MIT Art, Design and Technology University MIT School of Computing, Pune

Department of Information Technology

SUBJECT – FULL STACK FRONT END DEVELOPMENT

COURSE CODE – 23IT3001

Class – T.Y. SMAD

Name of Student

Attharv Banage – ADT24SOCB1514

Under the Guidance of

Prof. Rohini Bhosale

A.Y. 2025-2026 (SEM – I)

Assignment 1 : React Project Setup and Navigation

Aim :

To create a React project using **Vite (or create-react-app)**, set up routing using **react-router-dom**, and build a top navigation bar with links to different pages such as **Home, Products, Cart, Checkout, and Profile**.

Theory :

1. Introduction to React

React is an open-source JavaScript library developed by Facebook for building dynamic user interfaces.
It is mainly used to develop Single Page Applications (SPAs) where content updates without reloading the page.
React follows a component-based architecture, dividing the UI into reusable parts such as Navbar, Footer, and Page components.
It uses a Virtual DOM to efficiently update only the parts of the UI that have changed, improving performance.

---

2. React Project Setup

To start working with React, we use a setup tool like create-react-app or Vite.
Advantages of create-react-app / Vite setup:

1. Automatically configures Webpack/Babel for you.
2. Includes a local development server with hot reload.
3. Provides an optimized build configuration.
4. Supports JSX and ES6 syntax out of the box.

Example (using Vite):
npm create vite@latest dsa-learning --template react
cd dsa-learning
npm install
npm install react-router-dom bootstrap
npm run dev
This generates the basic folder structure and starts the development server.

---

3. React Router and SPA Concept

React apps are Single Page Applications — only one HTML page is loaded, and routing is handled on the client-side.
React Router enables navigation without reloading.
Core Components of React Router:

- BrowserRouter: Wraps the app and manages URL changes.

- Routes / Route: Define which component should load for each URL.
- Link / NavLink: Used instead of <a> tags for client-side navigation.

---

4. Navigation Bar (Navbar)

A navigation bar allows users to move between pages like Home, About, Cart, etc.
It is usually implemented as a reusable component and appears consistently across all pages.
Features of a Good Navbar:

1. Visible and consistent across all pages.
2. Contains brand logo and navigation links.
3. Is responsive on different screen sizes.
4. Uses React Router's <Link> or <NavLink> components for smooth navigation.

---

5. Components and Routing Structure

Each webpage (Home, Products, Cart, Checkout, Profile) is created as a React component and connected through routes.
When a user clicks a link, React Router dynamically updates the visible component without a page reload.

---

6. Working of Navigation in React

1. The entire app is wrapped in <BrowserRouter>.
2. Each screen (Home, Products, Cart, etc.) is registered as a <Route>.
3. The Navbar contains <NavLink>s to each route.
4. Clicking a link updates the URL and renders the matching component instantly.
5. Navigation happens client-side, giving a faster and smoother UX.

---

7. Advantages of Using React Router

1. Faster navigation — no full reload.
2. Component-based structure for easy maintenance.
3. Scalable — new routes can be added easily.
4. Improved performance — only the view changes.
5. SEO-friendly URLs — each route has a distinct path.

Code :

App.jsx

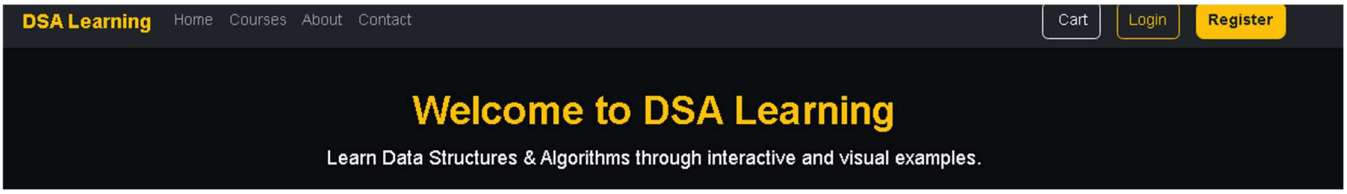
```
import { BrowserRouter as Router, Routes, Route } from "react-router-dom";
import Navbar from "./components/Navbar";

import Home from "./pages/Home";
import Products from "./pages/Products";
import Cart from "./pages/Cart";
import Checkout from "./pages/Checkout";
import Profile from "./pages/Profile";

function App() {
  return (
    <Router>
      <Navbar />
      <Routes>
        <Route path="/" element={<Home />} />
        <Route path="/products" element={<Products />} />
        <Route path="/cart" element={<Cart />} />
        <Route path="/checkout" element={<Checkout />} />
        <Route path="/profile" element={<Profile />} />
      </Routes>
    </Router>
  );
```

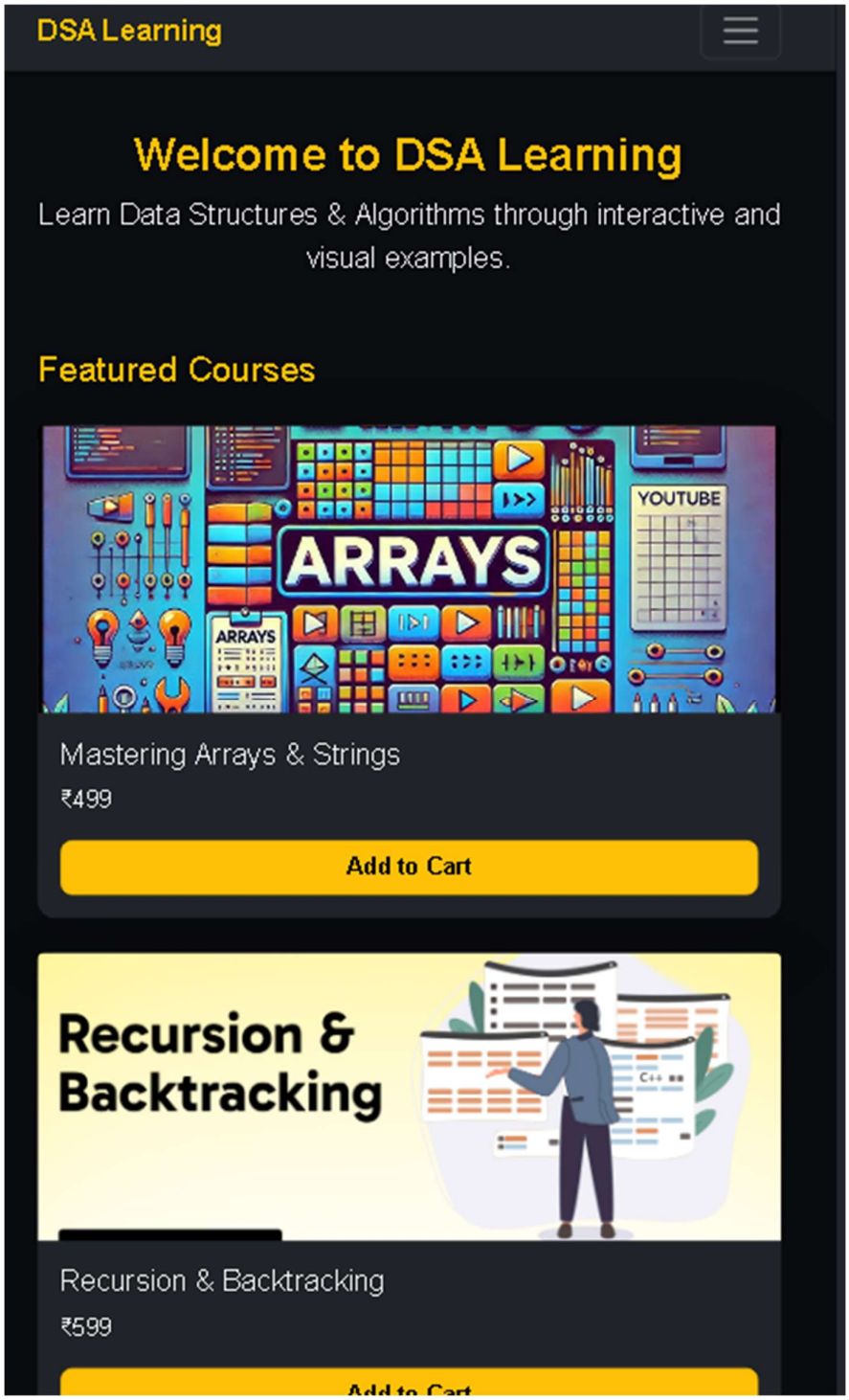
```
}

export default App;
```

Navbar.jsx

```
import { NavLink } from "react-router-dom";

export default function Navbar() {
  return (
    <nav className="navbar navbar-expand-lg navbar-dark bg-dark px-4">
      <NavLink className="navbar-brand fw-bold" to="/">
        <i className="fas fa-brain me-2"></i>DSA Learning
      </NavLink>

      <button
        className="navbar-toggler"
        type="button"
        data-bs-toggle="collapse"
        data-bs-target="#navbarNav"
      >
        <span className="navbar-toggler-icon"></span>
      </button>

      <div className="collapse navbar-collapse" id="navbarNav">
        <ul className="navbar-nav ms-auto">
          <li className="nav-item">
            <NavLink className="nav-link" to="/">Home</NavLink>
          </li>
```

```
        <li className="nav-item">
          <NavLink className="nav-link" to="/products">Products</NavLink>
        </li>
        <li className="nav-item">
          <NavLink className="nav-link" to="/cart">Cart</NavLink>
        </li>
        <li className="nav-item">
          <NavLink className="nav-link" to="/checkout">Checkout</NavLink>
        </li>
        <li className="nav-item">
          <NavLink className="nav-link" to="/profile">Profile</NavLink>
        </li>
      </ul>
    </div>
  </nav>
 );
}
```


DSA Learning
Home
Courses
About
Contact
Cart
Login
Register
Welcome to DSA Learning
Learn Data Structures & Algorithms through interactive and visual examples.


DSA Learning
Welcome to DSA Learning
Learn Data Structures & Algorithms through interactive and visual examples.
Featured Courses
ARRAYS
YOUTUBE
Mastering Arrays & Strings
₹499
Add to Cart
Recursion & Backtracking
Recursion & Backtracking
₹599

Conclusion :

We successfully created a React project and implemented routing using react-router-dom. A responsive navigation bar was built using JSX and Bootstrap, providing seamless navigation between pages in a single-page application architecture.

