**Assignment 1 – Home Page**

**Aim**

To design and implement a dynamic Home Page using React.js that displays upcoming events and allows users to add them to their cart interactively.

**Introduction**

The Home Page is the heart of EventHub, where users can browse a curated list of upcoming events. Each event is displayed inside an attractive card that includes an image, title, date, price, and location. Users can easily add an event to their cart for booking. The goal of this page is to showcase dynamic rendering, responsive layout, and React’s reusable component design.

**Theory**

React is a component-based library that makes it easy to create modular, data-driven UIs. In this page, the HomePage component takes in an array of events and a callback function onAddToCart() as props. Using JavaScript’s .map() function, each event is rendered dynamically into a card structure. This demonstrates the concept of declarative rendering — the UI automatically updates whenever the underlying data changes.

React’s Virtual DOM ensures efficient updates, refreshing only the parts of the UI that actually change. CSS Grid is used for a flexible, responsive layout that adapts to various screen sizes. The styling emphasizes soft shadows, rounded corners, and gradient buttons to create an engaging interface. Hover effects enhance interactivity and visual feedback. The use of props and callback functions also demonstrates unidirectional data flow, a fundamental React principle ensuring predictable state updates.

This component demonstrates how React simplifies dynamic content rendering while maintaining a clean and maintainable codebase.

**Conclusion**

The Home Page successfully demonstrates how React can dynamically display event data while maintaining an interactive, responsive design. It highlights modular coding practices and real-time UI updates through state and props.

**Code**

**Pages/HomePage.css**

.home-container {

  text-align: center;

  padding: 30px;

  animation: fadeIn 0.6s ease;

}

.event-grid {

  display: grid;

  grid-template-columns: repeat(auto-fit, minmax(260px, 1fr));

  gap: 20px;

  margin-top: 20px;

}

.event-card {

  background: white;

  border-radius: 16px;

  overflow: hidden;

  transition: 0.3s;

  box-shadow: 0 4px 10px rgba(0, 0, 0, 0.2);

}

.event-card:hover {

  transform: translateY(-5px) scale(1.02);

}

.event-card img {

  width: 100%;

  height: 160px;

  object-fit: cover;

}

.add-btn {

  margin: 10px;

  padding: 8px 15px;

  border: none;

  border-radius: 20px;

  background: linear-gradient(90deg, #0078d7, #00b4d8);

  color: white;

  font-weight: 500;

  cursor: pointer;

  transition: 0.3s;

}

.add-btn:hover {

  transform: scale(1.05);

}

**Pages/HomePage.jsx**

import React from "react";

import "./HomePage.css";

export const HomePage = ({ events, onAddToCart }) => (

  <div className="home-container">

    <h2 className="title">🎉 Upcoming Events</h2>

    <div className="event-grid">

      {events.map((e) => (

        <div key={e.id} className="event-card">

          <img src={e.imageUrl} alt={e.name} />

          <h3>{e.name}</h3>

          <p>{e.date}</p>

          <p>{e.location}</p>

          <p className="price">₹{e.price}</p>

          <button className="add-btn" onClick={() => onAddToCart(e.id)}>

            Add to Cart

          </button>

        </div>

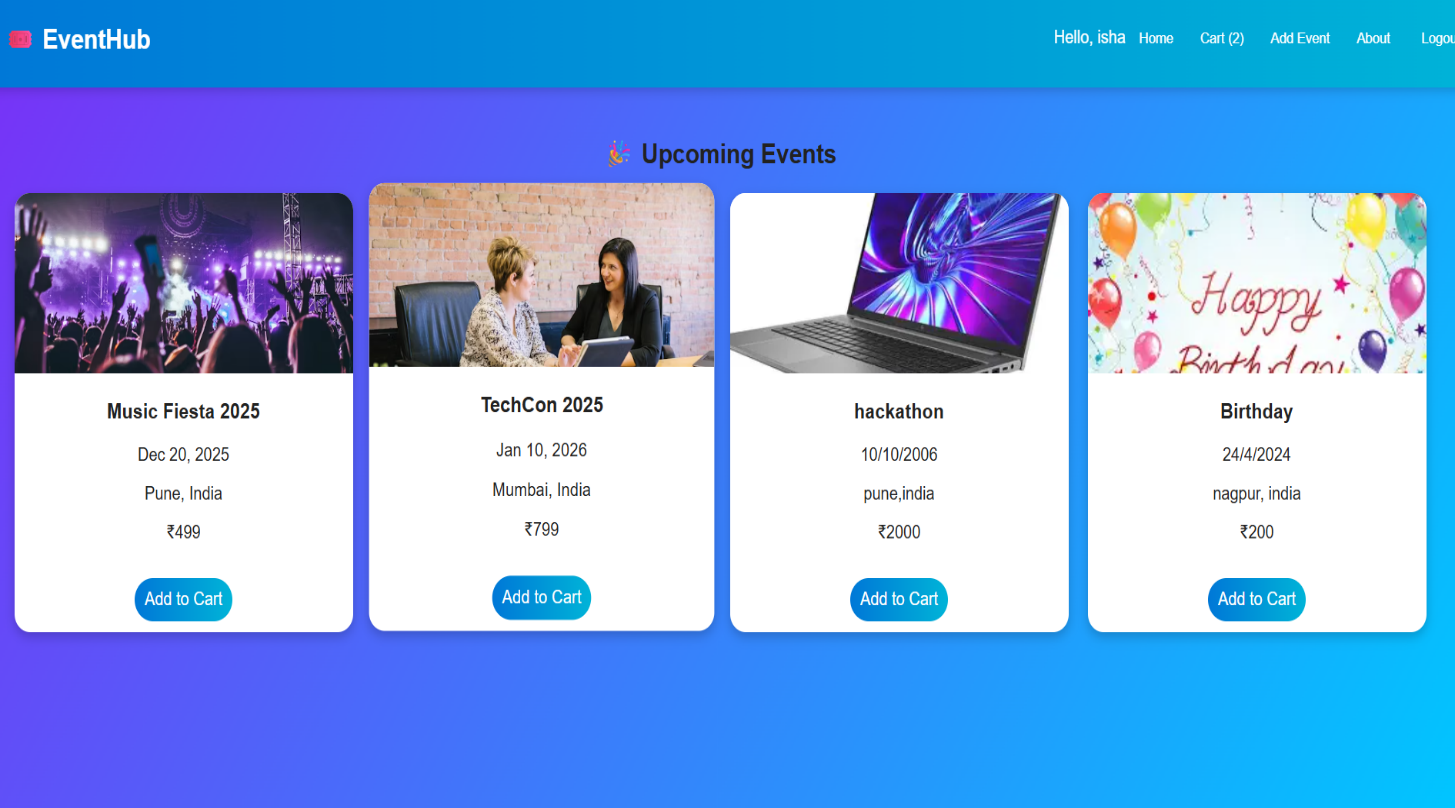
      ))}

    </div>

  </div>

);

**OUTPUT:**



**Assignment 2 – Login Page**

**Aim**

To implement a Login Page that allows users to authenticate using React.js with controlled form inputs and validation.

**Introduction**

The Login Page is the entry point to the EventHub application. It allows users to access their account using predefined credentials. The page demonstrates React form handling and conditional rendering while maintaining visual consistency through CSS.

