

# **WEB DEVELOPMENT 2**

## **ITE-365**

Submitted in partial fulfilment of the requirements for the award of the degree  
of  
**B.Tech.**  
in  
**Information Technology**

**SUBMITTED TO:**

Prof. S. k. Malik

**SUBMITTED BY:**

NAME – ABHISHEK KUMAR

ENROLLMENT NO.-04716401521

BATCH - 2021-25(7th sem)



**UNIVERSITY SCHOOL OF INFORMATION COMMUNICATION AND TECHNOLOGY**  
**GURU GOBIND SINGH INDRAPRASTHA UNIVERSITY**  
**NOVEMBER – 2024**

## INDEX

S.NO.	NAME OF PRACTICAL	DATE	SIGN
1	Using various HTML tags for creating different web pages.		
2	Using various Form tags for interactivity, authentication, date validation, etc.		
3	Using Semantic HTML tags /tags associated with interactivity.		
4	Using DHTML tags in concern to client server application.		
5	Using JavaScript/CSS for dynamic web pages.		
6	Utilize HTML/CSS and JavaScript frameworks (ReactJS, NextJS) to construct dynamic user interfaces.		
7	Create various databases using SQL/MongoDB/or other to show interactivity .		
8	Perform CRUD operations using React JS as frontend technology and Node JS as backend technology showing database interactivity using any database.		
9	Develop robust back-end systems using Node.js.		
10	Showing database interactivity using PHP/Python/or any current technology used in web industry.		
11	Any current web industry relevant example of database usage and interactivity using any suitable backend technology.		

# PRACTICAL 1

**Aim:** Using various HTML tags for creating different web pages.

## **CODE:**

### **1. Index.html**

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Home - My Website</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <header>
    <h1>Welcome to My Website</h1>
    <nav>
      <ul>
        <li><a href="index.html">Home</a></li>
        <li><a href="about.html">About</a></li>
        <li><a href="contact.html">Contact</a></li>
      </ul>
    </nav>
  </header>
  <main>
    <section>
      <h2>Introduction</h2>
      <p>This website is a showcase of HTML and CSS skills, featuring various types of content and layout options.</p>
    </section>
    <section>
      <h2>Featured Section</h2>
      <p>Explore the other pages to learn more about me, view the gallery, or get in touch through the contact page.</p>
    </section>
  </main>
  <footer>
    <p>&copy; 2024 My Website</p>
  </footer>
</body>
</html>
```

### **2. About.html**

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>About - My Website</title>
<link rel="stylesheet" href="style.css">
</head>
<body>
  <header>
    <h1>About Me</h1>
    <nav>
      <ul>
        <li><a href="index.html">Home</a></li>
        <li><a href="about.html">About</a></li>
        <li><a href="contact.html">Contact</a></li>
      </ul>
    </nav>
  </header>
  <main>
    <section>
      <h2>Who I Am</h2>
      <p>This section provides a brief background about me, my interests, and my professional
experience.</p>
    </section>
    <section>
      <h3>Hobbies</h3>
      <ul>
        <li>Photography</li>
        <li>Coding</li>
        <li>Traveling</li>
      </ul>
    </section>
  </main>
  <footer>
    <p>&copy; 2024 My Website</p>
  </footer>
</body>
</html>

```

### 3. Contact.html

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Contact - My Website</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <header>
    <h1>Contact Me</h1>
    <nav>
      <ul>
        <li><a href="index.html">Home</a></li>

```

```
<li><a href="about.html">About</a></li>
<li><a href="contact.html">Contact</a></li>
</ul>
</nav>
</header>
<main>
<section>
<h2>Get in Touch</h2>
<form action="#" method="post">
<div>
<label for="name">Name:</label>
<input type="text" id="name" name="name" required>
</div><br>
<div>
<label for="email">Email:</label>
<input type="email" id="email" name="email" required>
</div><br>
<div>
<label for="message">Message:</label>
<textarea id="message" name="message" required></textarea>
</div><br>
<button type="submit">Submit</button>
</form>
</section>
</main>
<footer>
<p>&copy; 2024 My Website</p>
</footer>
</body>
</html>
```

## **OUTPUT:**

# Welcome to My Website

- [Home](#)
- [About](#)
- [Contact](#)

## Introduction

This website is a showcase of HTML and CSS skills, featuring various types of content and layout options.

## Featured Section

Explore the other pages to learn more about me, view the gallery, or get in touch through the contact page.

© 2024 My Website

# About Me

- [Home](#)
- [About](#)
- [Contact](#)

## Who I Am

This section provides a brief background about me, my interests, and my professional experience.

### Hobbies

- Photography
- Coding
- Traveling

© 2024 My Website

# Contact Me

- [Home](#)
- [About](#)
- [Contact](#)

## Get in Touch

Name:

Email:

Message:

© 2024 My Website

# PRACTICAL 2

**Aim:** Using various Form tags for interactivity, authentication, date validation, etc.

## **CODE:**

### **1. Index.html**

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Interactive Form Example</title>
  <link rel="stylesheet" href="style.css">
  <script src="script.js" defer></script>
</head>
<body>

  <h1>Interactive Form with Validation</h1>

  <!-- Form with various input fields -->
  <form id="registrationForm" onsubmit="return validateForm()">
    <!-- Name Input -->
    <label for="name">Full Name:</label>
    <input type="text" id="name" name="name" required>

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

    <!-- Password Input -->
    <label for="password">Password:</label>
    <input type="password" id="password" name="password" required>

    <!-- Date of Birth Input -->
    <label for="dob">Date of Birth:</label>
    <input type="date" id="dob" name="dob" required>

    <!-- Gender Radio Buttons -->
    <label id="gender" for="gender">Gender:</label>
    <div class="gender-options">
      <label><input type="radio" name="gender" value="Male" required> Male</label>
      <label><input type="radio" name="gender" value="Female"> Female</label>
      <label><input type="radio" name="gender" value="Other"> Other</label>
    </div>

    <!-- Country Selection -->
    <label for="country">Country:</label>
    <select id="country" name="country" required>
      <option value="">Select your country</option>
```

```
<option value="USA">USA</option>
<option value="Canada">Canada</option>
<option value="UK">UK</option>
<option value="Australia">Australia</option>
</select>

<!-- Terms and Conditions Checkbox -->
<label id="checkbox">
  <input type="checkbox" id="terms" name="terms" required>
  I agree to the terms and conditions
</label>

<!-- Submit Button -->
<button type="submit">Register</button>
</form>

<p id="message"></p>

</body>
</html>
```

## 2. Style.css

```
body {
  font-family: Arial, sans-serif;
  display: flex;
  flex-direction: column;
  align-items: center;
}
h1 {
  color: #333;
}
```

```
form {
  width: 30%;
  margin: 20px;
  padding: 30px;
  border: 1px solid #ccc;
  border-radius: 8px;
}
```

```
label {
  display: block;
  margin-top: 20px;
  color: #555;
}
```

```
input, select, button {
  width: 95%;
  padding: 8px;
  margin-top: 5px;
  font-size: 16px;
```



```

}

.gender-options label {
  display: inline-block;
  width: 25%;
}

.gender-options {
  display: flex;
  align-items: center;
}

.gender-options input{
  width: 15%;
}
#checkbox input{
  width: 5%;
}

button {
  background-color: #4CAF50;
  color: white;
  border: none;
  cursor: pointer;
  margin-top: 15px;
}

button:hover {
  background-color: #45a049;
}

#message {
  color: red;
  font-weight: bold;
}

```

### 3. Script.js

```

function validateForm() {
  const name = document.getElementById("name").value;
  const email = document.getElementById("email").value;
  const password = document.getElementById("password").value;
  const dob = document.getElementById("dob").value;
  const terms = document.getElementById("terms").checked;
  const message = document.getElementById("message");

  // Clear previous message
  message.textContent = "";

  // Password length validation
  if (password.length < 6) {
    message.textContent = "Password must be at least 6 characters long.";
  }
}

```

```
        return false;
    }

    // Date of birth validation (age 18+)
    const dobDate = new Date(dob);
    const today = new Date();
    const age = today.getFullYear() - dobDate.getFullYear();
    const monthDifference = today.getMonth() - dobDate.getMonth();
    if (monthDifference < 0 || (monthDifference === 0 && today.getDate() < dobDate.getDate())) {
        age--;
    }

    if (age < 18) {
        message.textContent = "You must be at least 18 years old to register.";
        return false;
    }

    // Terms and conditions validation
    if (!terms) {
        message.textContent = "You must agree to the terms and conditions.";
        return false;
    }

    // Display success message if all validations pass
    message.textContent = "Registration successful!";
    message.style.color = "green";
    return false; // Prevent form submission for demonstration
}
```

## OUTPUT:

# Interactive Form with Validation

Full Name:

Email:

Password:

Date of Birth:

Gender:

☐ Male ☐ Female ☐ Other

Country:

☐ I agree to the terms and conditions

Register

# PRACTICAL 3

**Aim:** Using Semantic HTML tags associated with interactivity.

## **CODE:**

### **1. Index.html**

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

  <!-- Header -->
  <header>
    <h1>My Semantic HTML Webpage</h1>
    <p>Welcome to a page structured with semantic HTML elements for clarity and accessibility.</p>
  </header>

  <!-- Navigation -->
  <nav>
    <ul>
      <li><a href="#home">Home</a></li>
      <li><a href="#about">About</a></li>
      <li><a href="#articles">Articles</a></li>
      <li><a href="#contact">Contact</a></li>
    </ul>
  </nav>

  <!-- Main Content -->
  <main>
    <!-- About Section -->
    <section id="about">
      <h2>About Us</h2>
      <p>This section provides information about the purpose and mission of our website.</p>
    </section>

    <!-- Articles Section -->
    <section id="articles">
      <h2>Featured Articles</h2>