Assignment 2 : Home Page Development

Aim :

To design and develop the **Home Page layout** of a React application that includes featured products or categories and a functional **search bar** that filters products by name or category.

Theory :

1. Introduction

The Home Page is the first and most important part of any website or application.
It acts as a welcoming interface for users, highlights featured items, and provides quick navigation to other sections.
In this assignment, the Home Page is built using React components and styled using Bootstrap to ensure responsiveness and modern layout design.
It also includes a search functionality to dynamically filter the displayed products.

---

2. Concept of Components in React

React follows a component-based architecture, where each UI element is represented as an independent, reusable component.

In this assignment:

- **Home.jsx acts as a page-level component that manages product data and search logic.**
- **ProductCard.jsx is a reusable sub-component that displays individual product details such as name, category, price, and image.**

Each component manages its own state and behavior, promoting separation of concerns and modular design.

---

3. Designing the Home Page Layout

A well-structured Home Page includes:

1. **Banner/Header: Welcomes the user or introduces the brand.**
2. **Search Bar: Allows users to search products by name or category.**
3. **Featured Products Section: Displays a list of important or popular items.**
4. **Grid Layout: Uses Bootstrap's container, row, and col classes for alignment and spacing.**

This ensures a clean, organized, and responsive user interface suitable for all screen sizes.

---

4. Featured Products or Categories

The featured products section showcases the most relevant items.
In React, this is achieved using a product list array stored in state. Each product contains:

- **id – Unique identifier**
- **name – Product name**
- **category – Product category**
- **price – Product price**
- **image – Product image path**

The products are dynamically rendered using the .map() function, which allows easy scalability — new products can be added simply by updating the array.

---

5. Search Bar Functionality

The search bar enables real-time filtering of products based on user input.

Key React concepts used:

- **State (useState): To store the user's search term.**
- **Event handling (onChange): Updates the search term as the user types.**
- **Array filtering (filter()): Compares the search term with each product's name or category.**

This creates a dynamic search — product results update instantly without reloading the page.

---

6. Styling and Responsiveness

Bootstrap and CSS ensure a professional and responsive design:

- **container, row, col → Manage layout and spacing.**
- **form-control → Styles the search input field.**
- **card → Used to present each product neatly.**
- **Responsive grid ensures the design adapts to desktops, tablets, and mobile screens.**

Code :

Home.jsx

```
import { useState } from "react";

import ProductCard from "../components/ProductCard";
```

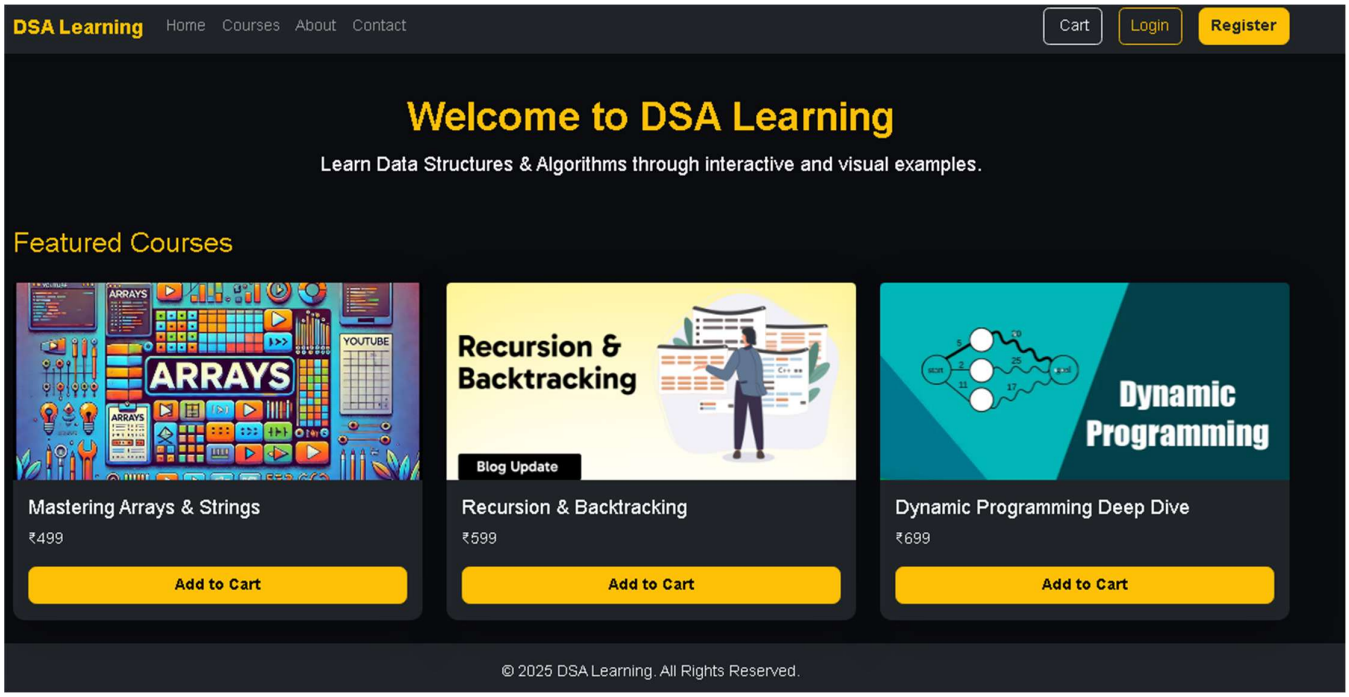
```
export default function Home() {
 const [searchTerm, setSearchTerm] = useState("");

 const products = [
  { id: 1, name: "Paracetamol", category: "Medicine", price: 40, image: "/med1.jpg" },
  { id: 2, name: "Face Mask", category: "Health Care", price: 25, image: "/med2.jpg" },
  { id: 3, name: "Vitamin C", category: "Supplement", price: 120, image: "/med3.jpg" },
  { id: 4, name: "Pain Relief Gel", category: "Medicine", price: 99, image: "/med4.jpg" },
 ];

 const filteredProducts = products.filter(
  (p) =>
   p.name.toLowerCase().includes(searchTerm.toLowerCase()) ||
   p.category.toLowerCase().includes(searchTerm.toLowerCase())
 );

 return (
  <div className="container mt-5 pt-5">
   <h2 className="text-center mb-4">Welcome to DSA Learning Store</h2>

   {/* Search Bar */}
   <div className="text-center mb-4">
    <input
     type="text"
     className="form-control w-50 mx-auto"
     placeholder="Search by name or category..."
     value={searchTerm}
     onChange={(e) => setSearchTerm(e.target.value)}
    />
   </div>

   {/* Featured Products */}
   <div className="row justify-content-center">
```

```
      {filteredProducts.length > 0 ? (
        filteredProducts.map((product) => (
          <div className="col-md-3 col-sm-6 mb-4" key={product.id}>
            <ProductCard product={product} />
          </div>
        ))
      ) : (
        <p className="text-center text-muted">No products found.</p>
      )}
    </div>
  </div>
 );
}
```

ProductCard.jsx

```
export default function ProductCard({ product }) {
 return (
  <div className="card shadow-sm border-0 text-center">
    <img
      src={product.image}
      className="card-img-top p-3"
      alt={product.name}
      style={{ height: "180px", objectFit: "contain" }}
    />
    <div className="card-body">
      <h5 className="card-title">{product.name}</h5>
      <p className="card-text text-muted">{product.category}</p>
      <p className="fw-bold text-primary">₹{product.price}</p>
    </div>
  </div>
 );
}
```

Output :



Conclusion :

We successfully developed a Home Page in React that includes:

- A functional search bar for real-time filtering,
- A featured products section dynamically rendered using components, and
- A responsive layout built with Bootstrap and React's component-based architecture.

This implementation demonstrates React's power in creating dynamic, modular, and interactive user interfaces.

Assignment 3 : Fetch product data from array or static JSON file

---

Aim :

To understand and implement **Routing** in a React application using the react-router-dom library, and to design a **Courses Page** that displays a list of DSA topics with their details (title, difficulty, duration, and image) and an option to add them to the learning cart.

Theory :

1. Introduction to Routing

Routing in React enables navigation between multiple views or components in a Single Page Application (SPA) without full-page reloads.
Instead of fetching new pages from the server, React Router swaps components on the client-side, providing faster transitions and a smoother experience.

In this project, routing connects pages like Home, Courses, Cart, Profile, and Contact.

---

2. What is React Router?

react-router-dom is a routing library used to manage page navigation in React.
It defines different URL paths and maps them to components.

Component	Description
BrowserRouter	Wraps the app and manages history and URL changes.
Routes	Groups multiple route definitions.
Route	Maps a URL path to a React component.
Link / NavLink	Used for navigation without page reloads.
useNavigate	Hook for programmatic navigation.

---

3. How Routing Works

1. **Wrap the entire app in <BrowserRouter>.**
2. **Define routes using <Routes> → <Route path="/courses" element={<Courses/>} />.**
3. **Navbar links use <NavLink> for smooth transitions.**
4. **When a link is clicked, React Router updates the URL and renders the relevant component dynamically.**

This gives the illusion of multiple pages while keeping everything within one HTML document — the essence of a SPA.

---

4. DSA Courses Data

For this assignment, course data is fetched from a static JavaScript array (or optionally a JSON file). Each course object contains:

- **id**
- **name**
- **category (e.g., Sorting, Graphs, DP)**
- **level (Beginner, Intermediate, Advanced)**
- **duration (in hours)**
- **image (course thumbnail)**

The data is rendered dynamically using .map() so you can easily add more courses without editing UI code.

---

5. Search and Filter Feature

The search bar filters courses by name or category:

- **useState stores the search term.**
- **onChange updates it as the user types.**
- **.filter() checks if the term matches the course name or category.**

This allows instant, real-time filtering and improves user interactivity.