**Theory**

React’s useState hook is used to manage form input values dynamically. The page includes fields for username and password. When the user clicks “Login,” the form triggers a function that validates credentials. Controlled components ensure synchronization between form fields and component state.  
If the entered credentials match, onAuthSuccess() updates the global login state. This demonstrates state lifting—child components communicating data changes to the parent through props.

The component design ensures that the login and register views share one layout. A conditional statement checks if the view is “login” or “register,” simplifying code reuse. From a UI perspective, the gradient background, box shadows, and rounded input fields create a professional, modern interface.

**Conclusion**

The Login Page implements effective form handling and state management in React. It provides users with a secure, interactive, and visually appealing authentication experience.

**Code**

export const AuthScreen = ({ view, onAuthSuccess, switchView }) => {

const [username, setUsername] = useState("");

const [password, setPassword] = useState("");

const handleSubmit = (e) => {

e.preventDefault();

if (username === "user@example.com" && password === "password") {

onAuthSuccess({ name: "User", email: username });

} else {

alert("Invalid credentials");

}

};

return (

<div className="login-wrapper">

<div className="login-card">

<h2>sign in</h2>

<form onSubmit={handleSubmit}>

<input type="text" placeholder="Username" value={username}

onChange={(e) => setUsername(e.target.value)} required />

<input type="password" placeholder="Password" value={password}

onChange={(e) => setPassword(e.target.value)} required />

<button className="login-btn" type="submit">LOGIN</button>

</form>

<p className="switch-text">

Don’t have an account?{" "}

<span onClick={() => switchView("register")}>Register here</span>

</p>

</div>

</div>

);

};

CSS:

.login-wrapper {

display: flex;

justify-content: center;

align-items: center;

height: 90vh;

background: linear-gradient(135deg, #7b2ff7, #00c6ff);

}

.login-card {

background: linear-gradient(135deg, #9be2fe, #67d1fb);

border-radius: 80px 10px 10px 10px;

padding: 40px 60px;

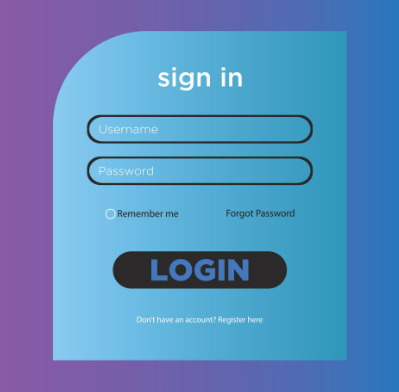
text-align: center;

width: 340px;

box-shadow: 0 6px 20px rgba(0, 0, 0, 0.25);

}

**OUTPUT:**



**Assignment 3 – Register Page**

**Aim**

To create a user registration page that allows new users to sign up and validates password confirmation.

**Introduction**

Registration allows new users to create accounts. In EventHub, the same AuthScreen component handles both login and registration, showing React’s flexibility.

**Theory**

The page uses useState hooks to manage inputs for username, password, and confirmation. The handleSubmit function checks if both passwords match before proceeding. Upon success, it triggers onAuthSuccess() and redirects to the home view.

The register form adds a confirm password input using conditional rendering (view === "register"). This demonstrates React’s ability to render dynamic UIs based on state. CSS enhances readability and adds smooth hover animations, providing feedback to users during interactions.

**Conclusion**

The Register Page demonstrates conditional rendering, form validation, and modular component reuse. It ensures data consistency and an intuitive sign-up process.

**Code**

export const AuthScreen = ({ view, onAuthSuccess, switchView }) => {

const [username, setUsername] = useState("");

const [password, setPassword] = useState("");

const [confirm, setConfirm] = useState("");

const handleSubmit = (e) => {

e.preventDefault();

if (password !== confirm) {

alert("Passwords do not match!");

return;

}

const name = username.split("@")[0];

onAuthSuccess({ name, email: username });

};

return (

<div className="login-wrapper">

<div className="login-card">

<h2>register</h2>

<form onSubmit={handleSubmit}>

<input type="text" placeholder="Username or Email"

value={username} onChange={(e) => setUsername(e.target.value)} required />

<input type="password" placeholder="Password"

value={password} onChange={(e) => setPassword(e.target.value)} required />

<input type="password" placeholder="Confirm Password"

value={confirm} onChange={(e) => setConfirm(e.target.value)} required />

<button className="login-btn" type="submit">REGISTER</button>

</form>

<p className="switch-text">

Already a member?{" "}

<span onClick={() => switchView("login")}>Login here</span>

</p>

</div>

</div>

);

};

CSS

.login-card h2 {

text-transform: lowercase;

font-size: 28px;

font-weight: 600;

color: #111;

}

.login-card input {

width: 100%;

padding: 10px 15px;

margin: 10px 0;

border-radius: 25px;

border: 2px solid #333;

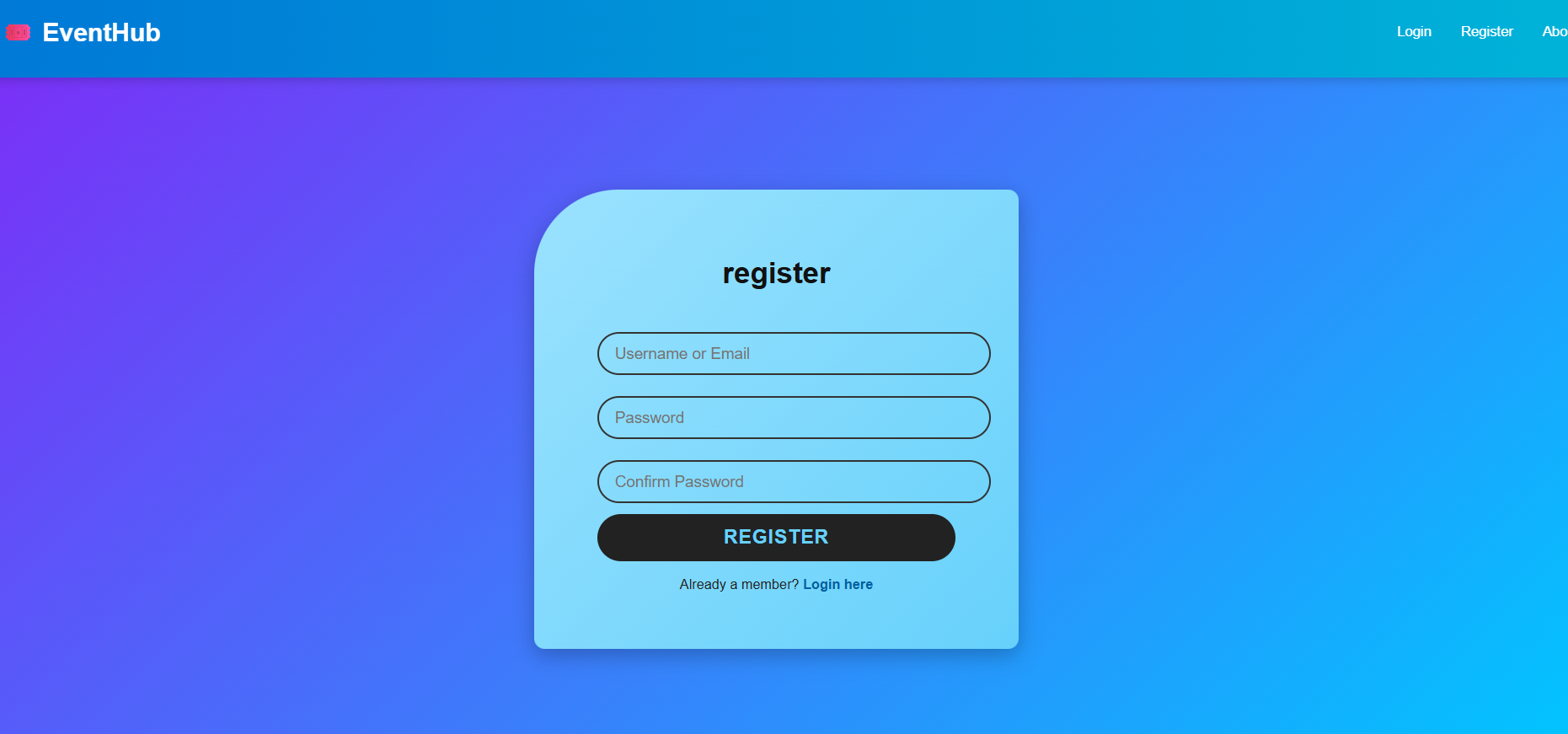
background: transparent;