      <!-- Article 1 -->
      <article>
        <h3>Understanding Semantic HTML</h3>
        <p>Semantic HTML improves accessibility and SEO by clearly defining sections of a webpage.</p>
```

```

</article>

<!-- Article 2 -->
<article>
  <h3>Why Semantic Tags Matter</h3>
  <p>Using semantic tags helps screen readers and search engines interpret your content more
effectively.</p>
</article>
</section>

<!-- Sidebar -->
<aside>
  <h3>Related Links</h3>
  <ul>
    <li><a href="https://developer.mozilla.org/en-US/docs/Web/HTML">HTML
Documentation</a></li>
    <li><a href="https://www.w3schools.com/html/">HTML Tutorials</a></li>
  </ul>
</aside>
</main>

<!-- Footer -->
<footer>
  <p>&copy; 2024 My Semantic HTML Webpage. All rights reserved.</p>
  <p>Contact us at: <a href="mailto:info@example.com">info@example.com</a></p>
</footer>
</body>
</html>

```

## 2. Style.css

```

body {
  font-family: Arial, sans-serif;
  margin: 0;
  padding: 0;
}
header {
  background-color: #4CAF50;
  color: white;
  padding: 20px;
  text-align: center;
}
nav ul {
  list-style-type: none;
  background-color: #333;
  padding: 20px;
  text-align: center;
  margin: 0px;
}

nav ul li {
  display: inline;

```

```
    margin: 0 45px;
}
nav a {
    color: white;
    text-decoration: none;
}
main {
    display: flex;
    padding: 20px;
}
section{
    padding: 10px;
}
#about{
    width: 25vw;
}
#articles{
    width: 48vw;
}
aside {
    flex: 1;
    background-color: #f4f4f4;
    padding: 20px;
    margin-left: 20px;
}
footer {
    background-color: #333;
    color: white;
    text-align: center;
    padding: 10px;
}
```

## OUTPUT:

# My Semantic HTML Webpage

Welcome to a page structured with semantic HTML elements for clarity and accessibility.

[Home](#)[About](#)[Articles](#)[Contact](#)

### About Us

This section provides information about the purpose and mission of our website.

### Featured Articles

#### Understanding Semantic HTML

Semantic HTML improves accessibility and SEO by clearly defining sections of a webpage.

#### Why Semantic Tags Matter

Using semantic tags helps screen readers and search engines interpret your content more effectively.

### Related Links

- [HTML Documentation](#)
- [HTML Tutorials](#)

© 2024 My Semantic HTML Webpage. All rights reserved.

Contact us at: [info@example.com](mailto:info@example.com)

# PRACTICAL 4

**Aim:** Using DHTML tags in concern to client server application.

## **CODE:**

### **1. Index.html**

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>DHTML Example Page</title>
  <link rel="stylesheet" href="style.css">
  <script src="script.js" defer></script>
</head>
<body>

  <!-- Header Section -->
  <header>
    <h1 id="dynamicText">Welcome to My DHTML Page!</h1>
    <button onclick="changeTextColor()">Click to Change Color</button>
  </header>

  <!-- Main Content Section -->
  <main>
    <section>
      <h2>About DHTML</h2>
      <p>DHTML is a combination of HTML, CSS, and JavaScript that allows web pages to be more
interactive and dynamic.</p>
      <button onclick="toggleContent()">Show More</button>
      <div id="extraContent" style="display: none;">
        <p>This additional content is revealed using JavaScript. DHTML allows for changes to HTML
content, styles, and elements based on user interactions without reloading the page.</p>
      </div>
    </section>
    <section>
      <h2>Interactive Elements</h2>
      <p>Move your mouse over the box to see it change color!</p>
      <div id="colorBox" onmouseover="changeBoxColor()" onmouseout="resetBoxColor()">
        Hover over me!
      </div>
    </section>
  </main>

  <!-- Footer Section -->
  <footer>
    <p>&copy; 2024 DHTML Example Page</p>
  </footer>
</body>
</html>
```

## 2. Style.css

```
body {
  font-family: Arial, sans-serif;
  margin: 0;
  padding: 0;
  display: flex;
  flex-direction: column;
  align-items: center;
  text-align: center;
}

header {
  background-color:rgb(252, 252, 137);
  color: darkgreen;
  padding: 20px;
  width: 100%;
}

button {
  margin-top: 10px;
  padding: 10px 20px;
  cursor: pointer;
}

main {
  display: flex;
  flex-direction: row;
}

section{
  padding: 35px 35px;
  max-width: 55%;
}

#colorBox {
  width: 150px;
  height: 150px;
  background-color: #f4f4f4;
  margin: 20px auto;
  border: 2px solid #333;
  display: flex;
  align-items: center;
  justify-content: center;
  color: #333;
  font-weight: bold;
}

footer {
  background-color: #333;
  color: white;
  padding: 10px;
  width: 100%;
}
```



### 3. Script.js

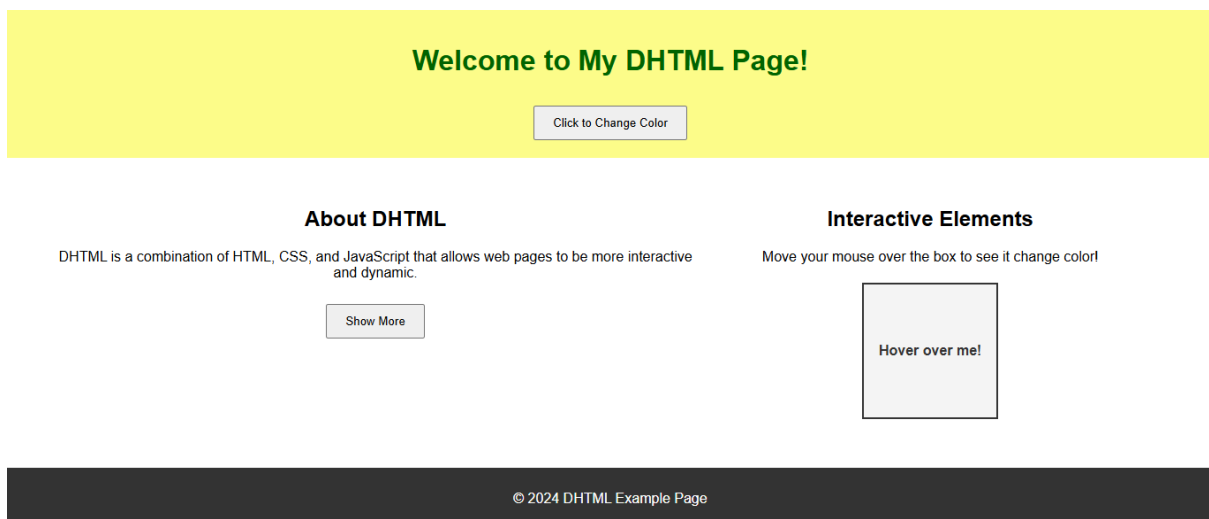
```
// Change the color of the text in the header on button click
function changeTextColor() {
    const headerText = document.getElementById("dynamicText");
    headerText.style.color = headerText.style.color === "red" ? "blue" : (headerText.style.color ===
"blue" ? "darkgreen" : "red");
}

// Toggle visibility of extra content in the "About DHTML" section
function toggleContent() {
    const extraContent = document.getElementById("extraContent");
    if (extraContent.style.display === "none") {
        extraContent.style.display = "block";
    } else {
        extraContent.style.display = "none";
    }
}

// Change box color on mouseover
function changeBoxColor() {
    document.getElementById("colorBox").style.backgroundColor = "crimson";
}

// Reset box color on mouseout
function resetBoxColor() {
    document.getElementById("colorBox").style.backgroundColor = "#f4f4f4";
}
```

### OUTPUT:



# PRACTICAL 5

**Aim:** Using JavaScript for dynamic web pages.

## **CODE:**

### **1. Index.html**

```
<!DOCTYPE html>
<html>
<head>
  <title>Todo List</title>
  <link rel="stylesheet" type="text/css" href="style.css">
</head>
<body>
  <div class="todo-container">
    <h1>Todo List</h1>
    <input type="text" id="taskInput" placeholder="Enter new task">
    <button id="addTaskButton">Add Task</button>
    <ul id="taskList"></ul>
  </div>

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

### **2. Style.css**

```
.todo-container {
  width: 400px;
  margin: 0 auto;
  padding: 20px;
  border: 1px solid #ccc;
  border-radius: 5px;
  background-color: #f5f5f5;
}

#taskInput {
  width: 70%;
  padding: 5px;
}

#addTaskButton {
  padding: 6px 10px;
}

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

```

li {
  display: flex;
  justify-content: space-between;
  padding: 5px;
  border-bottom: 1px solid #ccc;
}

button {
  margin-left: 10px;
}

```

### 3. Script.js

```

const taskInput = document.getElementById("taskInput");
const addTaskButton = document.getElementById("addTaskButton");
const taskList = document.getElementById("taskList");

```

```

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

```

```

addTaskButton.addEventListener("click", addTask);

```

```

function addTask() {
  const taskText = taskInput.value.trim();
  if (taskText) {
    tasks.push(taskText);
    saveTasks();
    displayTasks();
    taskInput.value = "";
  }
  else{
    alert("Please enter a task.");
  }
}

```

```

function displayTasks() {
  taskList.innerHTML = "";
  tasks.forEach((task, index) => {
    const li = document.createElement("li");
    li.innerHTML = `
      <span>${task}</span>
      <button onclick="editTask(${index})">Edit</button>
      <button onclick="deleteTask(${index})">Delete</button>
    `;
    taskList.appendChild(li);
  });
}

```

```

function editTask(index) {
  const newTaskText = prompt("Edit task:", tasks[index]);
  if (newTaskText !== null) {
    tasks[index] = newTaskText.trim();
  }
}