---

6. Advantages of Routing and Dynamic Rendering

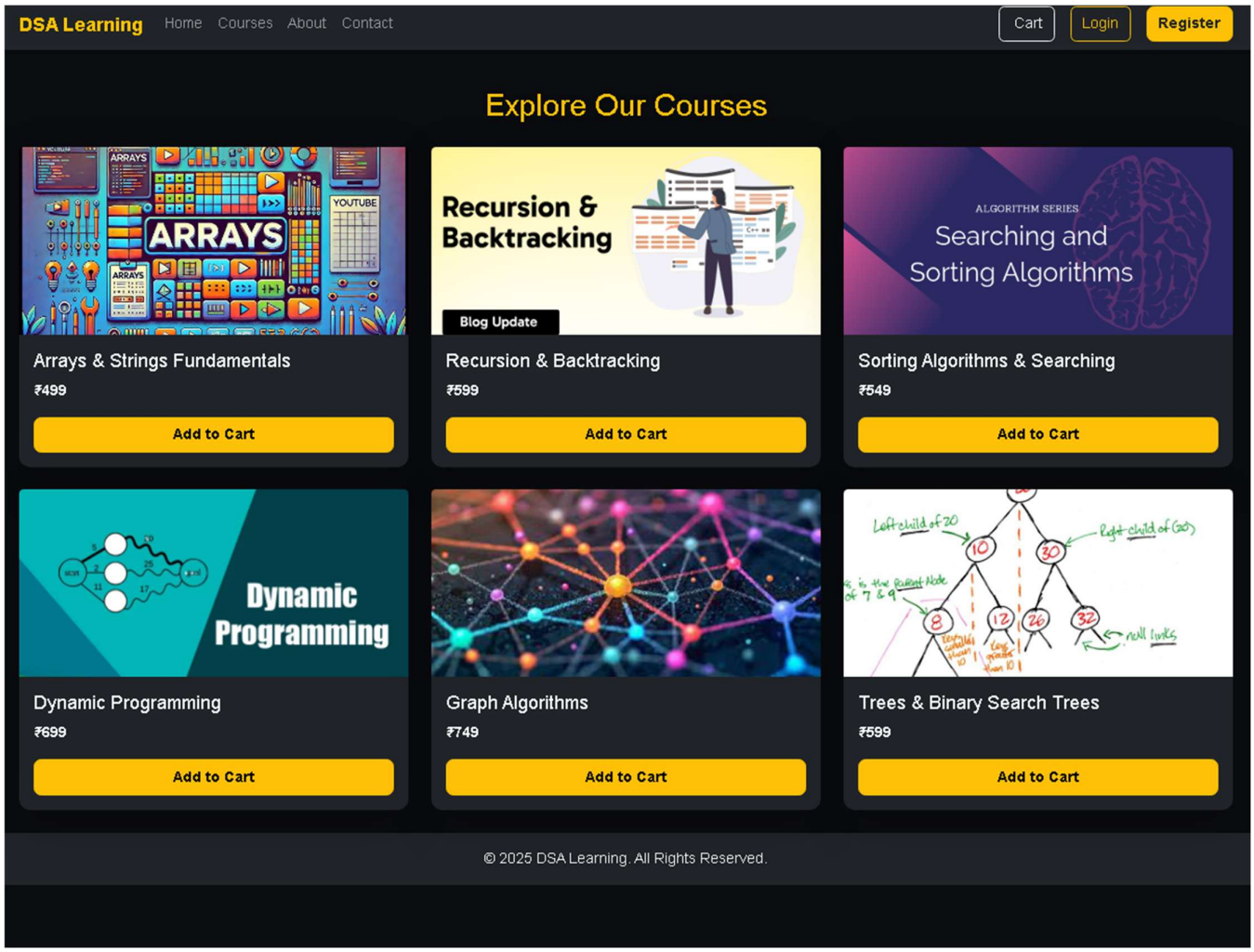
- **Fast navigation (no reloads)**
- **Better UX and SEO-friendly URLs**
- **Modular page organization**
- **Scalable architecture (easy to add more courses or routes)**
- **Real-time updates via React state**

Code :

```
import { useState } from "react";
import { useNavigate } from "react-router-dom";
import CourseCard from "../components/CourseCard";

export default function Courses() {
  const [searchTerm, setSearchTerm] = useState("");
  const navigate = useNavigate();

  const courses = [
    { id: 1, name: "Arrays and Strings", category: "Fundamentals", level: "Beginner", duration: "2h", image: "/arrays.jpg" },
    { id: 2, name: "Recursion Basics", category: "Core Concepts", level: "Beginner", duration: "1.5h", image: "/recursion.jpg" },
    { id: 3, name: "Sorting Algorithms", category: "Algorithms", level: "Intermediate", duration: "3h", image: "/sorting.jpg" },
    { id: 4, name: "Stacks and Queues", category: "Data Structures", level: "Intermediate", duration: "2.5h", image: "/stack.jpg" },
    { id: 5, name: "Linked Lists", category: "Data Structures", level: "Intermediate", duration: "2h", image: "/linkedlist.jpg" },
    { id: 6, name: "Binary Trees", category: "Trees", level: "Advanced", duration: "3.5h", image: "/tree.jpg" },
    { id: 7, name: "Graphs", category: "Graphs", level: "Advanced", duration: "4h", image: "/graph.jpg" },
    { id: 8, name: "Dynamic Programming", category: "Algorithms", level: "Advanced", duration: "5h", image: "/dp.jpg" },
    { id: 9, name: "Searching Techniques", category: "Algorithms", level: "Intermediate", duration: "2h", image: "/search.jpg" },
    { id: 10, name: "Hashing", category: "Data Structures", level: "Intermediate", duration: "1.5h", image: "/hash.jpg" },
    { id: 11, name: "Greedy Algorithms", category: "Algorithms", level: "Advanced", duration: "2.5h", image: "/greedy.jpg" },
    { id: 12, name: "Graph Traversals", category: "Graphs", level: "Advanced", duration: "3h", image: "/traversal.jpg" },
  ];

  const filteredCourses = courses.filter(
    (c) =>
      c.name.toLowerCase().includes(searchTerm.toLowerCase()) ||
      c.category.toLowerCase().includes(searchTerm.toLowerCase())
  );

  return (
    <div className="container mt-5 pt-5">
      <h2 className="text-center mb-4 fw-bold">Explore DSA Courses</h2>

      {/* Search Bar */}
      <div className="text-center mb-4">
        <input
```

```
        type="text"
        className="form-control w-50 mx-auto"
        placeholder="Search by topic or category..."
        value={searchTerm}
        onChange={(e) => setSearchTerm(e.target.value)}
      />
    </div>

    {/* Courses Grid */}
    <div className="row justify-content-center">
      {filteredCourses.length > 0 ? (
        filteredCourses.map((course) => (
          <div
            className="col-md-3 col-sm-6 mb-4"
            key={course.id}
            onClick={() => navigate("/cart")}
            style={{ cursor: "pointer" }}
          >
            <CourseCard course={course} />
          </div>
        ))
      ) : (
        <p className="text-center text-muted">No courses found.</p>
      )}
    </div>
  </div>
 );
}
```

Output :



Conclusion :

We successfully implemented **Routing** in the **DSA Learning React Application** using react-router-dom and fetched **DSA course data** from a static array.
The Courses page dynamically renders all topics with search and navigation functionality, demonstrating the **component-based** and **interactive** nature of React.

Assignment 4 : Products Page

Aim :

To design and develop a **Courses Page** in React that dynamically displays a list of DSA topics with complete information such as image, name, description, difficulty, and duration, along with an **"Add to Learning List"** button.
The page also includes a **functional search bar** to filter courses based on name or description.

Theory :

1. Introduction

The Courses Page is one of the core sections of the DSA Learning platform.
It lists all the available DSA topics in an organized grid layout so that learners can browse, view topic details, and add them to their learning cart.

In React, this page is implemented as a functional component that manages state and filtering logic, and uses reusable subcomponents for displaying each course card.

---

2. Component-Based Architecture

React's modular design divides the UI into independent, reusable components.
For the Courses Page:

- **Courses.jsx — acts as the main page-level component that handles search logic and dynamic rendering.**
- **CourseCard.jsx — acts as a subcomponent responsible for presenting individual course details (title, difficulty, duration, and image).**

This structure improves scalability and maintainability — any UI changes in course presentation require updates only in one place.

---

3. Featured DSA Topics

The featured topics section highlights important and frequently learned DSA concepts.
In React, this is achieved by storing a static array or JSON data inside the component, where each object has:

- **id**
- **name**
- **desc (description)**
- **difficulty**

- **duration**
- **image**

These are displayed dynamically using the .map() method, making it easy to add new topics later without editing UI logic.

---

4. Search Bar Functionality

The search bar enables learners to quickly find specific DSA topics.

It uses:

- **A state variable (searchTerm) to store user input.**
- **The onChange event to update the state as the user types.**
- **The .filter() method to compare input with course names and descriptions.**

This creates a real-time search experience where results update instantly, improving usability and interactivity.

---

5. Routing and Navigation

Navigation to the Courses Page occurs via the Navbar using React Router's <NavLink> component. When a user clicks "Courses", React Router navigates to /courses without reloading the entire page, maintaining the single-page flow of the application.

This client-side routing is key to React's Single Page Application (SPA) performance and smooth transitions.