font-size: 15px;

outline: none;

}

**OUTPUT:**



**Assignment 4 – Add Event Page**

**Aim**

To allow logged-in users to add new events dynamically into the EventHub system.

**Introduction**

This page empowers users to contribute events, demonstrating form handling and dynamic data updates.

**Theory**

The page maintains form data using useState. Each input field updates state via a single handler. On submission, it validates data and calls onAddEvent() in the parent, which updates the event list. CSS animations (fadeIn) and gradient buttons create an attractive design. The feature also represents **data lifting and immutability** in React—new data never overwrites existing state but creates updated copies.

**Conclusion**

The Add Event Page showcases form validation, state handling, and parent-child communication through props.

**Code**

import React, { useState } from "react";

import "./AddEventPage.css";

export const AddEventPage = ({ onAddEvent }) => {

  const [eventData, setEventData] = useState({

    name: "",

    date: "",

    price: "",

    location: "",

    imageUrl: "",

  });

  const handleChange = (e) => {

    const { name, value } = e.target;

    setEventData({ ...eventData, [name]: value });

  };

  const handleSubmit = (e) => {

    e.preventDefault();

    if (

      !eventData.name ||

      !eventData.date ||

      !eventData.price ||

      !eventData.location ||

      !eventData.imageUrl

    ) {

      alert("Please fill all fields!");

      return;

    }

    onAddEvent(eventData);

    setEventData({

      name: "",

      date: "",

      price: "",

      location: "",

      imageUrl: "",

    });

  };

  return (

    <div className="add-event-wrapper">

      <div className="add-event-card">

        <h2>➕ Add New Event</h2>

        <form onSubmit={handleSubmit}>

          <input

            type="text"

            name="name"

            placeholder="Event Name"

            value={eventData.name}

            onChange={handleChange}

          />

          <input

            type="text"

            name="date"

            placeholder="Event Date (e.g., Mar 12, 2026)"

            value={eventData.date}

            onChange={handleChange}

          />

          <input

            type="number"

            name="price"

            placeholder="Ticket Price (₹)"

            value={eventData.price}

            onChange={handleChange}

          />

          <input

            type="text"

            name="location"

            placeholder="Location"

            value={eventData.location}

            onChange={handleChange}

          />

          <input

            type="text"

            name="imageUrl"

            placeholder="Image URL"

            value={eventData.imageUrl}

            onChange={handleChange}

          />

          <button type="submit" className="add-btn">

            Add Event

          </button>

        </form>

      </div>

    </div>

  );

};

CSS :

.add-event-wrapper {

  display: flex;

  justify-content: center;

  align-items: center;

  height: 85vh;

  background: linear-gradient(135deg, #7b2ff7, #00c6ff);

}

.add-event-card {

  background: #fff;

  border-radius: 25px;

  padding: 40px;

  width: 360px;

  text-align: center;

  box-shadow: 0 4px 15px rgba(0, 0, 0, 0.3);

  animation: fadeIn 0.5s ease;

}

.add-event-card h2 {

  margin-bottom: 20px;

  color: #0078d7;

}

.add-event-card form {

  display: flex;

  flex-direction: column;

}

.add-event-card input {

  margin: 8px 0;

  padding: 10px 15px;

  border: 2px solid #ccc;

  border-radius: 10px;

  font-size: 15px;

  outline: none;

  transition: 0.3s;

}

.add-event-card input:focus {

  border-color: #0078d7;

  box-shadow: 0 0 6px #0078d7;

}

.add-btn {

  margin-top: 15px;

  background: linear-gradient(90deg, #0078d7, #00b4d8);

  color: white;

  border: none;

  border-radius: 25px;

  padding: 10px;

  font-size: 16px;

  cursor: pointer;

  transition: transform 0.2s ease;

}

.add-btn:hover {

  transform: scale(1.05);

  box-shadow: 0 5px 15px rgba(0, 184, 216, 0.4);