```

```

        saveTasks();
        displayTasks();
    }
}

function deleteTask(index) {
    if(confirm("Do you want to delete this task?")){
        tasks.splice(index, 1);
        saveTasks();
        displayTasks();
    }
}

function saveTasks() {
    localStorage.setItem("tasks", JSON.stringify(tasks));
}

```

## **OUTPUT:**

## Todo List

SPM FILE	<input type="button" value="Edit"/>	<input type="button" value="Delete"/>
JAVA FILE	<input type="button" value="Edit"/>	<input type="button" value="Delete"/>
AI-ML FILE	<input type="button" value="Edit"/>	<input type="button" value="Delete"/>
DBMD FILE	<input type="button" value="Edit"/>	<input type="button" value="Delete"/>
ASSIGNMENTS	<input type="button" value="Edit"/>	<input type="button" value="Delete"/>
WEB LAB FILE	<input type="button" value="Edit"/>	<input type="button" value="Delete"/>

# PRACTICAL 6

**Aim:** Utilize HTML/CSS and JavaScript frameworks (ReactJS, NextJS) to construct dynamic user interfaces.

## **CODE:**

### **1. App.js**

```
import React from 'react';
import './App.css';
import Calculator from './Calculator';

function App() {
  return (
    <div className="App">
      <h1><center>Simple Calculator</center></h1>
      <Calculator />
    </div>
  );
}

export default App;
```

### **2. Calculator.js**

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

const Calculator = () => {
  const [input, setInput] = useState("");
  const [result, setResult] = useState("");

  const handleButtonClick = (value) => {
    // Prevent multiple decimal points in the same number
    if (value === '.' && input.split(/[+\-*\^\/]/).pop().includes('.')) {
      return;
    }
    setInput((prevInput) => prevInput + value);
  };

  const calculateResult = () => {
    try {
      // Handle percentage calculation
      let processedInput = input.replace(/(\d+)\%/g, (match, number) => number / 100);
      setResult(eval(processedInput));
    } catch (error) {
      setResult('Error');
    }
  };

  const clearInput = () => {
    setInput("");
    setResult("");
  };
}
```

```

    setInput('');
    setResult('');
  };

  const deleteLastCharacter = () => {
    setInput((prevInput) => prevInput.slice(0, -1));
  };

  return (
    <div className="calculator">
      <div className="display">
        <div className="input">{input}</div>
        <div className="result">{result}</div>
      </div>
      <div className="buttons">
        <button onClick={() => handleButtonClick('1')}>1</button>
        <button onClick={() => handleButtonClick('2')}>2</button>
        <button onClick={() => handleButtonClick('3')}>3</button>
        <button onClick={() => handleButtonClick('+')}>+</button>
        <button onClick={() => handleButtonClick('4')}>4</button>
        <button onClick={() => handleButtonClick('5')}>5</button>
        <button onClick={() => handleButtonClick('6')}>6</button>
        <button onClick={() => handleButtonClick('-')}>-</button>
        <button onClick={() => handleButtonClick('7')}>7</button>
        <button onClick={() => handleButtonClick('8')}>8</button>
        <button onClick={() => handleButtonClick('9')}>9</button>
        <button onClick={() => handleButtonClick('*')}>*</button>
        <button onClick={() => handleButtonClick('00')}>00</button>
        <button onClick={() => handleButtonClick('0')}>0</button>
        <button onClick={() => handleButtonClick('.')}>.</button>
        <button onClick={() => handleButtonClick('/')}>/</button>
        <button onClick={deleteLastCharacter}>DEL</button>
        <button onClick={clearInput}>C</button>
        <button onClick={calculateResult}>=</button>
        <button onClick={() => handleButtonClick('%')}>%</button>
      </div>
    </div>
  );
};

export default Calculator;

```

### 3. App.css

```

.calculator {
  width: 320px;
  margin: 0 auto;
  padding: 20px;
  border: 1px solid #ccc;
  border-radius: 10px;
  box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);
}

```

```
.display {
  border: 1px solid #ddd;
  padding: 10px;
  background-color: #f9f9f9;
  border-radius: 5px;
  text-align: right;
}

.input {
  font-size: 24px;
}

.result {
  font-size: 32px;
  color: #333;
}

.buttons {
  display: grid;
  grid-template-columns: repeat(4, 1fr);
  gap: 10px;
  margin-top: 20px;
}

button {
  padding: 20px;
  font-size: 18px;
  border: none;
  border-radius: 5px;
  background-color: #007BFF;
  color: white;
  cursor: pointer;
}

button:hover {
  background-color: #0056b3;
}
```

**OUTPUT:**

## Simple Calculator

3\*10+8/2-1.5

32.5

1	2	3	+
4	5	6	-
7	8	9	*
00	0	.	/
DEL	C	=	%



# PRACTICAL 7

**Aim:** Create various databases using SQL/MongoDB/or other to show interactivity.

## **CODE:**

### **Database interactivity using Backend**

#### **1. User.java (Entity class)**

```
package com.restaurant.demo.entity;

import jakarta.persistence.*;
import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;
import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.authority.SimpleGrantedAuthority;
import org.springframework.security.core.userdetails.UserDetails;

import java.util.ArrayList;
import java.util.Collection;
import java.util.List;

@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor

@Entity
@Table(name = "User_details",
        uniqueConstraints = {
            @UniqueConstraint(
                name = "unique_details",
                columnNames = "username")
        }
)

public class User implements UserDetails {
    @Id
    @GeneratedValue(strategy= GenerationType.AUTO)
    private Long user_id;
    @Column(name="username",nullable = false)
    private String username;

    @Column(name="password",nullable = false)
    private String password;

    @Column(name="phone",nullable = false)
```

```

private String phone;

@Column(name="email",nullable = false)
private String email;

@Column(name = "name")
private String name;
@Column(name = "Address")
private String address;
@Column(name="type")
private String type;
@Column(name = "Gender")
private String gender;
@Column(name = "cardId")
private String cardId;

@OneToMany(cascade = CascadeType.ALL,orphanRemoval = true)
@JoinColumn(name = "owner_id",referencedColumnName = "user_id")
private List<Restaurant> restaurants;

@Override
public Collection<? extends GrantedAuthority> getAuthorities() {
    SimpleGrantedAuthority simpleGrantedAuthority = new SimpleGrantedAuthority(type);
    return List.of(simpleGrantedAuthority);
}

@Override
public String getPassword() {
    return password;
}

@Override
public String getUsername() {
    return username;
}
}

```

## 2. Restaurant.java (Entity class)

```

package com.restaurant.demo.entity;

import jakarta.persistence.*;
import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;
import org.springframework.boot.autoconfigure.web.WebProperties;

import java.util.List;

@Getter
@Setter

```

@NoArgsConstructor

@AllArgsConstructor

@Entity

@Table(name = "restaurant")

```
public class Restaurant{
    @Id
    @GeneratedValue(strategy= GenerationType.AUTO)
    private Long rest_id;

    @Column(name = "restaurant_name", nullable = false)
    private String rest_name;

    @Column(name = "location")
    private String location;

    @Column(name = "foodType")
    private String foodtype;

    @Column(name = "rest_rating")
    private int rest_rating;

    @OneToMany(cascade = CascadeType.ALL, orphanRemoval = true)
    @JoinColumn(name = "restaurant_id", referencedColumnName = "rest_id")
    private List<FoodItems> items;
}
```

### 3. RestaurantService.java (Service class)

```
package com.restaurant.demo.service.impl;

import com.restaurant.demo.dto.RestaurantDto;
import com.restaurant.demo.dto.UserDto;
import com.restaurant.demo.entity.Restaurant;
import com.restaurant.demo.entity.User;
import com.restaurant.demo.exception.ResourceNotFoundException;
import com.restaurant.demo.mapper.RestaurantMapper;
import com.restaurant.demo.mapper.UserMapper;
import com.restaurant.demo.repository.RestaurantRepository;
import com.restaurant.demo.repository.UserRepository;
import com.restaurant.demo.service.RestaurantService;
import lombok.AllArgsConstructor;
import org.springframework.stereotype.Service;
import org.springframework.web.bind.annotation.RequestBody;

import java.util.List;
import java.util.Optional;
import java.util.stream.Collectors;

@Service
@AllArgsConstructor
```

```

public class RestaurantServiceImpl implements RestaurantService {

    private UserRepository userRepository;

    private RestaurantRepository restaurantRepository;
    @Override
    public List<RestaurantDto> addRestaurant(String username, RestaurantDto restaurantDto) {
        User user = userRepository.findByUsername(username);
        if(user==null){
            throw new ResourceNotFoundException("User not present");
        }
        else {
            Restaurant restaurant = RestaurantMapper.mapToRestaurant(restaurantDto);
            user.getRestaurants().add(restaurant);
            userRepository.save(user);

            return (UserMapper.mapToUserDto(user).getRestaurants()).stream().map((rest)-
>RestaurantMapper.mapToRestaurantDto(rest)).collect(Collectors.toList());
        }
    }

    public List<RestaurantDto> getAllRestaurant(){
        List<Restaurant> restaurantList = restaurantRepository.findAll();
        return restaurantList.stream().map((restaurant)-
>RestaurantMapper.mapToRestaurantDto(restaurant)).collect(Collectors.toList());
    }

    public List<RestaurantDto> getAllRestaurantByUsername(String username){
        List<Restaurant> restaurantList = restaurantRepository.findAllByOwnername(username);
        return restaurantList.stream().map((restaurant)-
>RestaurantMapper.mapToRestaurantDto(restaurant)).collect(Collectors.toList());
    }

    public List<RestaurantDto> searchAllRestaurantByName(String username,String name){
        List<Restaurant> restaurantList =
restaurantRepository.findAllRestaurantByName(username,name);
        return restaurantList.stream().map((restaurant)-
>RestaurantMapper.mapToRestaurantDto(restaurant)).collect(Collectors.toList());
    }

    public RestaurantDto updateRestaurantbyID(RestaurantDto details){
        Restaurant restaurant = restaurantRepository.findById(details.getRest_id()).orElseThrow(()-> {
            return new ResourceNotFoundException("Restaurant does not exist with given ID: " +
details.getRest_id());
        });
        restaurant.setRest_name(details.getRest_name());
        Restaurant updatedRestaurant = restaurantRepository.save(restaurant);
        return RestaurantMapper.mapToRestaurantDto(restaurant);
    }

    public List<RestaurantDto> deleteRestaurantsbyID(String username,List<Long> restaurantIds){
        User user = userRepository.findByUsername(username);
        if(user==null){

```