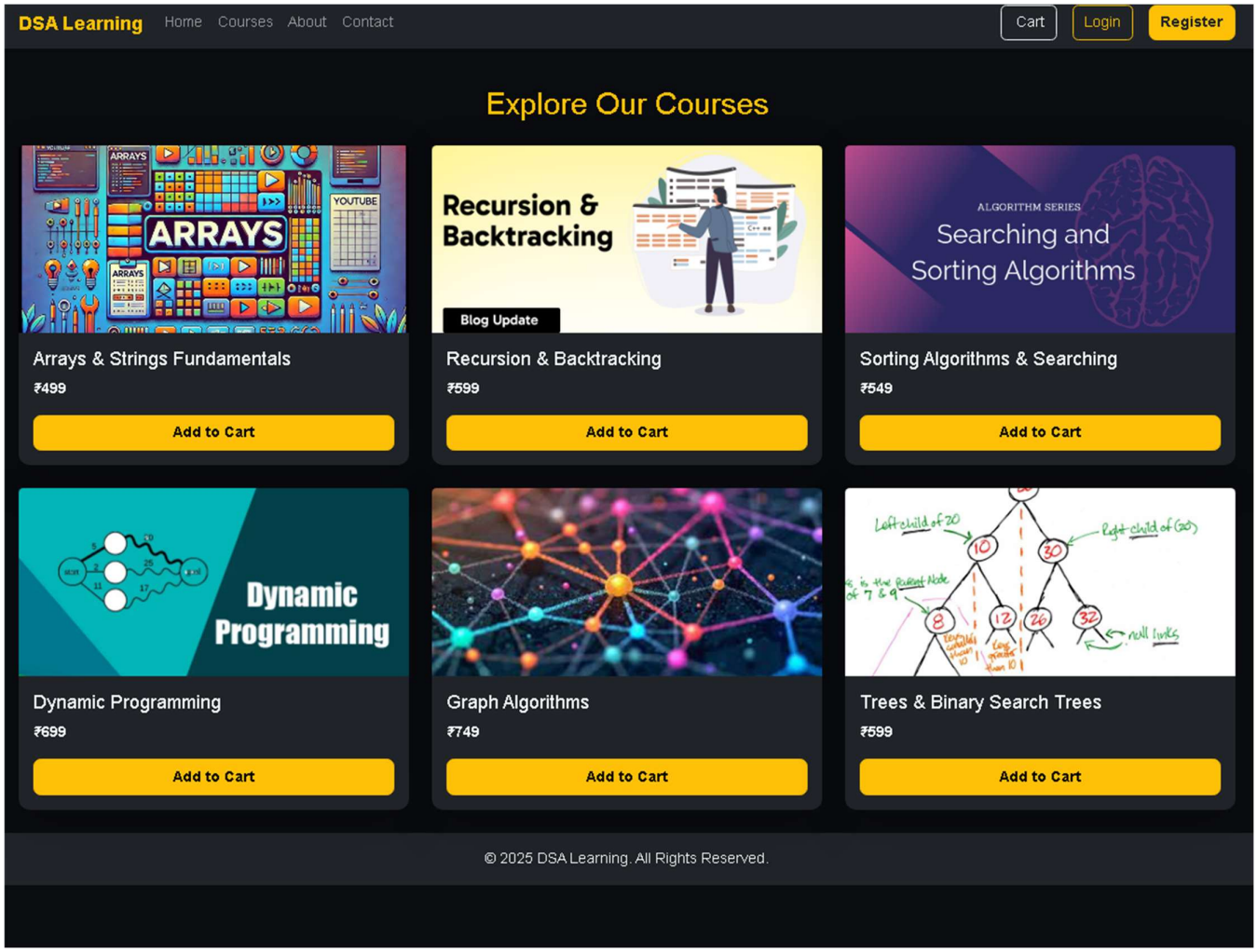
Code :

```
import { useState } from "react";
import CourseCard from "../components/CourseCard";

function Courses() {
  const [searchTerm, setSearchTerm] = useState("");

  const courses = [
    { id: 1, name: "Arrays and Strings", desc: "Learn the fundamentals of array manipulation and string algorithms.", difficulty: "Beginner", duration: "2h", image: "/arrays.jpg" },
    { id: 2, name: "Recursion and Backtracking", desc: "Master recursive thinking and backtracking techniques.", difficulty: "Intermediate", duration: "3h", image: "/recursion.jpg" },
    { id: 3, name: "Sorting Algorithms", desc: "Understand sorting logic like Bubble, Merge, and Quick sort.", difficulty: "Intermediate", duration: "2.5h", image: "/sorting.jpg" },
    { id: 4, name: "Linked Lists", desc: "Explore singly, doubly, and circular linked lists.", difficulty: "Intermediate", duration: "3h", image: "/linkedlist.jpg" },
    { id: 5, name: "Stacks and Queues", desc: "Learn about stack-based and queue-based data management.", difficulty: "Intermediate", duration: "2.5h", image: "/stack.jpg" },
```

```
    { id: 6, name: "Binary Trees",
desc: "Learn how to construct and
traverse binary trees.", difficulty:
"Advanced", duration: "3.5h",
image: "/tree.jpg" },

    { id: 7, name: "Graphs and
Traversals", desc: "Understand
graph theory, BFS, DFS, and
shortest path algorithms.",
difficulty: "Advanced", duration:
"4h", image: "/graph.jpg" },

    { id: 8, name: "Dynamic
Programming", desc: "Solve
optimization problems using
overlapping subproblems.",
difficulty: "Advanced", duration:
"5h", image: "/dp.jpg" },

    { id: 9, name: "Hashing", desc:
"Learn how to use hash tables and
maps efficiently.", difficulty:
"Intermediate", duration: "2h",
image: "/hash.jpg" },

    { id: 10, name: "Greedy
Algorithms", desc: "Understand
greedy logic for optimization
problems.", difficulty: "Advanced",
duration: "3h", image:
"/greedy.jpg" },

    { id: 11, name: "Divide and
Conquer", desc: "Break problems
into smaller subproblems and
conquer recursively.", difficulty:
"Intermediate", duration: "3h",
image: "/dac.jpg" },

    { id: 12, name: "Graph
Applications", desc: "Apply graph
theory in real-world scenarios.",
difficulty: "Advanced", duration:
"3h", image: "/applications.jpg" },

  ];


  const filteredCourses =
courses.filter(

    (c) =>

c.name.toLowerCase().includes(searchTerm.toL
owerCase()) ||


c.desc.toLowerCase().includes(searchTerm.toLo
werCase())

  );


  return (

    <div className="container mt-5 pt-5">

      <h2 className="text-center mb-4 fw-
bold">Explore DSA Topics</h2>


      {/* Search Bar */}

      <div className="text-center mb-4">

        <input

          type="text"

          className="form-control w-50 mx-auto"

          placeholder="Search by topic or
description..."

          value={searchTerm}

          onChange={(e) =>
setSearchTerm(e.target.value)}

        />

      </div>


      {/* Course Grid */}

      <div className="row justify-content-
center">

        {filteredCourses.length > 0 ? (

          filteredCourses.map((course) => (

            <div className="col-md-3 col-sm-6 mb-
4" key={course.id}>

              <CourseCard course={course} />

            </div>

          ))
```

```
      ) : (
        <p className="text-center text-muted">No topics found.</p>
      )}
    </div>
  </div>
  );
}

export default Courses;
```

Output :



Conclusion :

We successfully implemented the **DSA Courses Page** in the DSA Learning React application.
This page dynamically displays a list of DSA topics using React's **state**, **props**, and **map()** functions, includes a functional **search bar**, and uses **React Router** for seamless navigation.
The component-based design ensures **code reusability**, **scalability**, and a **smooth learning experience** for users.

Assignment 5 : Cart Page

Aim :

To design and develop a **Learning Cart Page** in React that allows users to view, manage, and modify selected DSA courses added from the Courses Page.

Theory :

1. Introduction

The Learning Cart Page is an essential part of the DSA Learning platform.
It acts as a temporary collection of the courses a learner intends to study later.
This page displays all selected DSA topics and provides functionalities such as:

- **Updating learning hours,**
- **Removing courses from the cart, and**
- **Viewing the total duration of all selected topics.**

All these features are managed efficiently using React's state and hooks.

---

2. Data Flow in the Cart Page

When the user clicks "Add to Learning List" on any course card, that course's data is passed to the Cart Page through React Router's navigate() function using state objects.
The Cart Page then receives the data via the useLocation() hook.

For example:

```
navigate("/cart", { state: { course } });
```

This enables seamless client-side data flow without requiring external storage or a backend.

---

3. State Management

React's useState() hook is used to manage cart data — including selected courses, their duration, and quantity.

Example:

```
const [cartItems, setCartItems] = useState(
  location.state?.course ? [{ ...location.state.course, studyHours: 1 }] : []
);
```

This stores all selected courses in memory and allows updates such as modifying study hours or removing topics.

---

4. Component Structure

Component	Responsibility
CourseCard.jsx	Contains the "Add to Learning List" button that sends data to Cart.
Cart.jsx	Displays selected courses and allows updating/removing them.
App.jsx	Handles routing between /courses and /cart.

This modular structure promotes reusability, scalability, and maintainability.

---

5. Quantity (Hours) Handling

Each selected course includes a numeric input to update study hours.
React dynamically updates the total learning duration using the onChange event.

---

6. Deletion of Courses

Removing a course is implemented using the .filter() method:

