**TASK 1**

**Restaurant App**

**App.js**import React, { useState } from 'react';

import './App.css';

const App = () => {

  const [customerName, setCustomerName] = useState('');

  const [customerNumber, setCustomerNumber] = useState('');

  const [selectedItem, setSelectedItem] = useState('1');

  const [itemSize, setItemSize] = useState('Medium');

  const [quantity, setQuantity] = useState(1);

  const [specialInstructions, setSpecialInstructions] = useState('');

  const [toppings, setToppings] = useState([]);

  const [orderType, setOrderType] = useState('pickup');

  const [termsAccepted, setTermsAccepted] = useState(false);

  const [orderDetails, setOrderDetails] = useState(null);

  const [errors, setErrors] = useState({});

  const menuItems = [

    { id: '1', name: 'Cake' },

    { id: '2', name: 'Ice Cream' },

    { id: '3', name: 'Juice' },

  ];

  const images = [

    '/images/1.jpg',

    '/images/2.jpg',

    '/images/3.jpg',

  ];

  const handleSubmit = (e) => {

    e.preventDefault();

    const newErrors = {};

    if (customerNumber.length !== 10) newErrors.customerNumber = 'Contact number must be exactly 10 digits.';

    if (quantity < 1 || quantity > 3) newErrors.quantity = 'Quantity must be between 1 and 3.';

    if (!termsAccepted) newErrors.terms = 'You must accept the terms and conditions.';

    if (Object.keys(newErrors).length > 0) {

      setErrors(newErrors);

      return;

    }

    const order = {

      customerName,

      customerNumber,

      menuItem: menuItems.find(item => item.id === selectedItem).name,

      itemSize,

      quantity,

      toppings,

      specialInstructions,

      orderType,

      totalOrder: quantity,

    };

    setOrderDetails(order);

    setErrors({});

  };

  const handleReset = () => {

    setCustomerName('');

    setCustomerNumber('');

    setSelectedItem('1');

    setItemSize('Medium');

    setQuantity(1);

    setSpecialInstructions('');

    setToppings([]);

    setOrderType('pickup');

    setTermsAccepted(false);

    setOrderDetails(null);

    setErrors({});

  };

  const handleToppingsChange = (e) => {

    const { value, checked } = e.target;

    if (checked) {

      setToppings([...toppings, value]);

    } else {

      setToppings(toppings.filter(topping => topping !== value));

    }

  };

  return (

    <div className="App">

      <header className="App-header">

        <h1>Restaurant Page</h1>

      </header>

      <main className="container">

        <section id="order">

          <div className="form-container">

            <h1>Restaurant Order Form</h1>

            <form onSubmit={handleSubmit}>

              <label>

                Customer Name:

                <input type="text" value={customerName} onChange={(e) => setCustomerName(e.target.value)} />

              </label>

              <label>

                Customer Number:

                <input type="text" value={customerNumber} onChange={(e) => setCustomerNumber(e.target.value)} maxLength="10" />

                {errors.customerNumber && <p className="error">{errors.customerNumber}</p>}

              </label>

              <label>

                Menu Item:

                <select value={selectedItem} onChange={(e) => setSelectedItem(e.target.value)}>

                  {menuItems.map(item => (

                    <option key={item.id} value={item.id}>{item.name}</option>

                  ))}

                </select>

              </label>

              <label>

                Size:

                <select value={itemSize} onChange={(e) => setItemSize(e.target.value)}>

                  <option value="Small">Small</option>

                  <option value="Medium">Medium</option>

                  <option value="Large">Large</option>

                </select>

              </label>

              <label>

                Quantity:

                <input type="number" value={quantity} onChange={(e) => setQuantity(Math.min(3, Math.max(1, e.target.value)))} />

                {errors.quantity && <p className="error">{errors.quantity}</p>}

              </label>

              <label>

                Extra Toppings:

                <div className="toppings">

                  <label>

                    <input type="checkbox" value="Cheese" checked={toppings.includes('Cheese')} onChange={handleToppingsChange} />

                    Cheese

                  </label>

                  <label>

                    <input type="checkbox" value="Pepperoni" checked={toppings.includes('Pepperoni')} onChange={handleToppingsChange} />

                    Pepperoni

                  </label>

                  <label>

                    <input type="checkbox" value="Mushrooms" checked={toppings.includes('Mushrooms')} onChange={handleToppingsChange} />

                    Mushrooms

                  </label>

                </div>

              </label>

              <label>

                Order Type:

                <div className="order-type">

                  <label>

                    <input type="radio" value="pickup" checked={orderType === 'pickup'} onChange={(e) => setOrderType(e.target.value)} />