```

        throw new ResourceNotFoundException("User not present");
    }
    else {
        List<Restaurant> restaurants = restaurantRepository.findAllById(restaurantIds);
        for (Restaurant restaurant : restaurants) {
            user.getRestaurants().remove(restaurant);
            restaurantRepository.delete(restaurant);
            userRepository.save(user);
        }
        return (UserMapper.mapToUserDto(user).getRestaurants()).stream().map((rest)-
>RestaurantMapper.mapToRestaurantDto(rest)).collect(Collectors.toList());
    }
}

```

#### 4. RestaurantRepository.java (Repository class)

```
package com.restaurant.demo.repository;
```

```

import com.restaurant.demo.dto.RestaurantDto;
import com.restaurant.demo.entity.Restaurant;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;

```

```
import java.util.List;
```

```

public interface RestaurantRepository extends JpaRepository<Restaurant, Long>{
    @Query("Select r from User u INNER JOIN u.restaurants r where u.username = :n")
    public List<Restaurant> findAllByOwnername(@Param("n") String username);

```

```

    @Query("Select r from User u INNER JOIN u.restaurants r where u.username = :n and r.rest_name
LIKE %:p%")
    public List<Restaurant> findAllRestaurantByName(@Param("n") String username,@Param("p")
String name);
}

```

#### OUTPUT:

```
mysql> select * from restaurant;
```

rest_id	food_type	location	restaurant_name	rest_rating	owner_id
3	Veg	Chennai	HealthyFOODS_3	0	3
4	Non-Veg	Mumbai	HealthyFOODS_4	0	3
303	Veg	Agra	HealthyFOODS_2	0	4
452	Veg	Delhi	HealthyFOODS_1	0	3
453	Both	Agra	HealthyFOODS_2	0	3
454	Veg	hyderabad	My Kitchen1	0	3
455	Veg	Delhi	Chat centre	0	3

```

7 rows in set (0.01 sec)

```

Restro-Maker
Home
About
Restaurants
Features

## Your Restaurant List

**HealthyFOODS\_3**  
Chennai

Some quick example text to build on the card title and make up the bulk of the card's content.

View
Edit

**HealthyFOODS\_4**  
Mumbai

Some quick example text to build on the card title and make up the bulk of the card's content.

View
Edit

**HealthyFOODS\_1**  
Delhi

Some quick example text to build on the card title and make up the bulk of the card's content.

View
Edit

**HealthyFOODS\_2**  
Agra

Some quick example text to build on the card title and make up the bulk of the card's content.

View
Edit

**My Kitchen1**  
hyderabad

Some quick example text to build on the card title and make up the bulk of the card's content.

View
Edit

**Chat centre**  
Delhi

Some quick example text to build on the card title and make up the bulk of the card's content.

View
Edit

Are you sure you want to remove the selected restaurants?

Yes, Remove
No, Keep them

Add Restaurant

Confirm Removal

## Table after performing delete operation

```
mysql> select * from restaurant;
```

rest_id	food_type	location	restaurant_name	rest_rating	owner_id
3	Veg	Chennai	HealthyFOODS_3	0	3
4	Non-Veg	Mumbai	HealthyFOODS_4	0	3
303	Veg	Agra	HealthyFOODS_2	0	4
453	Both	Agra	HealthyFOODS_2	0	3
454	Veg	hyderabad	My Kitchen1	0	3
455	Veg	Delhi	Chat centre	0	3

```
6 rows in set (0.00 sec)
```

# PRACTICAL 8

**Aim:** Perform CRUD operations using React JS as frontend technology and Node JS as backend technology showing database interactivity using any database.

## **CODE:**

### **FRONTEND**

#### **1. EmployeeForm.js**

```
import React from "react";
import { useForm } from "react-hook-form";
import { useNavigate } from "react-router-dom";

const EmployeeForm = () => {
  const { register, handleSubmit } = useForm();
  const navigate = useNavigate();

  const createEmployee = async (data) => {

    const savedUserResponse = await fetch(
      `${process.env.REACT_APP_BASE_URL}/createUser`,
      {
        method: "POST",
        headers: {
          "Content-Type": "application/json",
        },
        body: JSON.stringify({ ...data }),
      }
    );
    console.log("FORM RESPONSE.....", savedUserResponse);
    navigate("/")
  };

  return (
    <div>
      <form onSubmit={handleSubmit(createEmployee)} className="mt-8">
        <div className="space-y-5">
          <div>
            <label htmlFor="name"
              className="text-base font-medium text-gray-900 dark:text-gray-200"
            >
              {" "}
              Employee Name{" "}
            </label>
            <div className="mt-2.5">
              <input
                className="flex h-10 w-full rounded-md border border-gray-300 bg-transparent py-2 px-3 text-sm placeholder:text-gray-400 focus:outline-none focus:ring-1 focus:ring-gray-400 focus:ring-"
              />
            </div>
          </div>
        </div>
      </form>
    </div>
  );
};
```

offset-1 disabled:cursor-not-allowed disabled:opacity-50 dark:border-gray-700 dark:text-gray-50  
dark:focus:ring-gray-400 dark:focus:ring-offset-gray-900"

```
    type="text"
    placeholder="Enter You Full Name"
    {...register("name")}
  ></input>
</div>
</div>
```

```
<div>
  <label
    htmlFor="email"
    className="text-base font-medium text-gray-900 dark:text-gray-200"
  >
    {" "}
    Employee Email Id{" "}
  </label>
  <div className="mt-2.5">
    <input
      className="flex h-10 w-full rounded-md border border-gray-300 bg-transparent py-2 px-3
text-sm placeholder:text-gray-400 focus:outline-none focus:ring-1 focus:ring-gray-400 focus:ring-
offset-1 disabled:cursor-not-allowed disabled:opacity-50 dark:border-gray-700 dark:text-gray-50
dark:focus:ring-gray-400 dark:focus:ring-offset-gray-900"
      type="email"
      placeholder="Enter Your Email"
      {...register("email")}
    ></input>
  </div>
</div>
```

```
<div>
  <label
    htmlFor="title"
    className="text-base font-medium text-gray-900 dark:text-gray-200"
  >
    {" "}
    Employee Title{" "}
  </label>
  <div className="mt-2.5">
    <input
      className="flex h-10 w-full rounded-md border border-gray-300 bg-transparent py-2 px-3
text-sm placeholder:text-gray-400 focus:outline-none focus:ring-1 focus:ring-gray-400 focus:ring-
offset-1 disabled:cursor-not-allowed disabled:opacity-50 dark:border-gray-700 dark:text-gray-50
dark:focus:ring-gray-400 dark:focus:ring-offset-gray-900"
      type="text"
      placeholder="Enter Your Employee Title"
      {...register("title")}
    ></input>
  </div>
</div>
```



```

<div>
  <label
    htmlFor="department"
    className="text-base font-medium text-gray-900 dark:text-gray-200"
  >
    {" "}
    Employee Department{" "}
  </label>
  <div className="mt-2.5">
    <input
      className="flex h-10 w-full rounded-md border border-gray-300 bg-transparent py-2 px-3
text-sm placeholder:text-gray-400 focus:outline-none focus:ring-1 focus:ring-gray-400 focus:ring-
offset-1 disabled:cursor-not-allowed disabled:opacity-50 dark:border-gray-700 dark:text-gray-50
dark:focus:ring-gray-400 dark:focus:ring-offset-gray-900"
      type="text"
      placeholder="Enter Your Employee Department"
      {...register("department")}
    ></input>
  </div>
</div>

<div>
  <label
    htmlFor="role"
    className="text-base font-medium text-gray-900 dark:text-gray-200"
  >
    {" "}
    Employee Role{" "}
  </label>
  <div className="mt-2.5">
    <input
      className="flex h-10 w-full rounded-md border border-gray-300 bg-transparent py-2 px-3
text-sm placeholder:text-gray-400 focus:outline-none focus:ring-1 focus:ring-gray-400 focus:ring-
offset-1 disabled:cursor-not-allowed disabled:opacity-50 dark:border-gray-700 dark:text-gray-50
dark:focus:ring-gray-400 dark:focus:ring-offset-gray-900"
      type="text"
      placeholder="Enter Your Employee Role"
      {...register("role")}
    ></input>
  </div>
</div>

<div>
  <button
    type="submit"
    className="inline-flex w-full items-center justify-center rounded-md bg-indigo-600 px-3.5
py-2.5 text-base font-semibold leading-7 text-white hover:bg-indigo-500"
  >
    Create Employeee
  <svg
    xmlns="http://www.w3.org/2000/svg"

```

```

        fill="none"
        viewBox="0 0 24 24"
        strokeWidth={1.5}
        stroke="currentColor"
        className="ml-2 h-4 w-4"
      >
        <path
          strokeLinecap="round"
          strokeLinejoin="round"
          d="M4.5 12h15m0 0l-6.75-6.75M19.5 12l-6.75 6.75"
        />
      </svg>
    </button>
  </div>
</div>
</form>
</div>
);
};

```