setCartItems((prevItems) => prevItems.filter((item) => item.id !==

Code :

```
import React, { useState } from "react";
import { useLocation, useNavigate } from "react-router-dom";

function Cart() {
  const location = useLocation();
  const navigate = useNavigate();

  const [cartItems, setCartItems] = useState(
    location.state?.course ? [{ ...location.state.course, studyHours: 1 }] : []
  );

  const handleHoursChange = (id, newHours) => {
    setCartItems((prevItems) =>
      prevItems.map((item) =>
        item.id === id ? { ...item, studyHours: parseInt(newHours) } : item
      )
    );
  };

  const handleDelete = (id) => {
    setCartItems((prevItems) => prevItems.filter((item) => item.id !== id));
  };
```

```
  const calculateTotalHours = () =>
    cartItems.reduce(
      (total, item) => total + (item.studyHours || 1),
      0
    );

  const handleStartLearning = () => {
    navigate("/profile", { state: { cartItems, total: calculateTotalHours() } });
  };

  return (
    <div className="container mt-5 pt-5">
      <h2 className="text-center mb-4 fw-bold text-primary">Your Learning Cart</h2>

      {cartItems.length === 0 ? (
        <p className="text-center text-muted">Your learning cart is empty.</p>
      ) : (
        <>
          <div className="table-responsive">
            <table className="table table-striped align-middle">
              <thead className="table-dark">
                <tr>
                  <th>Course</th>
                  <th>Difficulty</th>
                  <th>Hours</th>
                  <th>Update</th>
                  <th>Remove</th>
                </tr>
              </thead>
              <tbody>
                {cartItems.map((item) => (
                  <tr key={item.id}>
                    <td>
                      <div className="d-flex align-items-center">
                        <img
                          src={item.image}
                          alt={item.name}
                          width="60"
                          height="60"
                          className="me-3 rounded"
                        />
                        <div>
                          <strong>{item.name}</strong>
                          <p className="text-muted small mb-0">{item.desc}</p>
                        </div>
                      </div>
                    </td>
                    <td>{item.difficulty}</td>
                    <td>
                      <input
                        type="number"
                        className="form-control w-50"
                        min="1"
                        value={item.studyHours}
                        onChange={(e) => handleHoursChange(item.id, e.target.value)}
                      />
                    </td>
```

```
            <td>{item.studyHours} hr(s)</td>
            <td>
              <button
                className="btn btn-danger btn-sm"
                onClick={() => handleDelete(item.id)}
              >
                Remove
              </button>
            </td>
          </tr>
        ))}
      </tbody>
    </table>
  </div>

  <div className="text-end mt-4">
    <h4>Total Learning Hours: {calculateTotalHours()} hrs</h4>
    <button
      className="btn btn-success mt-3 px-4 fw-bold"
      onClick={handleStartLearning}
    >
      Start Learning
    </button>
  </div>
</>
)}
</div>
);
}

export default Cart;
```

```
import React, { useState } from "react"; import { useLocation, useNavigate } from "reactrouter-dom"; //  Correct import

function Cart() {  const location = useLocation();
 const navigate = useNavigate(); //  For navigation

 const [cartItems, setCartItems] = useState(
  location.state?.product ? [{ ...location.state.product, orderQty: 1 }] : []
 );

 const handleQuantityChange = (id, newQty) => {  setCartItems((prevItems) =>
prevItems.map((item) =>
   item.id === id ? { ...item, orderQty: parseInt(newQty) } : item
  )
 );
};

 const handleDelete = (id) => {
  setCartItems((prevItems) => prevItems.filter((item) => item.id !== id));
};
 const calculateTotal = () => cartItems.reduce(
  (total, item) => total + item.price * (item.orderQty || 1),
```

```
    0
  );

 const handleCheckout = () => {
   navigate("/checkout", { state: { cartItems, total: calculateTotal() } });
 };

 return (
   <div className="container mt-5 pt-5">
     <h2 className="text-center mb-4 fw-bold text-primary">  Your Cart</h2>
      {cartItems.length === 0 ? (
       <p className="text-center text-muted">Your cart is empty.</p>
      ) : (
       <>
        <div className="table-responsive">
          <table className="table table-striped align-middle">
            <thead className="table-dark">
              <tr>
                <th>Product</th>
```

```
    <th>Price (₹)</th>
    <th>Quantity</th>
    <th>Total (₹)</th>
                                <th>Remove</th>
  </tr>
</thead>
<tbody>
onChange={(e) =>
handleQuantityChange(item.id,
                    e.target.value)
                                    }
                 className="form-control w-50"
                                    />
                                </td>
            <td>₹{item.price *
```

```
{cartItems.map((item) => (                                                    item.orderQty}</td>
 <tr key={item.id}>
                                                                                <td>
  <td>
                                                                                <button
                <div className="d-flex align-items                  className="btn btn-danger btn-sm"
center">                  onClick={() =>
   <img handleDelete(item.id)}
    src={item.image}
                                                                                >
    alt={item.name}
                                                                                 Delete
    width="60"
                                                                               </button>
    height="60"
                                                                                </td>
    className="me-3 rounded"
                                                                                </tr>
   />
                                                                                ))}
   <div>
                                                                              </tbody>
    <strong>{item.name}</strong>
                                                                              </table>
    <p className="text-muted small
mb-0">{item.desc}</p>
                                                                  <div className="text-end mt-4">
   </div>
                                                                        <h4>Total Amount:
  </div>
                                                                    ₹{calculateTotal()}</h4>
 </td>
                                                                        <button
                          <td>{item.price}</td>             className="btn btn-success mt-3 px-4
 <td> fw-bold"
  <input              onClick={handleCheckout}                type="number"
>                 min="1"             Proceed to Checkout                max={item.quantity}
</button>                 value={item.orderQty}          </div>
    </div>   );
  </> }
 )}
</div> export default Cart;
```

Output :

Conclusion :

We successfully developed the **Learning Cart Page** for the **DSA Learning React Application**.
It dynamically displays selected DSA courses, allows modification of learning hours, removal of topics, and shows total study duration using React's **state**, **props**, and **hooks**.
This demonstrates effective **state management**, **component reusability**, and **client-side navigation** in a modern React SPA.

Assignment 6 : Checkout Page

Aim :

To create a **Checkout Page** that collects user information and displays a summary of selected **DSA courses**, allowing users to confirm enrollment through a "Place Enrollment" button.

Theory :

1. Introduction

The Checkout Page represents the final step of the learning selection process in the DSA Learning Application.
It allows learners to:

- **Enter their personal and contact details,**
- **Review their selected courses, and**
- **Confirm their enrollment.**

This page demonstrates how to integrate form handling, state management, and data transfer between React components.

---

2. Role of the Checkout Page

The Checkout Page performs three main roles:

1. **Collecting User Details:**
   Captures the learner's information — name, email, city, and preferred learning schedule.
2. **Reviewing Enrollment Summary:**
   Displays all the selected courses with their durations and total hours.
3. **Confirming Enrollment:**
   Provides a button to finalize the learning plan, simulating an actual enrollment system.

---

3. Data Flow from Cart Page

When the learner clicks "Start Learning" on the Cart Page, all selected course data (including names, durations, and total study hours) are passed to the Checkout Page via React Router's state mechanism:

navigate("/checkout", { state: { cartItems, total } });

The Checkout component retrieves this data using useLocation(), ensuring smooth client-side transitions.

---

4. Form Handling and Validation

React's useState() hook stores user input for each form field.
Before final submission, the system verifies that all required fields are filled.
If any are missing, an alert notifies the learner.

---

5. Enrollment Summary

A detailed order (enrollment) summary is displayed alongside the form, listing:

- **Course name**
- **Difficulty level**
- **Selected learning hours**
- **Subtotal for each topic**
- **Grand total learning time**

This provides clarity and transparency before confirming enrollment.

---

6. Place Enrollment Confirmation

When the learner clicks the "Place Enrollment" button:

1. **The system validates the input fields.**
2. **Displays a confirmation message for successful registration.**
3. **Redirects the user to the Home Page for continuity.**

---

7. React Hooks and Concepts Used

Concept	Description
useState()	Manages user input and dynamic updates.
useLocation()	Retrieves data passed from the Cart Page.
useNavigate()	Redirects the learner after successful enrollment.
Form Handling	Captures and validates user input.

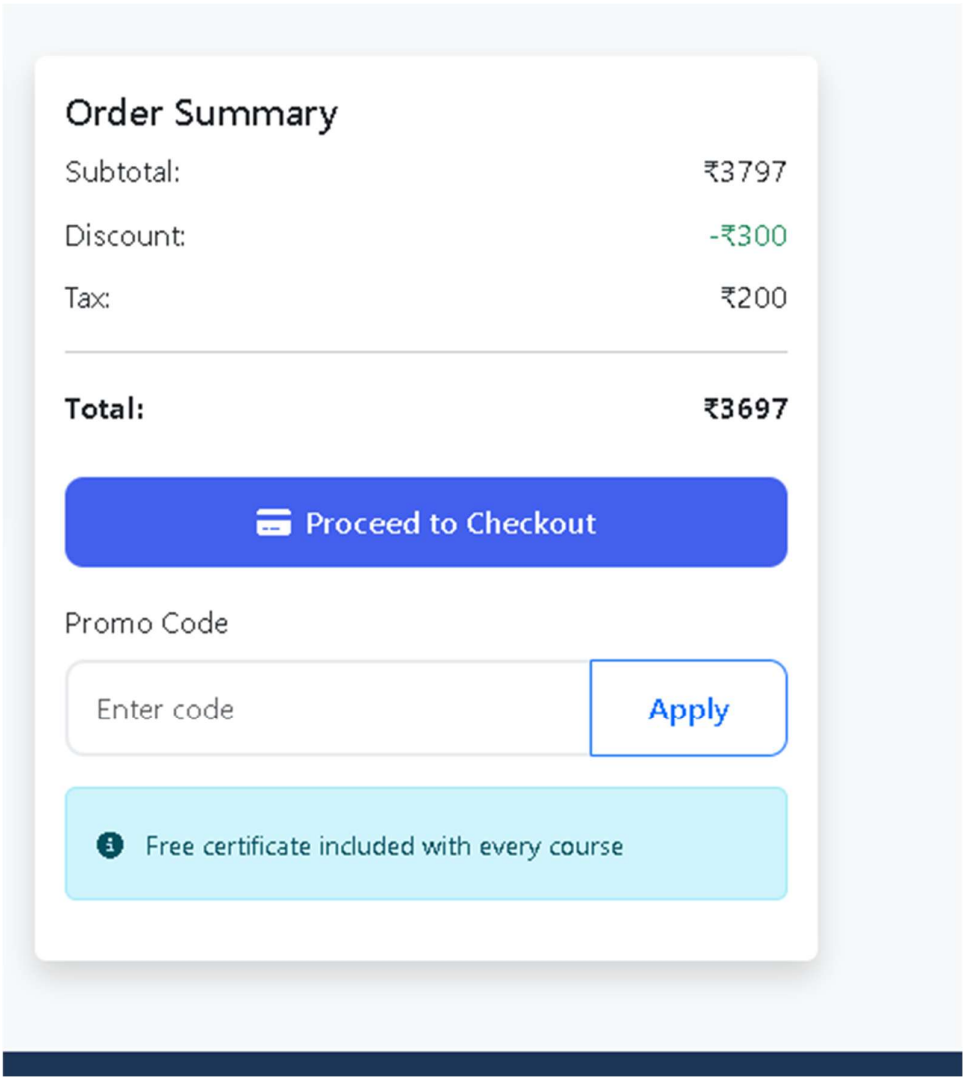
Code :

```
import React, { useState } from "react";
import { useLocation, useNavigate } from "react-router-dom";

function Checkout() {
  const location = useLocation();
  const navigate = useNavigate();
  const { cartItems = [], total = 0 } = location.state || {};

  const [formData, setFormData] = useState({
    name: "",
    email: "",
```

```
  city: "",
  schedule: "",
 });

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

 const handlePlaceOrder = () => {
  const { name, email, city, schedule } = formData;

  if (!name || !email || !city || !schedule) {
   alert("Please fill all details before confirming enrollment.");
   return;
  }

  alert("🎉 Enrollment confirmed! Welcome to DSA Learning.");
  navigate("/");
 };

 return (
  <div className="container mt-5 pt-5">
   <h2 className="text-center mb-4 fw-bold text-primary">
    Checkout & Enrollment
   </h2>

   <div className="row">

    {/* Learner Details Form */}
    <div className="col-md-6 mb-4">
     <h4 className="mb-3">Learner Information</h4>
     <form>
      <div className="mb-3">
       <label className="form-label">Full Name</label>
       <input
        type="text"
        name="name"
        className="form-control"
        value={formData.name}
        onChange={handleChange}
       />
      </div>

      <div className="mb-3">
       <label className="form-label">Email Address</label>
       <input
        type="email"
        name="email"
        className="form-control"
        value={formData.email}
        onChange={handleChange}
       />
      </div>

      <div className="mb-3">
       <label className="form-label">City</label>
```

```
          <input
            type="text"
            name="city"
            className="form-control"
            value={formData.city}
            onChange={handleChange}
          />
        </div>

        <div className="mb-3">
          <label className="form-label">Preferred Learning Schedule</label>
          <select
            name="schedule"
            className="form-select"
            value={formData.schedule}
            onChange={handleChange}
          >
            <option value="">Select a Schedule</option>
            <option value="Morning">Morning (8 AM - 12 PM)</option>
            <option value="Afternoon">Afternoon (1 PM - 5 PM)</option>
            <option value="Evening">Evening (6 PM - 9 PM)</option>
          </select>
        </div>
      </form>
    </div>

    {/* Order (Enrollment) Summary */}
    <div className="col-md-6">
      <h4 className="mb-3">Enrollment Summary</h4>
      {cartItems.length === 0 ? (
        <p>No courses selected.</p>
      ) : (
        <ul className="list-group mb-3">
          {cartItems.map((item) => (
            <li
              key={item.id}
              className="list-group-item d-flex justify-content-between align-items-center"
            >
              <div>
                <strong>{item.name}</strong>
                <p className="text-muted small mb-0">
                  Hours: {item.studyHours}
                </p>
              </div>
              <span>{item.difficulty}</span>
            </li>
          ))}
          <li className="list-group-item d-flex justify-content-between">
            <strong>Total Learning Hours</strong>
            <strong>{total} hrs</strong>
          </li>
        </ul>
      )}
```

```
        <button
          className="btn btn-success w-100 py-2 fw-bold"
          onClick={handlePlaceOrder}
        >
          Place Enrollment
        </button>
      </div>
    </div>
  );
}

export default Checkout;
```