}

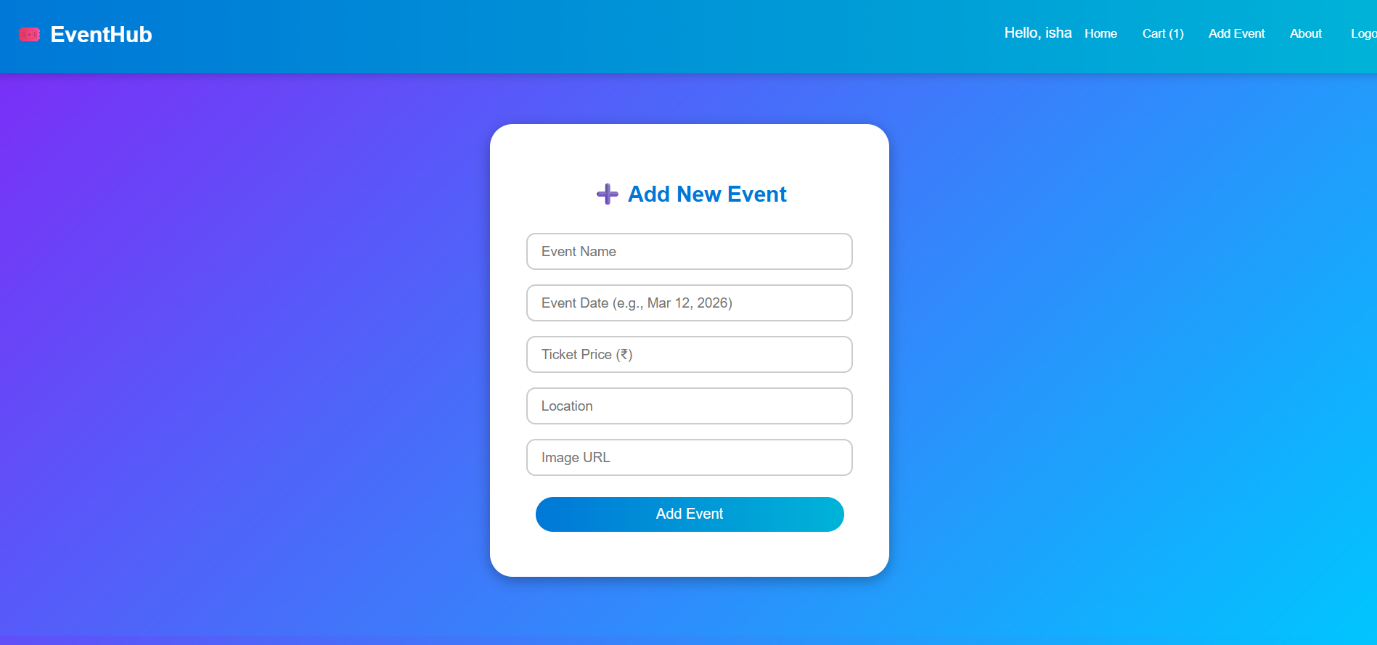
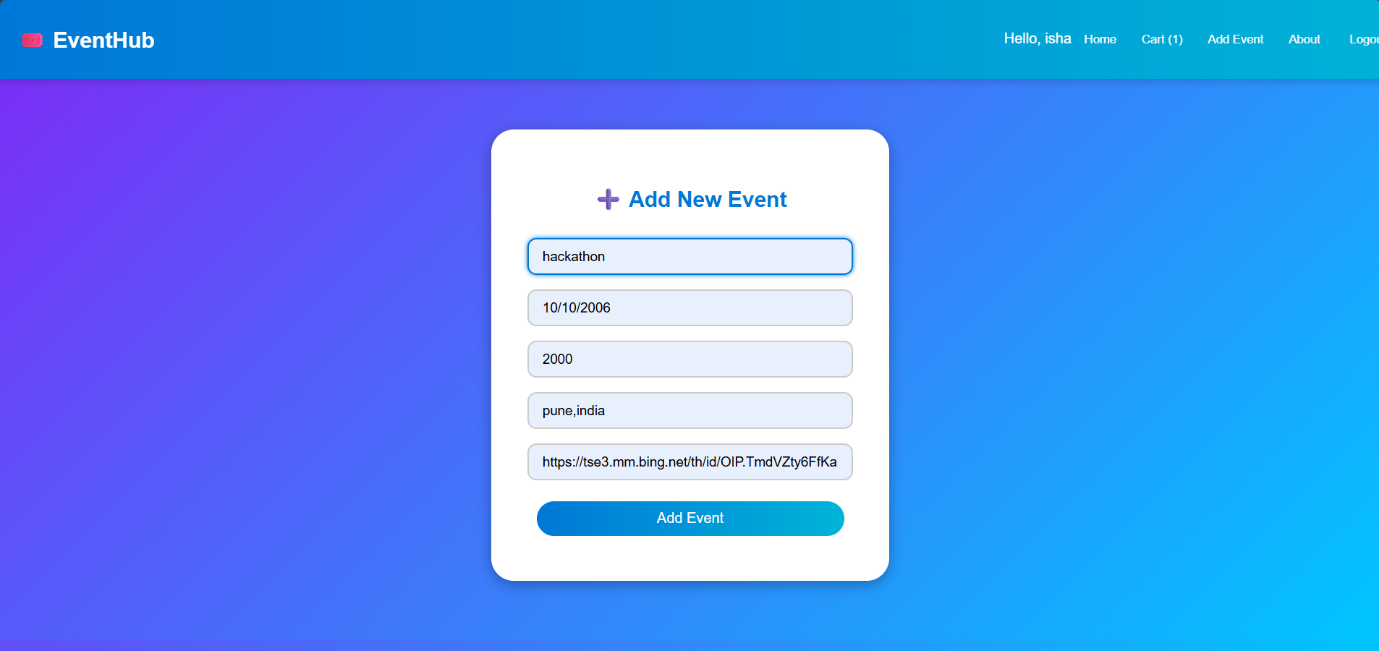
@keyframes fadeIn {

  from { opacity: 0; transform: translateY(10px); }

  to { opacity: 1; transform: translateY(0); }

}

**OUTPUT:**



**Assignment 5 – Cart Page**

**Aim**

To develop a Cart Page that displays added events and calculates the total cost.

**Introduction**

The cart provides a summary of selected events, showing item details and total amount.

**Theory**

It uses React’s filtering and array methods to display only events whose IDs exist in the cart array. Total price is computed using reduce(). The design employs flexible boxes for layout and clear typography. Buttons allow removing events dynamically using onRemove(id).

**Conclusion**

The Cart Page illustrates state synchronization, filtering logic, and aggregation in React.

**Code**

import React from "react";

import "./CartPage.css";

export const CartPage = ({ events, cart, onRemove }) => {

  const items = events.filter((e) => cart.includes(e.id));

  const total = items.reduce((sum, e) => sum + e.price, 0);

  return (

    <div className="cart-container">

      <h2 className="title">🛍️ Your Cart</h2>

      {items.length === 0 ? (

        <p className="empty">Your cart is empty.</p>

      ) : (

        <div className="cart-list">

          {items.map((e) => (

            <div className="cart-item" key={e.id}>

              <img src={e.imageUrl} alt={e.name} />

              <div>

                <h3>{e.name}</h3>

                <p>{e.date}</p>

                <p>₹{e.price}</p>

                <button onClick={() => onRemove(e.id)}>Remove</button>

              </div>

            </div>

          ))}

          <div className="cart-summary">

            <h3>Total: ₹{total}</h3>

            <button onClick={() => alert("Checkout simulated!")}>

              Proceed to Checkout

            </button>

          </div>

        </div>

      )}

    </div>

  );

};

CSS

.cart-container {

  padding: 30px;

  text-align: center;

}

.cart-item {

  display: flex;

  background: white;

  border-radius: 12px;

  box-shadow: 0 3px 10px rgba(0, 0, 0, 0.2);

  margin: 15px 0;

  overflow: hidden;

}

.cart-item img {

  width: 120px;

  object-fit: cover;

}

.cart-item div {

  padding: 15px;

  text-align: left;

  flex: 1;

}

.cart-item button {

  background: #d9534f;

  color: white;

  border: none;

  border-radius: 8px;

  padding: 6px 12px;

  cursor: pointer;

}

.cart-item button:hover {

  background: #c9302c;

}

.cart-summary {

  margin-top: 20px;