```
export default EmployeeForm;
```

## 2. HomePage.js

```

import React, { useEffect, useState } from "react";
import { Link } from "react-router-dom";

const HomePage = () => {
  const [empData, setEmpData] = useState();

  const getAllData = async () => {
    try {
      const getPeople = await fetch(
        `${process.env.REACT_APP_BASE_URL}/getAllUsers`,
        {
          method: "GET",
          headers: {
            "Content-Type": "application/json",
          },
        }
      );
    }
  };

  const res = await getPeople.json();
  setEmpData(res);
} catch (error) {
  console.log(error);
}
};

useEffect(() => {
  getAllData();

```



```

        scope="col"
        className="px-4 py-3.5 text-sm font-normal text-left rtl:text-right text-gray-500
dark:text-gray-400"
      >
        Role
      </th>
    </tr>
  </thead>

  <tbody className="bg-white divide-y divide-gray-200 dark:divide-gray-700 dark:bg-gray-
900">
    {empData?.data.map((person) => (
      <tr key={person.name}>
        <td className="py-4 px-4 whitespace-nowrap">
          <div className="flex items-center">
            <div className="flex-shrink-0 h-10 w-10">
              <img
                className="h-10 w-10 rounded-full object-cover"
                src={person.image}
                alt=""
              />
            </div>
            <div className="ml-4">
              <div className="text-sm font-medium text-gray-900 dark:text-white">
                {person.name}
              </div>
              <div className="text-sm text-gray-500 dark:text-gray-300">
                {person.email}
              </div>
            </div>
          </div>
        </td>
        <td className="px-12 py-4 whitespace-nowrap">
          <div className="text-sm text-gray-900 dark:text-white">
            {person.title}
          </div>
          <div className="text-sm text-gray-500 dark:text-gray-300">
            {person.department}
          </div>
        </td>
        <td className="px-4 py-4 whitespace-nowrap text-sm text-gray-500 dark:text-gray-
300">
          {person.role}
        </td>
      </tr>
    )})
  </tbody>
</table>
</div>
</div>

```

```

        </div>
      </div>
    </section>
  </>
);
};

export default HomePage;

```

### 3. App.js

```

import { Route, Routes } from "react-router-dom";
import HomePage from "../pages/HomePage";
import CreateEmployeePage from "../pages/CreateEmployeePage";

function App() {
  return (
    <div>
      <Routes>
        <Route path="/" element={<HomePage />} />
        <Route path="/addemployee" element={<CreateEmployeePage />} />
      </Routes>
    </div>
  );
}

export default App;

```

## BACKEND

### 1. Database.js

```

const mongoose = require("mongoose");
require("dotenv");

const dbConnect = () => {
  mongoose
    .connect(process.env.DATABASE_URL, {
      useNewUrlParser: true,
      useUnifiedTopology: true,
    })
    .then(() => console.log("DB CONNECTION SUCCESS"))
    .catch((err) => {
      console.log(`DB CONNECTION ISSUES`);
      console.error(err.message);
      process.exit(1);
    });
};

module.exports = dbConnect;

```

## 2. Controller (CreateUser.js)

```
const User = require("../models/User");

exports.createUser = async (req, res) => {
  try {
    console.log("req body", req.body);
    const { name, email, title, department, role } = req.body;
    if (!name || !email || !title || !role || !department) {
      console.log("not all fields...");
      return res.status(400).json({
        status: 400,
        message: "Please fill all fields",
      });
    }
    const user = await User.create({
      name,
      email,
      title,
      department,
      role,
      image: `https://api.dicebear.com/5.x/initials/svg?seed=${name}`,
    });
    return res.status(200).json({
      status: 201,
      message: "User created successfully",
      data: user,
    });
  } catch (error) {
    console.log("error", error);
    return res.status(500).json({
      status: 500,
      message: error.message,
    });
  }
};
```

## 3. Controller (getUser.js)

```
const User = require("../models/User");
exports.getUser = async (req, res) => {
  try {
    const userData = await User.find({});
    res.json({ success: true, data: userData });
  } catch (error) {
    res.status(500).json({ success: false, error: error });
  }
};
```

## 4. Model (User.js)

```
const mongoose = require("mongoose");

const userSchema = new mongoose.Schema({
  name: {
    type: String,
    required: true,
  },
  email: {
    type: String,
    required: true,
    unique: true,
  },
  title: {
    type: String,
    required: true,
  },
  department: {
    type: String,
    required: true,
    maxLength: 20,
  },
  role: {
    type: String,
    required: true,
  },
  image: {
    type: String,
    required: true,
  },
});

module.exports = mongoose.model("User", userSchema);
```

## 5. Routes (User.js)

```
const express = require("express");
const router = express.Router();
const { createUser } = require("../controller/createUser");
const { getUser } = require("../controller/getUsers");
router.post("/createUser", createUser);
router.get("/getAllUsers", getUser);

module.exports = router;
```

## OUTPUT:

### Home Page

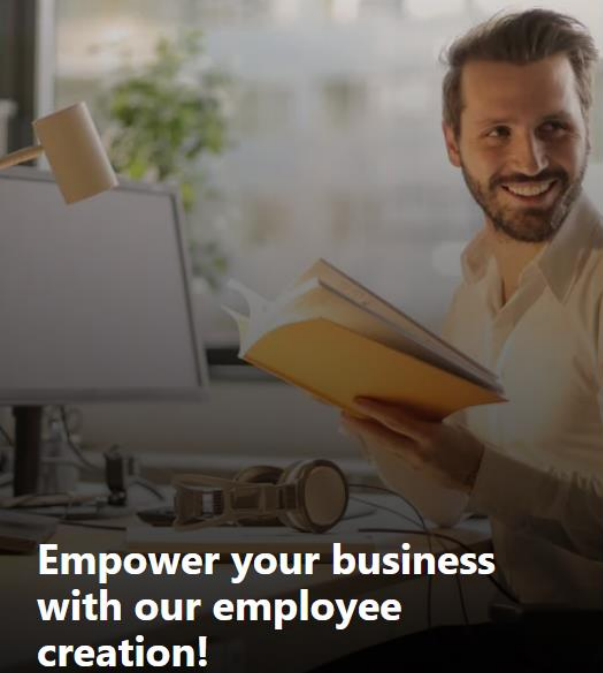
#### Employees

This is a list of all employees. You can add new employees, edit or delete existing ones.

[Add Employee](#)

Employee	Title	Role
<div>AS</div> <div>Anish Singhal</div> <div>anish.singhal@yahoo.in</div>	Junior Developer Market Research	Analyst
<div>JW</div> <div>John Wick</div> <div>wick.john@continental.us</div>	Project Manager Security	SDE 3

### Employee Form Page



**Empower your business  
with our employee  
creation!**

[← Back to all Employee List](#)  
  

Employee Name

Employee Email Id

Employee Title

Employee Department

Employee Role

[Create Employee →](#)



# PRACTICAL 9

**Aim:** Develop robust back-end systems using Node.js.

## **CODE:**

### **1. Index.ejs**

```
<!DOCTYPE html>
<html lang="en">
<head>
  <style>
    body {
      font-family: cursive;
      display: flex;
      align-items: center;
      flex-direction: column;
      margin: 0;
      padding: 20px;
      background-color: #f9f9f9;
    }
    h1 {
      font-size: 2rem;
      color: #333;
      margin-bottom: 20px;
    }
    form {
      display: flex;
      align-items: center;
      gap: 10px;
      margin-bottom: 20px;
    }
    label {
      font-weight: bold;
    }
    input[type="text"] {
      padding: 8px;
      width: 300px;
      border: 1px solid #ddd;
      border-radius: 4px;
    }
    button {
      padding: 8px 16px;
      background-color: #4CAF50;
      color: #fff;
      border: none;
      border-radius: 4px;
      cursor: pointer;
    }
    button:hover {
      background-color: #45a049;
```

```

    }
    table {
        width: 100%;
        max-width: 600px;
        border-collapse: collapse;
        margin-top: 20px;
        background-color: #fff;
        border: 1px solid #ddd;
        border-radius: 8px;
        overflow: hidden;
        box-shadow: 0 4px 8px rgba(0,0,0,0.1);
    }
    th, td {
        padding: 12px 15px;
        text-align: left;
        border-bottom: 1px solid #ddd;
    }
    th {
        background-color: #4CAF50;
        color: white;
        font-weight: bold;
    }
    tr:hover {
        background-color: #f1f1f1;
    }
    td:nth-child(4) {
        text-align: center;
    }
</style>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Home Page</title>
</head>

<body>
<h1>URL Shortener</h1>
<% if (locals.id) { %>
    <p>URL Generated: http://localhost:8001/url/<%= id %></p>
<% } %>
<div>
    <form method="POST" action="/url">
        <label>Enter Your Original URL</label>
        <input
            type="text"
            name="url"
            placeholder="https://example.com"
        />
        <button type="submit">Generate</button>
    </form>
</div>
<div>

```

```

<% if(locals.urls){ %>
<table>
  <thead>
    <tr>
      <th>S.No</th>
      <th>ShortID</th>
      <th>Redirect</th>
      <th>Clicks</th>
    </tr>
  </thead>
  <tbody>
    <% urls.forEach( (url,index) => { %>
      <tr>
        <td><%= index + 1 %></td>
        <td><%= url.shortId %></td>
        <td><%= url.redirectURL %></td>
        <td><%= url.visitHistory.length %></td>
      </tr>
    <% } ) %>
  </tbody>
</table>
<% } %>
</div>
</body>
</html>

```

## 2. database model (model > url.js )