Conclusion :

We successfully implemented the **Checkout Page** for the **DSA Learning React Application**.
It integrates form handling, validation, and data transfer from the Cart Page while providing a real-world simulation of an enrollment process using **React Router**, **state**, and **hooks**.

Assignment 7 : Profile Page

Aim :

To design and implement a **Profile Page** in the DSA Learning React Application that displays all learner details collected during registration.

Theory :

1. Introduction

The Profile Page is an essential component of modern, user-focused applications.
In the DSA Learning Platform, it provides learners with a personalized dashboard where they can view their stored registration details.
Using React state management and LocalStorage, user information remains available even after page reloads, ensuring persistence and personalization.

---

2. Purpose of the Profile Page

The Profile Page serves several key purposes:

1. **Displays all the learner's details collected during registration.**
2. **Provides an overview of personal, contact, and educational information.**
3. **Acts as a central hub for managing or updating user data.**
4. **Enhances user experience by maintaining personalization across sessions.**

---

3. Data Handling Mechanism

In the DSA Learning App:

1. **User information is entered during registration.**
2. **The data is stored using LocalStorage, allowing it to persist locally.**
3. **The Profile Page retrieves and displays this data using React's dynamic rendering.**
4. **The application can verify if a user is logged in and then show their details.**

This eliminates the need for a backend while simulating a realistic data flow process.

---

4. Component Design and Structure

Section	Details Displayed
Personal Information	Full Name, Gender, Date of Birth
Contact Information	Email, Phone Number, Address, City, Country
Educational/Professional Info	Current Occupation or Study Level
Health/Extra Info (Optional)	Allergies or Notes if included
Logout Functionality	Clears session and redirects to Home/Login

This modular structure helps organize data efficiently and makes the Profile Page easily extensible.

---

5. React Concepts Used

Concept	Description
useState()	Stores and manages dynamic user data.
useEffect()	Fetches stored data from LocalStorage when the page loads.
useNavigate()	Redirects the user to another page after logout.
LocalStorage	Retains user information across sessions.

Code :

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

function Profile() {
  const navigate = useNavigate();
  const [user, setUser] = useState(null);

  // Fetch user data from localStorage
  useEffect(() => {
    const storedUser =
localStorage.getItem("userData");
    if (storedUser) {
      setUser(JSON.parse(storedUser));
    } else {
alert("Please log in first!");
      navigate("/login");
    }
  }, [navigate]);

  // Logout function
  const handleLogout = () => {
    localStorage.removeItem("userData");
    alert("Logged out successfully!");
    navigate("/");
  };

  return (
    <div className="container mt-5 pt-5">
```

```
{user ? (
 <>
  <h2 className="text-center fw-bold text-primary mb-4">
   My Profile
  </h2>

  {/* Personal Information */}
  <div className="card shadow-lg p-4 mb-4">
   <h4 className="text-secondary">Personal Information</h4>
   <p><strong>Name:</strong> {user.firstName} {user.lastName}</p>
   <p><strong>Gender:</strong> {user.gender}</p>
   <p><strong>Date of Birth:</strong> {user.dob}</p>
   <p><strong>Blood Group:</strong> {user.bloodGroup}</p>
  </div>

  {/* Contact Information */}
  <div className="card shadow-lg p-4 mb-4">
   <h4 className="text-secondary">Contact Information</h4>
   <p><strong>Email:</strong> {user.email}</p>
   <p><strong>Phone:</strong> {user.phone}</p>
   <p><strong>Alternate Phone:</strong> {user.altPhone}</p>
   <p><strong>Emergency Contact:</strong> {user.emergencyContact}</p>
   <p><strong>Address:</strong> {user.address}, {user.city}, {user.state}, {user.country} - {user.pincode}</p>
  </div>

  {/* Educational / Professional Information */}
  <div className="card shadow-lg p-4 mb-4">
   <h4 className="text-secondary">Educational & Professional Details</h4>
   <p><strong>Occupation:</strong> {user.occupation}</p>
   <p><strong>Organization:</strong> {user.organization}</p>
   <p><strong>Annual Income:</strong> ₹{user.annualIncome}</p>
  </div>

  {/* Health Information */}
  <div className="card shadow-lg p-4 mb-4">
   <h4 className="text-secondary">Health Information</h4>
   <p><strong>Allergies:</strong> {user.allergies}</p>
   <p><strong>Medical History:</strong> {user.medicalHistory}</p>
  </div>

  <div className="text-center">
   <button
    className="btn btn-danger px-4 fw-bold"
    onClick={handleLogout}
   >
    Logout
   </button>
  </div>
 </>
) : (
```

```
      <p className="text-center mt-5">Loading
profile...</p>
    )}
  </div>
  );
}

export default Profile;
```

Conclusion:

We successfully implemented the **Profile Page** for the **DSA Learning Application**.
This page retrieves and displays user data stored during registration using **React Hooks** and **LocalStorage**, ensuring data persistence and user personalization within a **Single Page Application** structure.

Assignment 8 : State Management

Aim :

To implement centralized **state management** in the **DSA Learning React Application** using the **Context API**, ensuring that course selections, cart data, and user session information are globally accessible and persistent across all components.
Additionally, the assignment aims to restrict access to the **Checkout page** for non-logged-in users, introducing authentication-based state control for secure navigation.

Theory :

1. Introduction

In a React application like **DSA Learning**, multiple pages (Home, Courses, Cart, Checkout, Profile) need to share and update common data — for example, selected courses or logged-in user information.
Managing this data separately in each component using useState becomes complex as the app grows.

To solve this, **React Context API** provides a way to store global state and share it efficiently across all components without prop drilling.
This assignment demonstrates how to use **Context API** to manage the **cart system** globally and integrate **authentication control** for secure operations like checkout.

---

2. Role of Context API in State Management

The **Context API** acts as a **global memory** for the app.
In the DSA Learning app:

- Learners can add courses to their cart.
- The cart data is accessible from any page.
- Updates in one component (like removing a course) instantly reflect in others.
- Data persists across navigations using **localStorage**.

Without Context API, the data would be lost during navigation or page refresh, but with centralized state, the entire app stays in sync.

---

3. Key Components Used

Component	Purpose
CartContext	Defines global cart state and methods like add, remove, update.

Component	Purpose
CartProvider	Wraps the entire app and provides access to global data.
Cart Page	Displays selected courses and updates cart dynamically.
Checkout Page	Accesses cart data via Context and validates user login before confirming enrollment.
LocalStorage	Maintains persistent data even after reload.

---

4. Authentication-Based Restriction

To simulate real-world functionality:

- Only **logged-in users** can proceed to checkout.
- The system checks for a flag (isLoggedIn) in localStorage.
- If not logged in, the user is redirected to /login.
- The current cart data is temporarily saved as pendingCheckout and restored after login, ensuring a seamless experience.

---

5. Data Flow in Context API

1. **User selects DSA courses** → added to the cart using addToCart().
2. **CartContext** stores these selections globally.
3. **Cart page** reflects live updates via the same shared context.
4. **Checkout page** consumes data from context and verifies login status.
5. **LocalStorage** ensures cart persistence after refresh or logout.

This creates a **single source of truth**, ensuring consistency and reliability across the application.

Code :

```
import React, { useContext } from "react";
import { useNavigate } from "react-router-dom";
import { CartContext } from "../context/CartContext";

function Cart() {
  const { cartItems, updateQuantity, removeFromCart } = useContext(CartContext);
  const navigate = useNavigate();

  const calculateTotal = () =>
    cartItems.reduce(
      (total, item) => total + item.price * (item.orderQty || 1),
      0
    );

  const handleCheckout = () => {
    const loggedIn = localStorage.getItem("isLoggedIn");

    if (!loggedIn) {
      alert("Please log in before proceeding to checkout.");
      localStorage.setItem(
        "pendingCheckout",
        JSON.stringify({ cartItems })
      );
      navigate("/login");
    } else {
      navigate("/checkout", { state: { cartItems, total: calculateTotal() } });
    }
  };

  return (
    <div className="container mt-5 pt-5">
      <h2 className="text-center mb-4 fw-bold text-primary">Your DSA Cart</h2>

      {cartItems.length === 0 ? (
        <p className="text-center text-muted">Your cart is empty.</p>
      ) : (
        <>
          <div className="table-responsive">
            <table className="table table-striped align-middle">
              <thead className="table-dark">
                <tr>
                  <th>Course</th>
                  <th>Price (₹)</th>
                  <th>Duration</th>
                  <th>Quantity</th>
                  <th>Total (₹)</th>
                  <th>Remove</th>
                </tr>
              </thead>
              <tbody>
                {cartItems.map((item) => (
                  <tr key={item.id}>
                    <td>
                      <strong>{item.name}</strong>
```