                    Pickup

                  </label>

                  <label>

                    <input type="radio" value="delivery" checked={orderType === 'delivery'} onChange={(e) => setOrderType(e.target.value)} />

                    Delivery

                  </label>

                </div>

              </label>

              <label>

                Special Instructions:

                <textarea value={specialInstructions} onChange={(e) => setSpecialInstructions(e.target.value)} />

              </label>

              <label>

                <input type="checkbox" checked={termsAccepted} onChange={(e) => setTermsAccepted(e.target.checked)} />

                I accept the terms and conditions

              </label>

              {errors.terms && <p className="error">{errors.terms}</p>}

              <button type="submit">Order Now</button>

              <button type="button" onClick={handleReset}>Reset</button>

            </form>

            {orderDetails && (

              <div className="order-summary-modal">

                <div className="modal-content">

                  <h2>Order Details</h2>

                  <p>Customer Name: {orderDetails.customerName}</p>

                  <p>Customer Number: {orderDetails.customerNumber}</p>

                  <p>Menu Item: {orderDetails.menuItem} - {orderDetails.itemSize}</p>

                  <p>Quantity: {orderDetails.quantity}</p>

                  <p>Toppings: {orderDetails.toppings.join(', ')}</p>

                  <p>Order Type: {orderDetails.orderType}</p>

                  <p>Special Instructions: {orderDetails.specialInstructions}</p>

                  <p>Total Order: {orderDetails.totalOrder}</p>

                </div>

              </div>

            )}

          </div>

        </section>

        <section id="gallery">

          <h2>Popular Dishes</h2>

          <div className="gallery">

            {images.map((src, index) => (

              <img key={index} src={src} alt={`Dish ${index + 1}`} />

            ))}

          </div>

        </section>

      </main>

      <footer>

        <p>&copy; 2024 Restaurant</p>

      </footer>

    </div>

  );

};

export default App;

**App.css**

/\* App.css \*/

/\* Set a smaller base font size \*/

body {

  font-family: Arial, sans-serif;

  font-size: 14px; /\* Smaller font size \*/

  background-color: #f8f8f8;

  margin: 0;

  padding: 0;

}

.App {

  text-align: center;

}

.App-header {

  background-color: #2c63d3;

  padding: 20px;

  color: white;

}

.container {

  max-width: 800px;

  margin: 20px auto;

  padding: 20px;

  background-color: white;

  box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);

  border-radius: 8px;

}

h1, h2 {

  color: #333;

  font-size: 1.6em; /\* Slightly smaller heading \*/

}

.form-container {

  margin-bottom: 20px;

}

label {

  display: block;

  margin-bottom: 10px;

  text-align: left;

  font-weight: bold;

  font-size: 0.9em; /\* Smaller label font \*/

}

input[type="text"],

input[type="number"],

textarea,

select {

  width: 100%;

  padding: 8px;

  margin: 5px 0;

  box-sizing: border-box;

  border: 1px solid #ccc;

  border-radius: 4px;

  font-size: 0.9em; /\* Smaller input font \*/

}

textarea {

  height: 100px;

}

.toppings,

.order-type {

  margin: 10px 0;

  text-align: left;

}

.toppings label,

.order-type label {

  display: block;

  margin-bottom: 5px;

  font-size: 0.9em; /\* Smaller label font \*/

}

.error {

  color: red;

  font-size: 0.8em; /\* Smaller error font \*/

}

button {

  background-color: #007bff;

  color: white;

  border: none;

  padding: 8px 16px; /\* Smaller button padding \*/

  text-align: center;

  font-size: 0.9em; /\* Smaller button font \*/

  border-radius: 4px;

  cursor: pointer;

  margin: 10px 5px;

}

button:hover {

  background-color: #0056b3;

}

.order-summary-modal {

  position: fixed;

  top: 50%;

  left: 50%;

  transform: translate(-50%, -50%);

  background-color: white;

  padding: 20px;

  box-shadow: 0 4px 12px rgba(0, 0, 0, 0.3);

  border-radius: 8px;

  z-index: 1000;

  max-width: 90%;

}

.modal-content {

  text-align: left;

  font-size: 0.9em; /\* Smaller font in modal \*/

}

.gallery {

  display: flex;

  justify-content: space-between;

  flex-wrap: wrap;

  margin-top: 20px;

}

.gallery img {

  max-width: 100%;

  height: auto;

  border-radius: 8px;

  margin: 5px;

  flex: 1;

}

footer {

  margin-top: 20px;

  padding: 20px;

  background-color: #1753cc;

  color: white;

  font-size: 0.9em; /\* Smaller footer font \*/

}

**Output:**