```

const mongoose = require('mongoose');

const urlSchema = new mongoose.Schema({
  shortId:{
    type:String,
    require:true,
    unique:true,
  },
  redirectURL: {
    type: String,
    require: true,
  },
  visitHistory: [{
    timestamp: {type:Number}
  }],
},{timestamps:true}
});

```

## 3. Controller (controller > url.js )

```

const URL = mongoose.model('url' , urlSchema);
module.exports = URL;

const shortid = require("shortid");

```

```

const URL = require('../models/url.js');

async function handleGenerateNewShortURL(req, res){
  const body = req.body;
  if(!body.url) return res.status(400).json({error: 'url is required'})

  const shortID = shortid();

  await URL.create({
    shortId: shortID,
    redirectURL: body.url,
    visitHistory: [],
  });
  return res.render('home' , {
    id: shortID,
  });
}

async function handleGetAnalytics(req , res){
  const shortId = req.params.shortId;
  const result = await URL.findOne({shortId});
  return res.json({totalClicks: result.visitHistory.length , analytics: result.visitHistory})
}

module.exports = {
  handleGenerateNewShortURL,
  handleGetAnalytics,
}

```

#### 4. Router (routes > staticRouter.js )

```

const express = require("express");
const router = express.Router();
const URL = require("../models/url")

router.get('/', async(req , res) => {

  const allurls = await URL.find({});
  return res.render('home' , {
    urls: allurls
  })
})

module.exports = router;

```

#### 5. Router (routes > url.js )

```

const express = require("express");
const router = express.Router();
const { handleGenerateNewShortURL , handleGetAnalytics } = require('../controllers/url')

router.post('/', handleGenerateNewShortURL);
router.get('/analytics/:shortId' , handleGetAnalytics);
module.exports = router;

```

## 6. Index.js

```
const express = require('express')
const app = express();
const path = require('path')
const PORT = 8001;
const urlRoute = require('./routes/url');

const {connectToMongoDB} = require("./connect");
const URL = require("./models/url");
app.set("view engine" , "ejs");
app.set('views' , path.resolve("./views"));
const staticRoute = require("./routes/staticRouter");

app.use(express.json());
app.use(express.urlencoded({extended:false}));

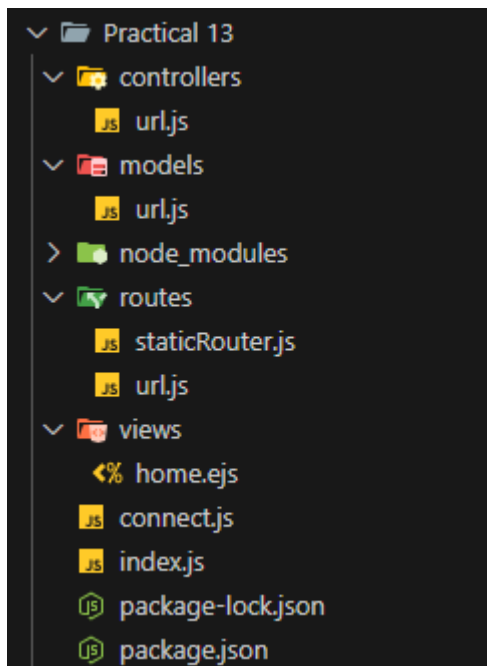
connectToMongoDB('mongodb://localhost:27017/short-url')
.then(()=>console.log('Mongo DB connected'));

app.use("/url" , urlRoute);
app.get('/url/:shortID' , async (req ,res)=>{
  const shortId = req.params.shortID;
  const entry = await URL.findOneAndUpdate(
    {
      shortId
    },
    {
      $push: {
        visitHistory: {
          timestamp: Date.now(),
        },
      },
    }
  );
  return res.redirect(entry.redirectURL);
});
app.use('/', staticRoute);
app.listen(PORT , () => console.log(`Server Starrted at PORT at ${PORT}`))
```

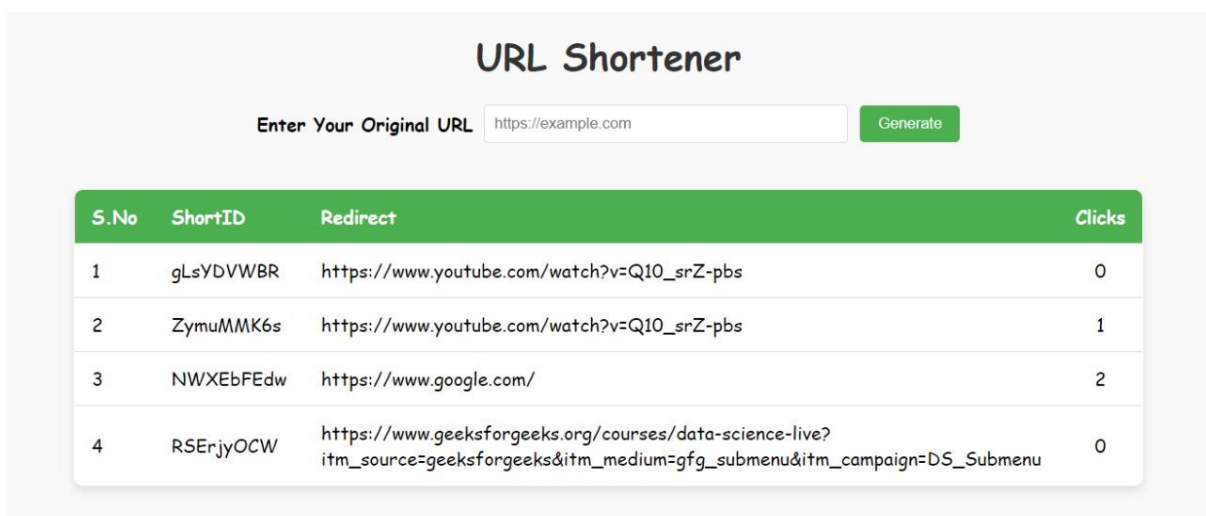
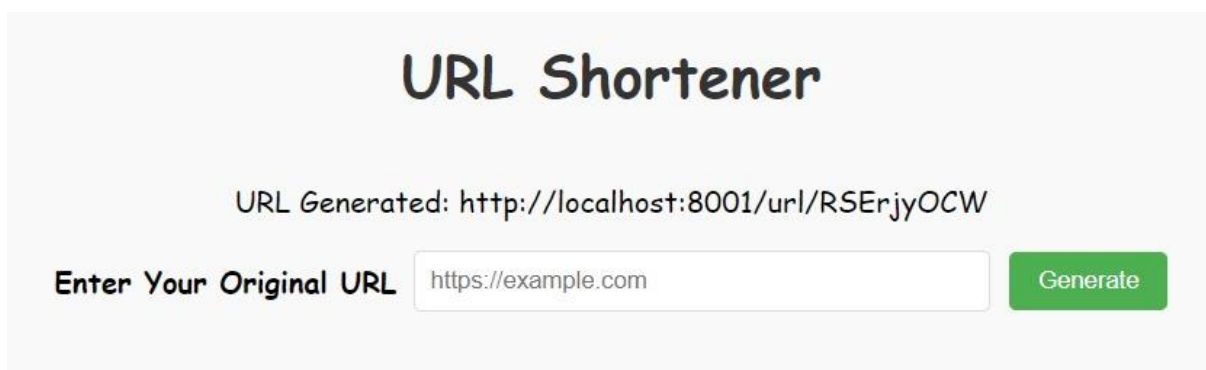
## 7. Connect.js

```
const mongoose = require('mongoose');
mongoose.set("strictQuery" , true);
async function connectToMongoDB(url){
  return mongoose.connect(url);
}
module.exports = {
  connectToMongoDB
}
```

## File Structure:



## OUTPUT:



# PRACTICAL 10

**Aim:** Showing database interactivity using PHP/Python/or any current technology used in web industry.

## **CODE:**

### **1. Index.html**

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Weather App</title>
  <link rel="stylesheet" href="../static/style.css">
</head>
<body>
  <div class="container">
    <h1>Weather App</h1>
    <input type="text" id="cityInput" placeholder="Enter city name">
    <button onclick="getWeather()">Get Weather</button>
    <div id="weatherResult"></div>
  </div>
  <script src="../static/script.js"></script>
</body>
</html>
```

### **2. Style.css**

```
body {
  font-family: Arial, sans-serif;
  display: flex;
  justify-content: center;
  align-items: center;
  height: 100vh;
  margin: 0;
  background-color: #f4f4f9;
}

.container {
  text-align: center;
  width: 300px;
}

h1 {
  color: #333;
}

#cityInput {
  width: 80%;
```

```

padding: 10px;
margin-top: 10px;
margin-bottom: 10px;
}

button {
padding: 10px 15px;
background-color: #007bff;
color: white;
border: none;
cursor: pointer;
border-radius: 5px;
}

button:hover {
background-color: #0056b3;
}

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

```

### 3. Script.js

```

function getWeather() {
const city = document.getElementById("cityInput").value;
fetch("/get_weather", {
method: "POST",
headers: {
"Content-Type": "application/json"
},
body: JSON.stringify({ city: city })
})
.then(response => response.json())
.then(data => {
if (data.error) {
document.getElementById("weatherResult").innerText = data.error;
} else {
document.getElementById("weatherResult").innerHTML = `
<p><strong>City:</strong> ${data.city}</p>
<p><strong>Temperature:</strong> ${data.temperature}°C</p>
<p><strong>Feels Like:</strong> ${data.feels_like}°C</p>
<p><strong>Weather:</strong> ${data.weather}</p>
<p><strong>Wind Speed:</strong> ${data.wind_speed} m/s</p>
`;
}
})
.catch(error => {
console.error("Error:", error);
document.getElementById("weatherResult").innerText = "An error occurred. Please try again.";
});
}

```



```
}
```

## 4. app.py (For Backend)

```
from flask import Flask, render_template, request, jsonify
import requests

app = Flask(__name__)

API_KEY = "e4c2903df887d05c7ce3dc39fb60a143"
BASE_URL = "http://api.openweathermap.org/data/2.5/weather"

@app.route('/')
def index():
    return render_template("index.html")

@app.route('/get_weather', methods=['POST'])
def get_weather():
    data = request.get_json()
    city = data.get("city")
    if not city:
        return jsonify({"error": "City name is required"}), 400

    try:
        params = {'q': city, 'appid': API_KEY, 'units': 'metric'}
        response = requests.get(BASE_URL, params=params)
        weather_data = response.json()

        if weather_data['cod'] == 200:
            return jsonify({
                "city": city,
                "temperature": weather_data['main']['temp'],
                "feels_like": weather_data['main']['feels_like'],
                "weather": weather_data['weather'][0]['description'],
                "wind_speed": weather_data['wind']['speed']
            })
        else:
            return jsonify({"error": "City not found"}), 404
    except Exception as e:
        return jsonify({"error": str(e)}), 500

if __name__ == '__main__':
    app.run(debug=True)
```

## OUTPUT:

# Weather App

Get Weather

City name is required

# Weather App

Get Weather

City: Delhi

Temperature: 23.05°C

Feels Like: 23.45°C

Weather: haze

Wind Speed: 0 m/s

# PRACTICAL 11

**Aim:** Any current web industry relevant example of database usage and interactivity using any suitable backend technology.