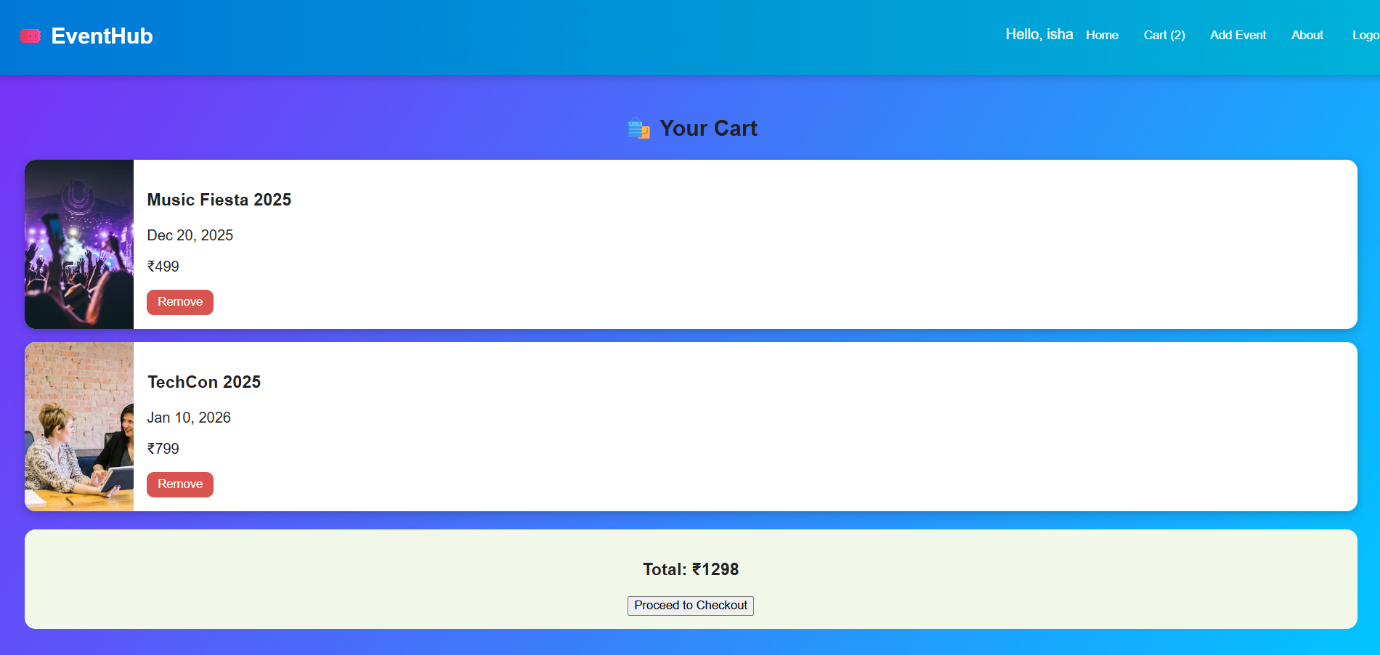
  background: #f1f8e9;

  padding: 15px;

  border-radius: 12px;

}

**OUTPUT:**



**Assignment 6 – About Page**

**Aim**

To create an informative About Page introducing the platform’s mission and team.

**Introduction**

The About Page strengthens brand identity and builds trust.

**Theory**

It contains structured sections (mission, team, contact) styled using gradients and grid layouts. The card-based team design uses hover animations and readable typography.

**Conclusion**

The About Page enhances UX and demonstrates static content structuring with React.

**Code**

import React from "react";

import "./AboutPage.css";

export const AboutPage = () => {

  return (

    <div className="about-wrapper">

      <div className="about-card">

        <h2>🌐 About EventHub</h2>

        <p>

          <strong>EventHub</strong> is your one-stop platform for discovering,

          booking, and managing exciting events happening around you.

        </p>

        <section className="mission">

          <h3>🎯 Our Mission</h3>

          <p>

            To make event discovery seamless and enjoyable. We bring

            communities together by promoting events that inspire, educate, and

            entertain.

          </p>

        </section>

        <section className="team">

          <h3>👥 Meet Our Team</h3>

          <div className="team-grid">

            <div className="team-member">

              <img

                src="https://images.unsplash.com/photo-1607746882042-944635dfe10e?auto=format&fit=crop&w=400&q=60"

                alt="Founder"

              />

              <h4>Isha Yadav</h4>

              <p>Founder & Project Lead</p>

            </div>

            <div className="team-member">

              <img

                src="https://images.unsplash.com/photo-1607746882042-944635dfe10e?auto=format&fit=crop&w=400&q=60"

                alt="Designer"

              />

              <h4>Neha Yadav</h4>

              <p>UI/UX Designer</p>

            </div>

            <div className="team-member">

  <imgsrc="https://images.unsplash.com/photo-1607746882042-944635dfe10e?auto=format&fit=crop&w=400&q=60"

                alt="Developer"

              />

              <h4>Priya Mehta</h4>

              <p>Frontend Developer</p>

            </div>

          </div>

        </section>

        <section className="contact">

          <h3>📩 Contact Us</h3>

          <p>

            Have questions or suggestions? We'd love to hear from you!

          </p>

          <a href="mailto:support@eventhub.com" className="contact-btn">

            ✉️ support@eventhub.com

          </a>

        </section>

      </div>

    </div>

  );

};

CSS

.about-wrapper {

  display: flex;

  justify-content: center;

  align-items: center;

  padding: 60px 20px;

  background: linear-gradient(135deg, #7b2ff7, #00c6ff);

  min-height: 85vh;

  animation: fadeIn 0.6s ease;

}

.about-card {

  background: #fff;

  border-radius: 20px;

  max-width: 900px;

  padding: 40px;

  box-shadow: 0 5px 20px rgba(0, 0, 0, 0.3);

  text-align: center;

}

.about-card h2 {

  color: #0078d7;

  margin-bottom: 15px;

}

.about-card p {

  color: #333;

  line-height: 1.6;

  margin-bottom: 20px;

  font-size: 15px;

}

/\* Mission Section \*/

.mission {

  background: #f0f9ff;

  border-radius: 12px;

  padding: 20px;

  margin: 25px 0;

}

/\* Team Section \*/

.team h3 {

  margin-bottom: 15px;

}

.team-grid {

  display: grid;

  grid-template-columns: repeat(auto-fit, minmax(220px, 1fr));

  gap: 20px;

  margin-top: 10px;

}

.team-member {

  background: linear-gradient(135deg, #e3f2fd, #e8eaf6);

  border-radius: 15px;

  padding: 15px;

  box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);

  transition: transform 0.3s ease;

}

.team-member:hover {

  transform: scale(1.05);

}

.team-member img {

  width: 100%;

  border-radius: 12px;

  margin-bottom: 10px;

}

/\* Contact Section \*/

.contact {

  margin-top: 30px;

  background: #f9fbe7;

  padding: 20px;

  border-radius: 12px;

}

.contact-btn {

  display: inline-block;

  margin-top: 10px;

  text-decoration: none;

  color: white;

  background: linear-gradient(90deg, #0078d7, #00b4d8);

  padding: 10px 20px;

  border-radius: 25px;

  font-weight: 500;

  transition: transform 0.3s ease;

}

.contact-btn:hover {

  transform: scale(1.05);

  background: linear-gradient(90deg, #00b4d8, #0078d7);

}

@keyframes fadeIn {

  from {

    opacity: 0;

    transform: translateY(15px);

  }

  to {

    opacity: 1;