```
                <p className="text-muted small mb-0">{item.desc}</p>
              </td>
              <td>₹{item.price}</td>
              <td>{item.duration}</td>
              <td>
                <input
                  type="number"
                  min="1"
                  className="form-control w-50"
                  value={item.orderQty}
                  onChange={(e) =>
                    updateQuantity(item.id, e.target.value)
                  }
                />
              </td>
              <td>₹{item.price * item.orderQty}</td>
              <td>
                <button
                  className="btn btn-danger btn-sm"
                  onClick={() => removeFromCart(item.id)}
                >
                  Delete
                </button>
              </td>
            </tr>
          ))}
        </tbody>
      </table>
    </div>

    <div className="text-end mt-4">
      <h4>Total Amount: ₹{calculateTotal()}</h4>
      <button
        className="btn btn-success mt-3 px-4 fw-bold"
        onClick={handleCheckout}
      >
        Proceed to Checkout
      </button>
    </div>
  </>
 )}
</div>
);
}

export default Cart;
```

Output :

Conclusion:

We successfully implemented centralized **state management** using **React Context API** in the **DSA Learning Application**.
The cart system is globally accessible, persistent, and synchronized across all pages.
Authentication checks ensure only verified users can proceed to checkout.
This demonstrates how Context API provides a **lightweight, scalable, and maintainable alternative** to Redux for medium-sized React applications.

Assignment 9 : Routing and Component Structure

Aim :

The aim of this assignment is to implement **efficient navigation** and **modular code design** in a **React-based DSA Learning Application** using **React Router** for page transitions and reusable components for the user interface.
The goal is to create a **Single Page Application (SPA)** structure where pages such as **Home**, **Courses**, **Cart**, **Checkout**, **Login**, and **Profile** can be accessed seamlessly without page reloads.
Additionally, this assignment emphasizes **component reusability** and **maintainable structure**, improving readability and scalability.

Theory :

1. Importance of Routing in React

In traditional web development, moving from one page to another required loading new HTML files — causing delays and loss of app state.
React Router eliminates this by enabling **client-side routing**, where navigation happens virtually without reloading the entire page.

In this project, **React Router DOM** is used to handle navigation between:

- / → Home page
- /courses → List of available DSA courses
- /cart → User's selected courses
- /checkout → Payment/enrollment details
- /profile → User information
- /login and /register → Authentication screens

This ensures that users experience smooth transitions while maintaining app state (like cart items) across pages.

---

2. React Router Structure

React Router provides several core components:

- **BrowserRouter**: Wraps the entire application and enables routing.
- **Routes**: Groups multiple routes.
- **Route**: Maps a path (like /cart) to a component (like <Cart />).
- **Link**: Replaces traditional <a> tags for navigation without reloading.

In the **DSA Learning App**, the **Navbar** component contains navigation links to all pages, while the Router setup ensures that only the relevant component re-renders when a user navigates.

---

3. Component-Based Modularity

React applications are built using **modular, reusable components**.
Each component serves a unique purpose and can be reused across multiple pages.
For example:

- **Navbar.jsx** → Persistent top navigation
- **ProductCard.jsx** → Reusable course card
- **Footer.jsx** → Common footer for all pages
- **CartItem.jsx** → Displays each selected course in the cart

This modularity ensures:
☑ Code reusability
☑ Better organization
☑ Easier debugging and maintenance

---

4. Protected Routing

Certain pages (like **Checkout** or **Profile**) should only be accessible to logged-in users.
By checking the isLoggedIn flag from localStorage, unauthorized users are redirected to the login page.
This adds **realistic authentication flow** and user session control.

---

5. Benefits

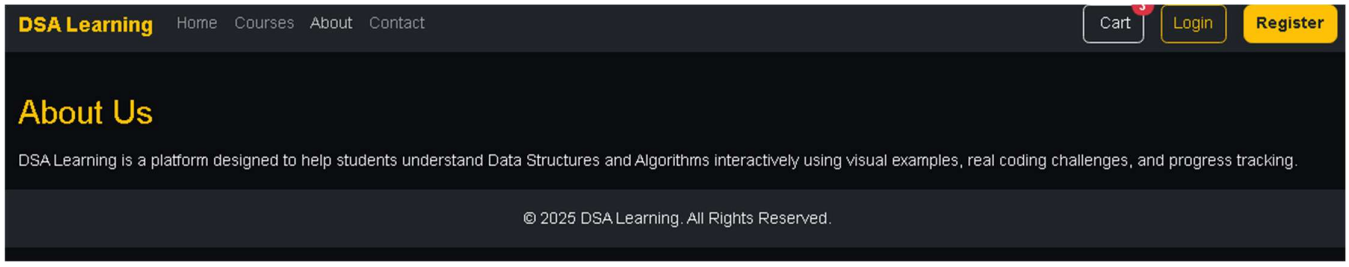
- Instant navigation without reloads (SPA behavior)
- Global state maintained across all routes
- Clean and modular code structure
- Enhanced scalability for adding new pages

Code :

```
 import { BrowserRouter as Router, Routes, Route } from "react-router-dom";

import Navbar from "./components/Navbar";

import Footer from "./components/Footer";

import Home from "./pages/Home";

import Courses from "./pages/Courses";

import Cart from "./pages/Cart";

import Checkout from "./pages/Checkout";
```

```
import Profile from "./pages/Profile";
import Login from "./pages/Login";
import Register from "./pages/Register";
import Contact from "./pages/Contact";

function App() {
  return (
    <Router>
      <Navbar />
      <div className="main-content" style={{ paddingTop: "70px", minHeight: "80vh" }}>
        <Routes>
          <Route path="/" element={<Home />} />
          <Route path="/courses" element={<Courses />} />
          <Route path="/cart" element={<Cart />} />
          <Route path="/checkout" element={<Checkout />} />
          <Route path="/profile" element={<Profile />} />
          <Route path="/contact" element={<Contact />} />
          <Route path="/login" element={<Login />} />
          <Route path="/register" element={<Register />} />
        </Routes>
      </div>
      <Footer />
    </Router>
  );
}

export default App;
```

Output :



Conclusion :

We successfully implemented **Routing and Component Structure** in the **DSA Learning Application** using **React Router**.
The application now behaves as a **Single Page Application**, allowing users to navigate instantly between pages.
By organizing the app into **modular, reusable components**, we achieved clean architecture and scalability.
This assignment demonstrates how **React Router + Component-based structure** form the foundation of modern, maintainable front-end applications.

Assignment 10 : Full React Application

Aim :

The aim of this assignment is to set up a **React project** using **Create React App (CRA)** or **Vite**, structure the project into modular components, and implement **client-side routing** using **React Router DOM**.
It also integrates **Context API** for cart management and **localStorage-based authentication** for persistence across sessions.

Theory :

1. All About React

React is a component-based JavaScript library used to build dynamic and interactive user interfaces. Setting up with Create React App gives a complete ready-to-use environment — including Webpack, Babel, ESLint, and a structured folder system for scalability.

Routing in React is handled via React Router DOM, where the BrowserRouter component enables navigation without full page reloads.
Routes and Route map URLs to components, and Link is used instead of <a> tags to ensure smooth navigation within a single-page application (SPA).

---

2. Home Page

The Home page displays featured courses and allows searching through them dynamically using useState.
Each course is displayed using a reusable CourseCard component, leveraging Bootstrap grid for responsiveness.

---

3. Courses (Products) Page

This page lists all available DSA courses with details like title, price, and difficulty level.
It includes a search bar to filter courses by name or description and an Add to Cart button that uses the global CartContext to manage cart state.

---

4. Cart Page

The Cart page displays selected items, allowing users to change quantities or remove items.
The total amount updates instantly without reloading, demonstrating React's two-way binding and state reactivity.

---

5. Checkout Page

Implements a controlled form for collecting user details like name, email, and address.
Data validation ensures fields are filled before proceeding.
After order confirmation, the app shows a success alert and clears the cart.

---

6. Registration, Login, and Profile

The user registration and login system use localStorage for persistence.
Once registered, user details are saved, and on login, a session flag (isLoggedIn) is set.
The Profile page fetches this stored data and displays it neatly, demonstrating state persistence and conditional rendering.

---

7. State Management (Context API)

The CartContext holds global state for cart operations.
Functions like addToCart, removeFromCart, and updateQuantity are accessible to all components.
This eliminates prop drilling and makes the app cleaner and scalable.

Code :

```
import React from "react";

import { BrowserRouter as Router, Routes, Route } from "react-router-dom";

import Navbar from "./components/Navbar";

import Home from "./pages/Home";

import Courses from "./pages/Courses";

import Cart from "./pages/Cart";

import Checkout from "./pages/Checkout";

import Login from "./pages/Login";

import Register from "./pages/Register";

import Profile from "./pages/Profile";

import Contact from "./pages/Contact";

import { CartProvider } from "./context/CartContext";
```

```
function App() {
  return (
    <CartProvider>
      <Router>
        <Navbar />
        <Routes>
          <Route path="/" element={<Home />} />
          <Route path="/courses" element={<Courses />} />
          <Route path="/cart" element={<Cart />} />
          <Route path="/checkout" element={<Checkout />} />
          <Route path="/login" element={<Login />} />
          <Route path="/register" element={<Register />} />
          <Route path="/profile" element={<Profile />} />
          <Route path="/contact" element={<Contact />} />
        </Routes>
      </Router>
    </CartProvider>
  );
}

export default App;
```

```
import React, { createContext, useState, useEffect } from "react";

export const CartContext = createContext();

export const CartProvider = ({ children }) => {
  const [cartItems, setCartItems] = useState([]);
```

```
useEffect(() => {
  const saved = localStorage.getItem("cartItems");
  if (saved) setCartItems(JSON.parse(saved));
}, []);

useEffect(() => {
  localStorage.setItem("cartItems", JSON.stringify(cartItems));
}, [cartItems]);

const addToCart = (course) => {
  setCartItems((prev) => {
    const existing = prev.find((item) => item.id === course.id);
    return existing
      ? prev.map((item) =>
          item.id === course.id
            ? { ...item, qty: item.qty + 1 }
            : item
        )
      : [...prev, { ...course, qty: 1 }];
  });
};

const updateQuantity = (id, qty) =>
  setCartItems((prev) =>
    prev.map((item) =>
      item.id === id ? { ...item, qty: parseInt(qty) } : item
```