## **CODE:**

### **FRONTEND**

#### **1. App.js**

```
import './App.css';
import React from 'react';
import {HashRouter,Routes,Route,} from "react-router-dom";
import Homepage from './restaurant components/HomePage';
import Login from './restaurant components/Login';
import RestaurantPage from './restaurant components/RestuarantList';
import SignUp from './restaurant components/SignUp';
import ForgotPassword from './restaurant components/ForgotPassword';
import Menu from './restaurant components/Menu';
import NavState from './restaurant components/Context/NavState';
import About from './restaurant components/About';
import ViewDetails from './restaurant components/View details';
import ScrollToTop from './restaurant components/ScrollToTop';
import UserPage from './restaurant components/Userpage';
import Order from './restaurant components/Order';
import Profile from './restaurant components/Profile';

function App() {

  return (
    <>
    <NavState>
      <HashRouter>
        <ScrollToTop/>
        <Routes>
          <Route exact path = "/" element={<Homepage/>}></Route>
          <Route exact path = '/login' element={<Login/>}></Route>
          <Route exact path = '/signup' element={<SignUp/>}></Route>
          <Route exact path = '/password' element={<ForgotPassword/>}></Route>
          <Route exact path = '/about' element={<About/>}></Route>
          <Route exact path = '/restaurant' element={<RestaurantPage/>}></Route>
          <Route exact path = '/restaurant/menu' element={<Menu/>}/>
          <Route exact path = '/restaurant/view' element={<ViewDetails/>}/>
          <Route exact path = '/orderpage' element={<UserPage/>}/>
          <Route exact path = '/order' element={<Order/>}/>
          <Route exact path = '/profile' element={<Profile/>}/>
        </Routes>
      </HashRouter>
    </NavState>
  </>
)
```

```
);  
}
```

```
export default App;
```

## 2. Homepage.js

```
import React, { useContext, useEffect } from 'react'  
import logo from './Card-2.png'  
import Navbar from './Navbar'  
import { NavLink } from 'react-router-dom'  
import Footer from './Footer'  
import navContext from './Context/navContext'  
import resto_maker from './Restro-maker.jpg'  
import background from './background3.JPG'
```

```
export default function HomePage(props) {
```

```
  const context = useContext(navContext);  
  const {logstate, setTitle} = context;
```

```
  useEffect(()=>{  
    setTitle("home");  
  },[setTitle])
```

```
  return (  
    <div className='background-home'>  
      <img src={background} alt="background" style={{height:"135vh",width:"100vw",zIndex:"-  
1",position:"absolute",opacity:"0.9"}}></img>  
      <div className='d-flex w-100' style={{backgroundColor:"rgb(0,0,0,0.4)}}>  
        <img src={resto_maker} alt="Restro-maker logo" height="150px"/>  
        <div className="w-100 fw-bold text-align-center">  
          <h2 className="w-100 fs-1 fw-bold" style={{ color: "gold",textAlign:"center", margin:  
"10px auto", textShadow:"-1px -1px 0 black, 1px -1px 0 black, 1px 1px 0 black,1px 1px 0 black"  
}}>Welcome to Restro Maker</h2>  
          <p className="fs-4" style={{ color: "#fff", margin: "auto",textAlign:"center",width:"90%"  
}}>  
            Your one-stop solution for managing your restaurants. From creating profiles to  
organizing menus and placing orders, explore all the features we offer to streamline your business.  
          </p>  
        </div>  
      </div>  
      <Navbar/>  
    </div> */>  
    <img src={resto_maker} alt="Restro-maker logo" height="240vh" className="my-0  
rounded-top-0 rounded-end-circle mx-auto d-block" style={{zIndex:"2",position:"absolute",left:"-  
0.5vw",top:"-30px"}}/>  
    <Navbar/>  
    </div> */>  
  
    <Navbar/>  
    <*/logstate===0 && <div className="fw-bold fs-3 mt-5 alert alert-warning" role="alert">Login  
to access restaurants and other features.</div>*/>
```

```

    <div style={{display: "flex", alignItems:"center", justifyContent:"center", flexWrap:
"wrap",height:"81vh"}}>
      <div className="card rounded" style={{maxWidth: 16 +"rem", margin: "0rem
2rem",border:"1px solid black"}}>
        <img src={logo} className="card-img-top" alt="..." />
        <div className="card-body rounded">
          <h5 className="card-title">Create Profile</h5>
          <p className="card-text">Create a profile to get started. You can add your restaurants,
menu card and other information.</p>
          <NavLink to={logstate===0?"/login":"/restaurant"} className="btn btn-
primary">{logstate===0?"Sign-in":"Edit Profile"}</NavLink>
        </div>
      </div>
      <div className="card" style={{maxWidth: 16 +"rem", margin: "0rem 2rem",border:"1px solid
black"}}>
        <img src={logo} className="card-img-top" alt="..." />
        <div className="card-body rounded">
          <h5 className="card-title">How to use</h5>
          <p className="card-text">To know what this website provides, and manage multiple
restaurants details - Click the button below.</p>
          <NavLink to="/about" className="btn btn-primary">Read Content</NavLink>
        </div>
      </div>
      <div className="card" style={{maxWidth: 16 +"rem", margin: "0rem 2rem",border:"1px solid
black"}}>
        <img src={logo} className="card-img-top" alt="..." />
        <div className="card-body rounded">
          <h5 className="card-title">Features</h5>
          <p className="card-text">Some quick example text to build on the card title and make
up the bulk of the card's content.</p>
          <NavLink to={logstate===0?"/login":"/orderpage"} className="btn btn-primary">Order
Food</NavLink>
        </div>
      </div>
    </div>
    {logstate===1 && <Footer/>}
  </div>
)
}

```

### 3. UserService.js

```

import axios from "axios";
const config = {
  Headers:{
    "Access-Control-Allow-Origin":"*",
    "Access-Control-Allow-Methods":"GET,PUT,POST,DELETE",
    "Authorization": `Bearer ${localStorage.getItem("token")}`
  }
}

axios.interceptors.request.use(

```

```

config => {
  const token = localStorage.getItem("token");
  if (token) {
    config.headers.Authorization = `Bearer ${token}`;
  }
  return config;
},
error => {
  return Promise.reject(error);
}
);

const REST_API_BASE_URL = "http://localhost:8080/api/user";

export const UserLogin = (details) => axios.post(REST_API_BASE_URL+"/login",details,config);
export const ChangePassword = (details) =>
  axios.post(REST_API_BASE_URL+"/password",details,config);
export const NewPassword = (details) =>
  axios.put(REST_API_BASE_URL+"/password",details,config);

export const CreateNewUser = (user) => axios.post(REST_API_BASE_URL+"/signup",user,config);
export const findDuplicateUser = (user) =>
  axios.get(REST_API_BASE_URL+"/signup",{params:{username:user}},config);

export const AddRestaurant = (user_rest) =>
  axios.post(REST_API_BASE_URL+"/restaurant/new",user_rest,config);
export const RemoveRestaurant = (user_rest) =>
  axios.post(REST_API_BASE_URL+"/restaurant/remove",user_rest,config);
export const ChangeRestaurantName = (rest_details) =>
  axios.put(REST_API_BASE_URL+"/restaurant/rename",rest_details,config);
export const GetAllUserRestaurants = () =>
  axios.get(REST_API_BASE_URL+"/restaurant/userpage",config);
export const GetAllOwnerRestaurants = () =>
  axios.get(REST_API_BASE_URL+"/restaurant/all",config);
export const SuggestAllRestaurants = (name) =>
  axios.get(REST_API_BASE_URL+"/restaurant/"+name,{params:{}},config);

export const AddFoodItem = (id,item) =>
  axios.post(REST_API_BASE_URL+"/menu/add",item,{params:{rest_id:id}},config);
export const GetAllItems = (id) =>
  axios.get(REST_API_BASE_URL+"/menu/all",{params:{rest_id:id}},config);
export const GetAllCategories = (id) =>
  axios.get(REST_API_BASE_URL+"/menu/categories",{params:{rest_id:id}},config);
export const RemoveItems = (id,items) =>
  axios.post(REST_API_BASE_URL+"/menu/remove/items",items,{params:{rest_id:id}},config);
export const RemoveCategories = (id,items) =>
  axios.post(REST_API_BASE_URL+"/menu/remove/category",items,{params:{rest_id:id}},config);