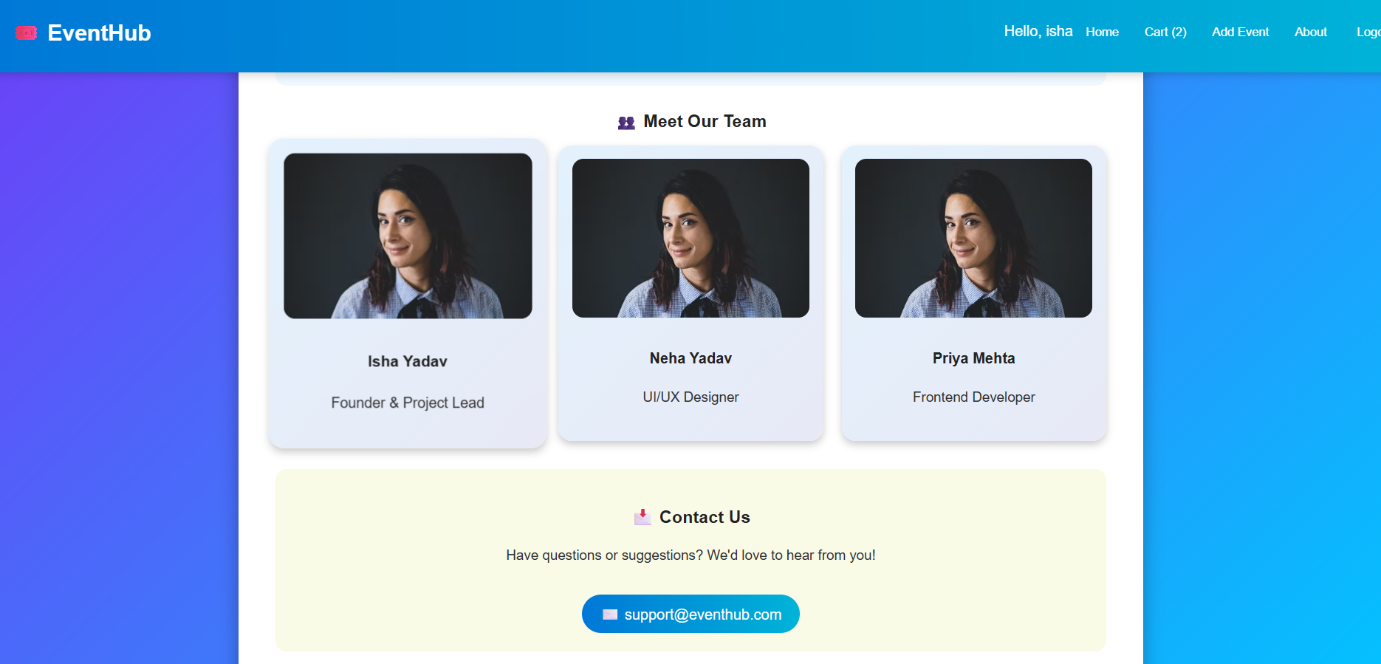
    transform: translateY(0);

  }

}

**OUTPUT:**





**Assignment 7 – Navigation Bar**

**Aim**

To implement a responsive top navigation bar with dynamic buttons and user greeting.

**Introduction**

Navigation is essential for accessibility and user flow.

**Theory**

The NavBar component uses props like isLoggedIn, cartCount, and navigate() for rendering. Conditional rendering shows different options for logged-in and guest users.

**Conclusion**

The NavBar improves usability through dynamic, role-based options.

**Code**

import React from "react";

import "./NavBar.css";

export const NavBar = ({ isLoggedIn, currentUser, cartCount, navigate, onLogout }) => (

  <nav className="navbar">

    <h1 onClick={() => navigate("home")}>🎟️ EventHub</h1>

    <div className="links">

      {isLoggedIn ? (

        <>

          <span>Hello, {currentUser.name}</span>

          <button onClick={() => navigate("home")}>Home</button>

          <button onClick={() => navigate("cart")}>Cart ({cartCount})</button>

          <button onClick={() => navigate("addEvent")}>Add Event</button>

          <button onClick={() => navigate("about")}>About</button> {/\* 👈 NEW \*/}

          <button onClick={onLogout}>Logout</button>

        </>

      ) : (

        <>

          <button onClick={() => navigate("login")}>Login</button>

          <button onClick={() => navigate("register")}>Register</button>

          <button onClick={() => navigate("about")}>About</button> {/\* 👈 NEW \*/}

        </>

      )}

    </div>

  </nav>

);

CSS

.navbar {

  background: linear-gradient(90deg, #0078d7, #00b4d8);

  color: white;

  padding: 12px 20px;

  display: flex;

  justify-content: space-between;

  align-items: center;

  position: fixed;

  top: 0;

  width: 100%;

  z-index: 5;

  box-shadow: 0 2px 10px rgba(0, 0, 0, 0.2);

}

.navbar h1 {

  font-size: 1.5em;

  cursor: pointer;

}

.links button {

  background: transparent;

  border: none;

  color: white;

  margin: 0 8px;

  cursor: pointer;

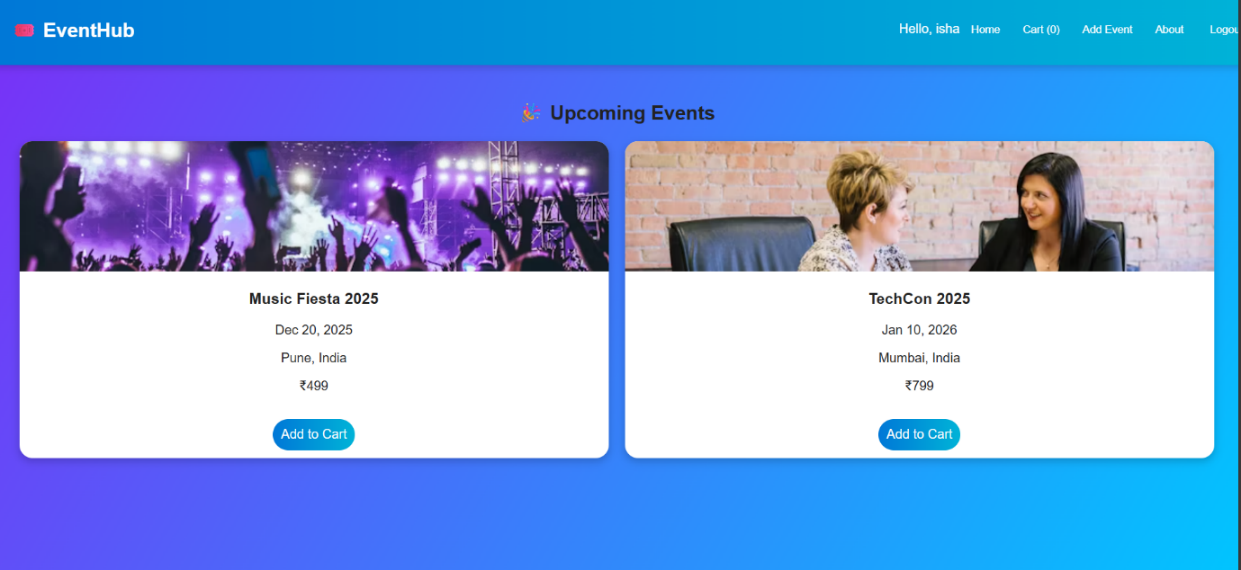
  font-weight: 500;

