>{user.age}</td>
              <td>
                <Link to={`/edit/${user._id}`} style={{ marginRight: '10px' }}>Edit</Link>
                <button onClick={() => deleteUser(user._id)}>Delete</button>
              </td>
            </tr>
          ))}
          {users.length === 0 && (
            <tr>
              <td colSpan="4">No users found.</td>
            </tr>
          )}
        </tbody>
      </table>
    </div>
  );
};

export default UserList;
```

OUTPUT

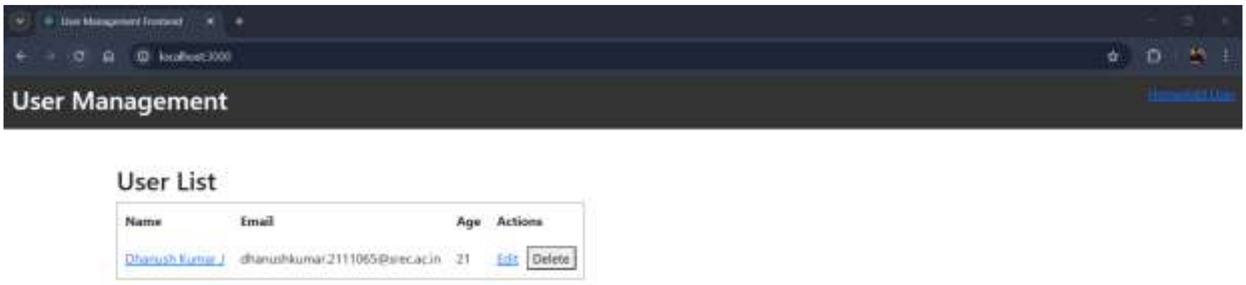


Add User

Name *
dhanush

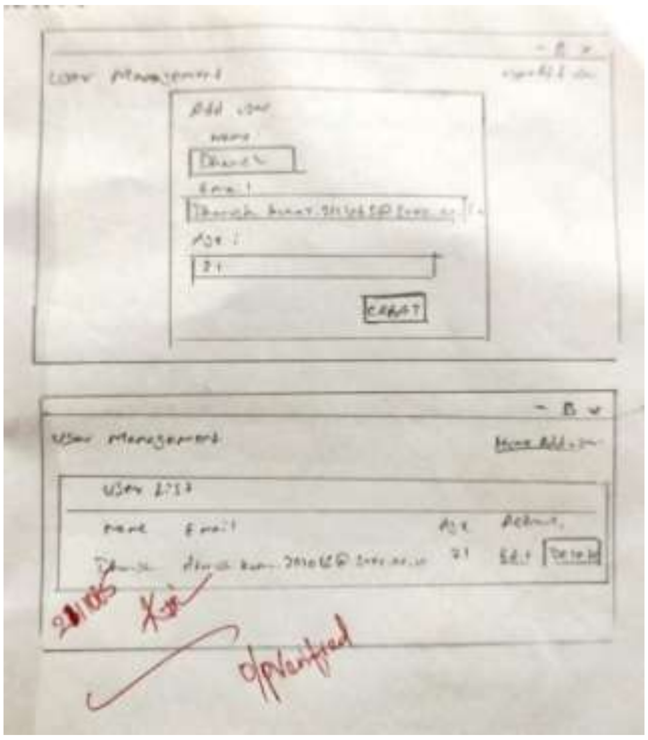
Email *
kumar

Age
21



User List

Name	Email	Age	Actions
Dhanush Kumar J	dhanushkumar.2111065@srec.ac.in	21	Edit Delete



Developed a full-stack web application using React, Node.js, and MongoDB to manage user data with CRUD operations. The React frontend integrates components for displaying, adding, and updating user details, while the Node.js backend handles API requests and connects to MongoDB for data persistence.

RESULT

Thus, the web application using React, Nodejs and MongoDB is executed successfully.