```

## BACKEND (partial code)

### 1. Application.java

```
package com.restaurant.demo;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class RestoMakerApplication {
    public static void main(String[] args) {
        SpringApplication.run(RestoMakerApplication.class, args);
    }
}
```

### 2. Restaurant.java (Entity class)

```
package com.restaurant.demo.entity;

import jakarta.persistence.*;
import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;
import org.springframework.boot.autoconfigure.web.WebProperties;

import java.util.List;

@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor

@Entity
@Table(name = "restaurant")

public class Restaurant{
    @Id
    @GeneratedValue(strategy= GenerationType.AUTO)
    private Long rest_id;

    @Column(name = "restaurant_name", nullable = false)
    private String rest_name;

    @Column(name = "location")
    private String location;

    @Column(name = "foodType")
    private String foodtype;

    @Column(name = "rest_rating")
```

```

private int rest_rating;

@OneToMany(cascade = CascadeType.ALL, orphanRemoval = true)
@JoinColumn(name = "restaurant_id", referencedColumnName = "rest_id")
private List<FoodItems> items;
}

```

### 3. RestaurantController.java ( Controller class )

```

package com.restaurant.demo.controller;

import com.restaurant.demo.dto.RestaurantDto;
import com.restaurant.demo.dto.RestaurantIdDto;
import com.restaurant.demo.security.JwtHelper;
import com.restaurant.demo.service.RestaurantService;
import lombok.AllArgsConstructor;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@CrossOrigin("*")
@RequestMapping("/api/user/restaurant")
@AllArgsConstructor
public class RestaurantController {
    private RestaurantService restaurantService;
    @Autowired
    private JwtHelper jwtHelper;

    @PostMapping("/new")
    public ResponseEntity<List<RestaurantDto>> addRestaurant(@RequestBody RestaurantDto
restaurantDto, @RequestHeader("Authorization") String token){
        if (token.startsWith("Bearer ")) {
            token = token.substring(7);
        }
        String username = jwtHelper.getUsernameFromToken(token);
        List<RestaurantDto> savedRestaurant =
restaurantService.addRestaurant(username, restaurantDto);
        return new ResponseEntity<>(savedRestaurant, HttpStatus.CREATED);
    }

    @GetMapping("/userpage")
    public ResponseEntity<List<RestaurantDto>> getAllRestaurant(){
        List<RestaurantDto> restaurantDto = restaurantService.getAllRestaurant();
        return ResponseEntity.ok(restaurantDto);
    }

    @GetMapping("/all")
    public ResponseEntity<List<RestaurantDto>>

```



```

getAllRestaurantByUsername(@RequestHeader("Authorization") String token){
    if (token.startsWith("Bearer ")) {
        token = token.substring(7);
    }
    String username = jwtHelper.getUsernameFromToken(token);
    List<RestaurantDto> restaurantDto =
restaurantService.getAllRestaurantByUsername(username);
    return ResponseEntity.ok(restaurantDto);
}

// For search restaurant option.
@GetMapping("/{rest_name}")
public ResponseEntity<List<RestaurantDto>>
searchAllRestaurantByUserName(@RequestHeader("Authorization") String token,
@PathVariable("rest_name") String name){
    if (token.startsWith("Bearer ")) {
        token = token.substring(7);
    }
    String username = jwtHelper.getUsernameFromToken(token);
    List<RestaurantDto> restaurantDto =
restaurantService.searchAllRestaurantByName(username,name);
    return ResponseEntity.ok(restaurantDto);
}

@PostMapping("/remove")
public ResponseEntity<List<RestaurantDto>> deleteRestaurantsbyID(@RequestBody
RestaurantIdDto restaurantIdDto,@RequestHeader("Authorization") String token){
    if (token.startsWith("Bearer ")) {
        token = token.substring(7);
    }
    String username = jwtHelper.getUsernameFromToken(token);
    List<Long> restaurantIds = restaurantIdDto.getRestaurantIds();
    List<RestaurantDto> savedRestaurant =
restaurantService.deleteRestaurantsbyID(username,restaurantIds);
    return new ResponseEntity<>(savedRestaurant, HttpStatus.CREATED);
//    return ResponseEntity.ok("Restaurant deleted Successfully");
}

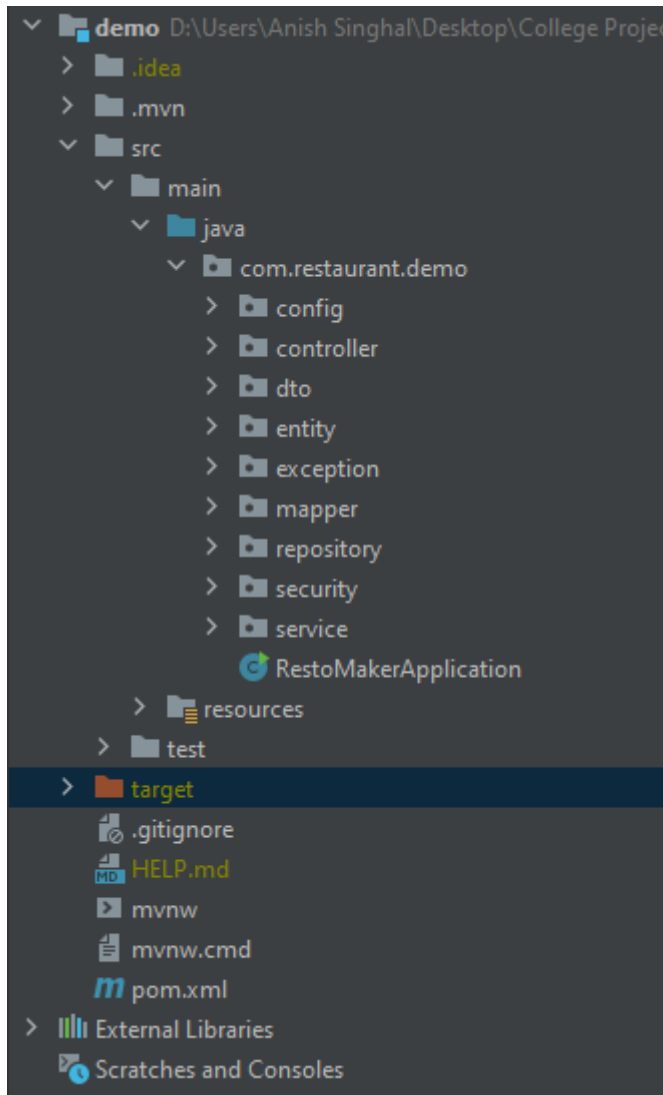
// @GetMapping("/id")
// public ResponseEntity<RestaurantDto> getRestaurantbyID(@PathVariable("id") Long
restaurantID){
//    RestaurantDto restaurantDto = restaurantService.getRestaurantbyID(restaurantID);
//    return ResponseEntity.ok(restaurantDto);
// }

@PutMapping("/rename")
public ResponseEntity<RestaurantDto> updateRestaurantbyID(@RequestBody RestaurantDto
details,@RequestHeader("Authorization") String token){
    if (token.startsWith("Bearer ")) {
        token = token.substring(7);
    }
}

```


```
String username = jwtHelper.getUsernameFromToken(token);
RestaurantDto restaurantDto = restaurantService.updateRestaurantbyID(details);
return ResponseEntity.ok(restaurantDto);
}
}
```

## Complete Backend File Structure



OUTPUT:



HOME PAGE




## Welcome to Restro Maker

Your one-stop solution for managing your restaurants. From creating profiles to organizing menus and placing orders, explore all the features we offer to streamline your business.

[Restro-Maker](#)[Home](#)[About](#)[Restaurants](#)[Features](#)






### Create Profile

Create a profile to get started. You can add your restaurants, menu card and other information.


[Edit Profile](#)



### How to use

To know what this website provides, and manage multiple restaurants details - Click the button below.

[Read Content](#)



### Features

Some quick example text to build on the card title and make up the bulk of the card's content.

[Order Food](#)

#### User

- Profile
- Order Page
- Restaurants

#### Owner

- Profile
- Edit details
- View details

#### General

- Login
- About
- Contact



#### Subscribe to our website

Get monthly update of new and exciting restaurants along with their amazing dishes from us.


[Subscribe](#)

© 2024 Company, Inc. All rights reserved.

# OWNER'S RESTAURANT PAGE


[Restro-Maker](#) [Home](#) [About](#) [Restaurants](#) [Features ▾](#)  

## Your Restaurant List




**HealthyFOODS\_3**  
Chennai

Some quick example text to build on the card title and make up the bulk of the card's content.




**HealthyFOODS\_4**  
Mumbai

Some quick example text to build on the card title and make up the bulk of the card's content.




**HealthyFOODS\_1**  
Delhi

Some quick example text to build on the card title and make up the bulk of the card's content.




**HealthyFOODS\_2**  
Agra

Some quick example text to build on the card title and make up the bulk of the card's content.



**My Kitchen1**  
hyderabad

Some quick example text to build on the card title and make up the bulk of the card's content.



**Chat centre**  
Delhi


Some quick example text to build on the card title and make up the bulk of the card's content.

The restaurant data is fetched from the backend via API requests. When the page loads, a GET request is sent from the React frontend to the Spring Boot backend. Updates, additions, or deletions are handled through POST, PUT, or DELETE requests, ensuring the data is synchronized between the frontend and backend.

## ADD RESTAURANT FORM

### Create new Restaurant ×

Enter Restaurant Details (required)



**Default Restaurant Image**

Choose File

No file chosen

Restaurant Name

Enter Name

Location

Enter Location

Food Type

Select Type ▼

Close

Create

On clicking the "Add Restaurant" button the above modal form appears asking the details of the restaurant. On clicking the "Create" button a POST request is send and the restaurant is added to your list in the database and the page gets updated.