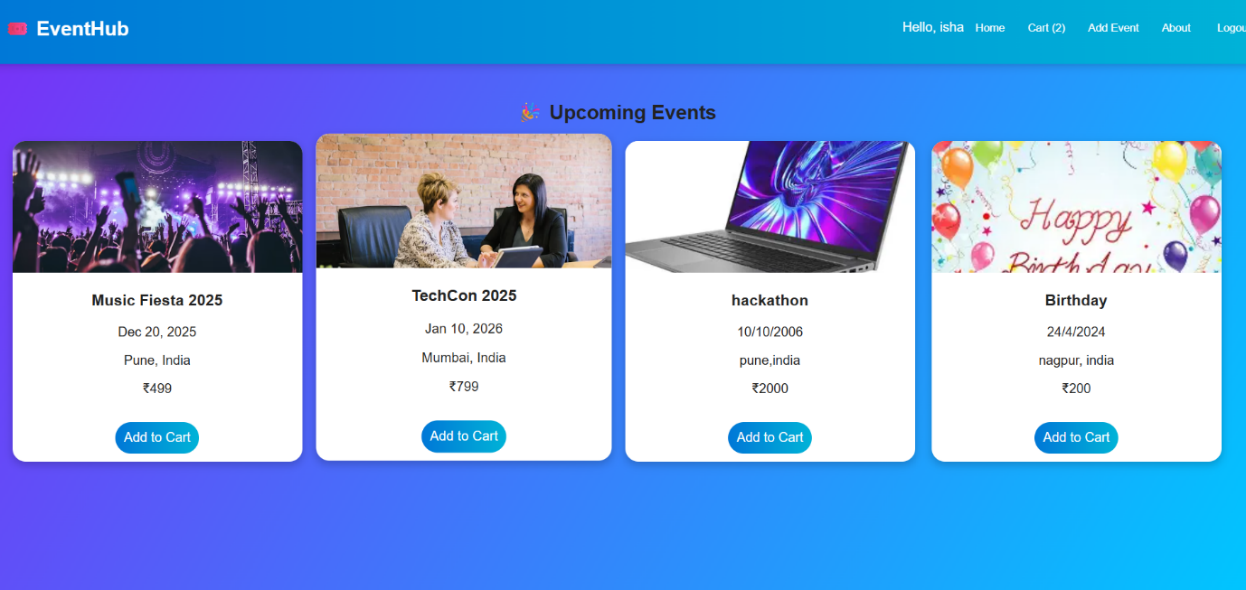
}

.links button:hover {

  color: #ffeb3b;

}

**OUTPUT:**



**Assignment 8 – MessageBox Component**

**Aim**

To create a floating message box for success, error, and info notifications.

**Introduction**

The MessageBox provides feedback to user actions like login success or cart updates.

**Theory**

React props (message and type) determine color and text. CSS animations fade in/out smoothly, and messages auto-clear using setTimeout.

**Conclusion**

The MessageBox improves UX with real-time user feedback.

**Code**

import React from "react";

import "./MessageBox.css";

export const MessageBox = ({ message, type }) => (

  <div className={`msg-box ${type}`}>{message}</div>

);

CSS

.msg-box {

  position: fixed;

  bottom: 25px;

  right: 25px;

  padding: 12px 20px;

  border-radius: 10px;

  font-weight: 500;

  color: white;

  animation: fadeIn 0.4s ease;

}

.msg-box.success { background: #4caf50; }

.msg-box.error { background: #f44336; }

.msg-box.info { background: #0078d7; }

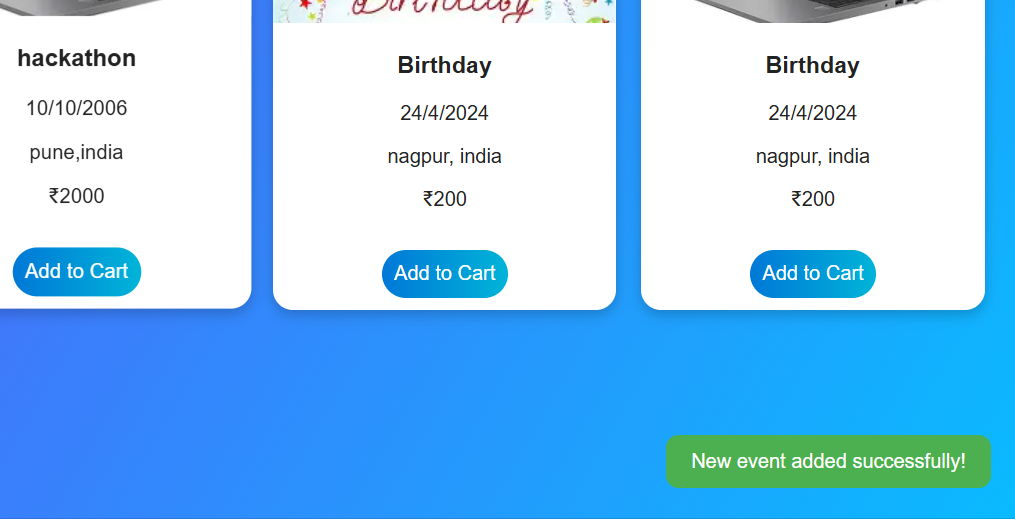
@keyframes fadeIn {

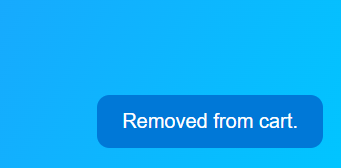
  from { opacity: 0; transform: translateY(10px); }

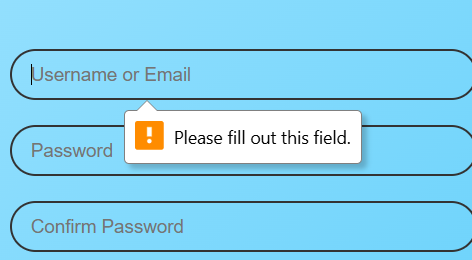
  to { opacity: 1; transform: translateY(0); }

}

**OUTPUT:**







**Assignment 9 – App.js**

**Aim**

To control global state and navigation through a single App component.

**Introduction**

App.js integrates all components and handles authentication, cart, and routing.

**Theory**

It manages global state using multiple useState() hooks and updates UI dynamically using conditional rendering (switch(currentView)). Each function updates relevant state portions immutably.

**Conclusion**

App.js is the central controller linking logic, data, and presentation layers cohesively.

**Code**

import React, { useState } from "react";

import "./App.css";

import { HomePage } from "./pages/HomePage";

import { AuthScreen } from "./pages/AuthScreen";

import { CartPage } from "./pages/CartPage";

import { AddEventPage } from "./pages/AddEventPage";

import { MessageBox } from "./components/MessageBox";

import { NavBar } from "./components/NavBar";    // ✅ Ensure file name matches exactly (NavBar.jsx)

import { initialEvents } from "./data/events";

import { AboutPage } from "./pages/AboutPage"; // ✅ New page

export default function App() {

  const [isLoggedIn, setIsLoggedIn] = useState(false);

  const [currentUser, setCurrentUser] = useState(null);

  const [currentView, setCurrentView] = useState("login");

  const [events, setEvents] = useState(initialEvents);

  const [cart, setCart] = useState([]);

  const [message, setMessage] = useState("");

  const [messageType, setMessageType] = useState("");

  // 🔹 Show temporary message (3 seconds)

  const showMessage = (msg, type = "info") => {

    setMessage(msg);

    setMessageType(type);

    setTimeout(() => setMessage(""), 3000);

  };

  // 🔹 Handle login/register success

  const handleAuthSuccess = (user) => {

    setIsLoggedIn(true);

    setCurrentUser(user);

    setCurrentView("home");

    showMessage(`Welcome, ${user.name}!`, "success");

  };

  // 🔹 Logout

  const handleLogout = () => {

    setIsLoggedIn(false);

    setCurrentUser(null);

    setCart([]);

    setCurrentView("login");

    showMessage("You have been logged out.", "info");

  };

  // 🔹 Add item to cart

  const handleAddToCart = (id) => {

    if (!isLoggedIn) {

      showMessage("Please log in to add events.", "error");

      setCurrentView("login");

      return;

    }

    if (cart.includes(id)) return showMessage("Already in cart!", "error");

    setCart([...cart, id]);

    showMessage("Added to cart!", "success");

  };

  // 🔹 Remove from cart

  const handleRemoveFromCart = (id) => {

    setCart(cart.filter((i) => i !== id));

    showMessage("Removed from cart.", "info");

  };

  // 🔹 Add new event

  const handleAddEvent = (newEvent) => {

    setEvents([...events, { id: events.length + 1, ...newEvent }]);

    showMessage("New event added successfully!", "success");

    setCurrentView("home");

  };

  // 🔹 Render current view

  const renderView = () => {

    switch (currentView) {

      case "login":

      case "register":

        return (

          <AuthScreen

            view={currentView}

            onAuthSuccess={handleAuthSuccess}

            switchView={setCurrentView}

          />

        );

      case "home":

        return <HomePage events={events} onAddToCart={handleAddToCart} />;

      case "cart":

        return (

          <CartPage

            events={events}

            cart={cart}

            onRemove={handleRemoveFromCart}

          />

        );

      case "addEvent":

        return <AddEventPage onAddEvent={handleAddEvent} />;

      case "about":

        return <AboutPage />;

      default:

        return null;

    }

  };

  return (

    <div className="app-container">

      <NavBar

        isLoggedIn={isLoggedIn}

        currentUser={currentUser}

        cartCount={cart.length}

        navigate={setCurrentView}

        onLogout={handleLogout}

      />

      <div className="content">{renderView()}</div>

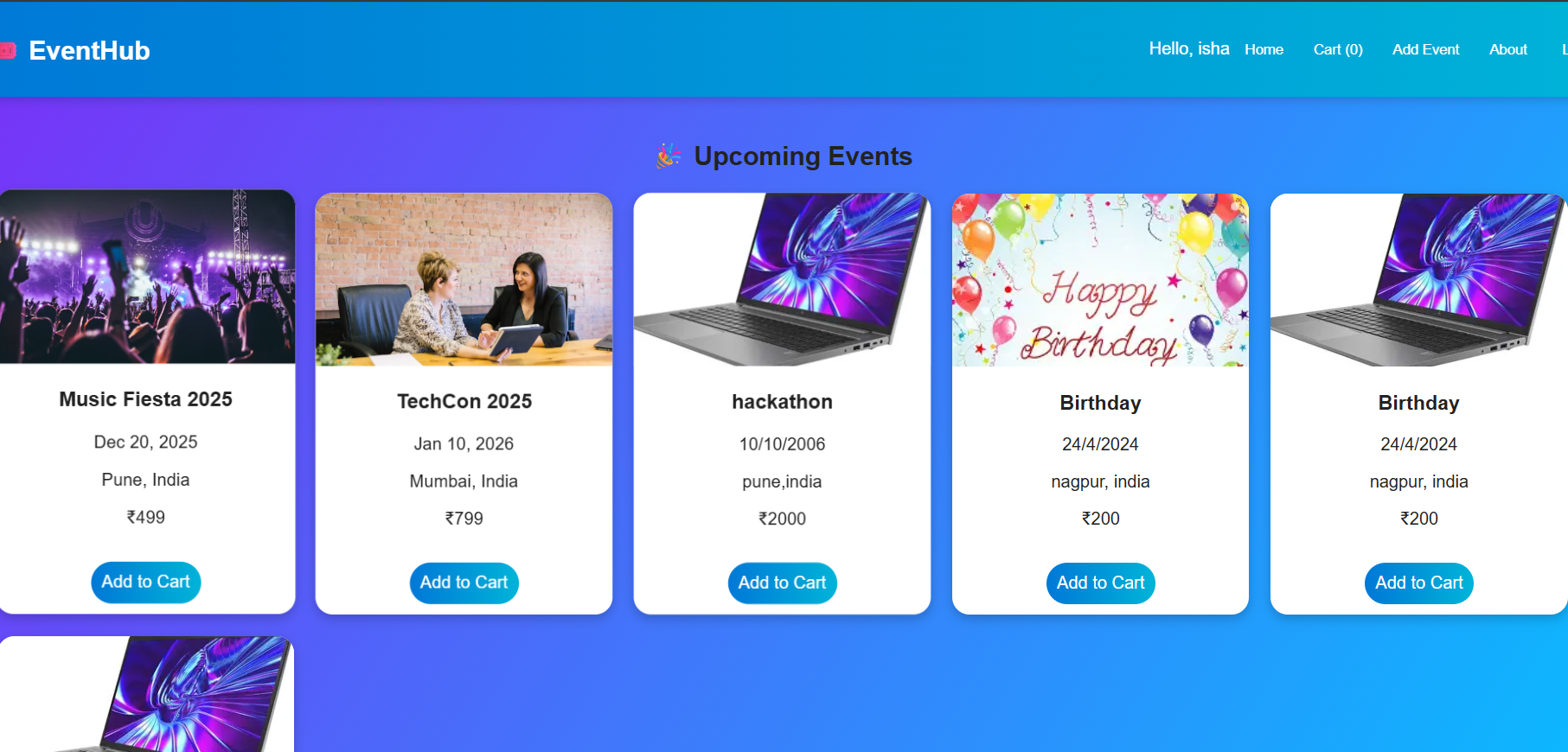
      {message && <MessageBox message={message} type={messageType} />}

    </div>

  );

}

**OUTPUT:**



**Assignment 10 – Styling and Responsiveness**

**Aim**

To create a visually consistent, responsive UI using CSS3, gradients, and grid layouts.

**Introduction**

CSS enhances user engagement and visual hierarchy. EventHub uses gradients, rounded corners, and animations.

**Theory**

CSS Grid and Flexbox create adaptable layouts. Media queries and relative units ensure usability across devices. Animations (@keyframes fadeIn) make transitions smoother. Consistent color palette (#0078d7, #00c6ff) maintains brand identity.

**Conclusion**

The responsive CSS framework ensures modern aesthetics and device adaptability in EventHub.

**Code**

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>

);

// If you want to start measuring performance in your app, pass a function

// to log results (for example: reportWebVitals(console.log))

// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals

reportWebVitals();

CSS

body {

  margin: 0;

  font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', 'Roboto', 'Oxygen',

    'Ubuntu', 'Cantarell', 'Fira Sans', 'Droid Sans', 'Helvetica Neue',

    sans-serif;

  -webkit-font-smoothing: antialiased;

  -moz-osx-font-smoothing: grayscale;

}

code {

  font-family: source-code-pro, Menlo, Monaco, Consolas, 'Courier New',

    monospace;

}

**OUTPUT:**