```
    )
  );

  const removeFromCart = (id) =>
    setCartItems((prev) => prev.filter((item) => item.id !== id));

  const clearCart = () => setCartItems([]);

  return (
    <CartContext.Provider
      value={{ cartItems, addToCart, updateQuantity, removeFromCart, clearCart }}
    >
      {children}
    </CartContext.Provider>
  );
};
import { Link, useNavigate } from "react-router-dom";
import { useState, useEffect } from "react";

function Navbar() {
  const [isLoggedIn, setIsLoggedIn] = useState(localStorage.getItem("isLoggedIn") === "true");
  const navigate = useNavigate();

  useEffect(() => {
    const check = () => setIsLoggedIn(localStorage.getItem("isLoggedIn") === "true");
    window.addEventListener("storage", check);
    return () => window.removeEventListener("storage", check);
```

```
}, []);


const handleLogout = () => {
  localStorage.removeItem("isLoggedIn");
  alert("You have been logged out.");
  navigate("/");
  setIsLoggedIn(false);
};


return (
  <nav className="navbar navbar-expand-lg navbar-dark bg-dark fixed-top shadow-sm">
    <div className="container-fluid">
      <Link className="navbar-brand fw-bold text-info" to="/">DSA Learning</Link>
      <div className="collapse navbar-collapse">
        <ul className="navbar-nav ms-auto">
          <li><Link className="nav-link" to="/">Home</Link></li>
          <li><Link className="nav-link" to="/courses">Courses</Link></li>
          <li><Link className="nav-link" to="/cart">Cart</Link></li>
          <li><Link className="nav-link" to="/contact">Contact</Link></li>
          {isLoggedIn ? (
            <>
              <li><Link className="nav-link" to="/profile">Profile</Link></li>
              <li><button onClick={handleLogout} className="btn btn-danger btn-sm ms-2">Logout</button></li>
            </>
          ) : (
            <>
              <li><Link className="nav-link" to="/register">Sign Up</Link></li>
```

```
            <li><Link className="nav-link" to="/login">Login</Link></li>
          </>
        )}
      </ul>
    </div>
  </div>
 </nav>
 );
}

export default Navbar;
import React, { useState } from "react";
import CourseCard from "../components/CourseCard";

function Home() {
 const [search, setSearch] = useState("");
 const courses = [
  { id: 1, name: "DSA Fundamentals", price: 499, image: "1.jpg" },
  { id: 2, name: "Algorithms Mastery", price: 799, image: "1.jpg" },
  { id: 3, name: "Competitive Programming", price: 999, image: "1.jpg" },
 ];

 const filtered = courses.filter(c =>
  c.name.toLowerCase().includes(search.toLowerCase())
 );

 return (
```

```
    <div className="container mt-5 pt-5">
      <h2 className="text-center text-primary fw-bold mb-4">Welcome to DSA Learning</h2>
      <input
        type="text"
        className="form-control w-50 mx-auto mb-4"
        placeholder="Search courses..."
        value={search}
        onChange={(e) => setSearch(e.target.value)}
      />
      <div className="row">
        {filtered.map((course) => (
          <div className="col-md-4 mb-4" key={course.id}>
            <CourseCard course={course} />
          </div>
        ))}
      </div>
    </div>
  );
}
export default Home;
import React, { useContext } from "react";
import { CartContext } from "../context/CartContext";

function CourseCard({ course }) {
  const { addToCart } = useContext(CartContext);

  return (
```

```
    <div className="card shadow-sm border-0 rounded">
      <img
        src={course.image}
        alt={course.name}
        className="card-img-top rounded-top"
        style={{ height: "180px", objectFit: "cover" }}
      />
      <div className="card-body text-center">
        <h5 className="fw-bold">{course.name}</h5>
        <p className="text-success fw-semibold">₹{course.price}</p>
        <button
          className="btn btn-primary btn-sm w-100"
          onClick={() => addToCart(course)}
        >
          Add to Cart
        </button>
      </div>
    </div>
  );
}
export default CourseCard;
import React, { useState } from "react";
import CourseCard from "../components/CourseCard";

function Courses() {
  const [search, setSearch] = useState("");
  const courses = [
```

```
  { id: 1, name: "DSA Basics", desc: "Learn arrays, stacks, queues", price: 499, image: "1.jpg" },
  { id: 2, name: "Graph Algorithms", desc: "DFS, BFS, Dijkstra", price: 699, image: "1.jpg" },
  { id: 3, name: "Dynamic Programming", desc: "Memoization & Tabulation", price: 899, image: "1.jpg" },
  { id: 4, name: "Sorting Mastery", desc: "Quick, Merge, Heap Sort", price: 399, image: "1.jpg" },
];

const filtered = courses.filter(
  (c) =>
    c.name.toLowerCase().includes(search.toLowerCase()) ||
    c.desc.toLowerCase().includes(search.toLowerCase())
);

return (
  <div className="container mt-5 pt-5">
    <h2 className="text-center fw-bold mb-4 text-primary">All DSA Courses</h2>
    <input
      type="text"
      className="form-control w-50 mx-auto mb-4"
      placeholder="Search courses..."
      value={search}
      onChange={(e) => setSearch(e.target.value)}
    />
    <div className="row">
      {filtered.map((course) => (
        <div className="col-md-3 mb-4" key={course.id}>
          <CourseCard course={course} />
        </div>
```

```
        ))}
      </div>
    </div>
  );
}
export default Courses;
import React, { useContext } from "react";
import { CartContext } from "../context/CartContext";
import { useNavigate } from "react-router-dom";

function Cart() {
  const { cartItems, updateQuantity, removeFromCart } = useContext(CartContext);
  const navigate = useNavigate();

  const total = cartItems.reduce(
    (sum, item) => sum + item.price * item.qty,
    0
  );

  const handleCheckout = () => {
    const loggedIn = localStorage.getItem("isLoggedIn");
    if (!loggedIn) {
      alert("Please log in before checkout!");
      navigate("/login");
      return;
    }
    navigate("/checkout", { state: { cartItems, total } });
```

```
};

return (
 <div className="container mt-5 pt-5">
  <h2 className="text-center text-primary fw-bold mb-4">Your Cart</h2>

  {cartItems.length === 0 ? (
   <p className="text-center text-muted">Your cart is empty.</p>
  ) : (
   <>
    <table className="table table-striped align-middle">
     <thead className="table-dark">
      <tr>
       <th>Course</th>
       <th>Price (₹)</th>
       <th>Quantity</th>
       <th>Total (₹)</th>
       <th>Remove</th>
      </tr>
     </thead>
     <tbody>
      {cartItems.map((item) => (
       <tr key={item.id}>
        <td>{item.name}</td>
        <td>₹{item.price}</td>
        <td>
         <input
```

```
          type="number"
          min="1"
          value={item.qty}
          onChange={(e) =>
            updateQuantity(item.id, e.target.value)
          }
          className="form-control w-50"
        />
      </td>
      <td>₹{item.price * item.qty}</td>
      <td>
        <button
          className="btn btn-danger btn-sm"
          onClick={() => removeFromCart(item.id)}
        >
          Delete
        </button>
      </td>
    </tr>
  ))}
 </tbody>
</table>
<h4 className="text-end mt-3">
 Total Amount: <span className="text-success">₹{total}</span>
</h4>
<div className="text-end">
 <button className="btn btn-success px-4 fw-bold" onClick={handleCheckout}>
```

```
            Proceed to Checkout
          </button>
        </div>
      </>
    )}
  </div>
 );
}
export default Cart;
import React, { useState, useContext } from "react";
import { useLocation, useNavigate } from "react-router-dom";
import { CartContext } from "../context/CartContext";

function Checkout() {
 const location = useLocation();
 const navigate = useNavigate();
 const { clearCart } = useContext(CartContext);
 const [formData, setFormData] = useState({
  name: "",
  address: "",
  city: "",
  pincode: "",
  phone: "",
 });

 const handleChange = (e) =>
  setFormData({ ...formData, [e.target.name]: e.target.value });
```

```
const handlePlaceOrder = () => {
  if (Object.values(formData).some((v) => v.trim() === "")) {
    alert("Please fill all fields before placing the order.");
    return;
  }
  alert("☑ Order placed successfully! Thank you for learning with DSA Learning.");
  clearCart();
  navigate("/");
};

const { cartItems, total } = location.state || { cartItems: [], total: 0 };

return (
  <div className="container mt-5 pt-5">
    <h2 className="text-center text-primary fw-bold mb-4">Checkout</h2>
    <div className="row">
      <div className="col-md-6">
        <h4>Shipping Details</h4>
        <input name="name" placeholder="Full Name" className="form-control mb-2" onChange={handleChange} />
        <textarea name="address" placeholder="Address" className="form-control mb-2" onChange={handleChange} />
        <input name="city" placeholder="City" className="form-control mb-2" onChange={handleChange} />
        <input name="pincode" placeholder="Pincode" className="form-control mb-2" onChange={handleChange} />
        <input name="phone" placeholder="Phone" className="form-control mb-2" onChange={handleChange} />
        <button className="btn btn-success w-100 mt-3" onClick={handlePlaceOrder}>
          Place Order
        </button>
```

```
      </div>

      <div className="col-md-6">
        <h4>Order Summary</h4>
        {cartItems.map((item) => (
          <p key={item.id}>
            {item.name} × {item.qty} = ₹{item.price * item.qty}
          </p>
        ))}
        <h5>Total: ₹{total}</h5>
      </div>
    </div>
  </div>
 );
}
export default Checkout;
import React, { useState } from "react";
import { useNavigate } from "react-router-dom";

function Register() {
 const [user, setUser] = useState({ name: "", email: "", password: "" });
 const navigate = useNavigate();

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

 const handleSignup = (e) => {
```

```
    e.preventDefault();
    if (!user.name || !user.email || !user.password) {
      alert("All fields are required!");
      return;
    }
    localStorage.setItem("userData", JSON.stringify(user));
    alert("Registration successful!");
    navigate("/login");
  };

  return (
    <div className="container mt-5 pt-5">
      <h2 className="text-center fw-bold text-primary mb-4">User Registration</h2>
      <form className="w-50 mx-auto shadow p-4 rounded" onSubmit={handleSignup}>
        <input name="name" placeholder="Full Name" className="form-control mb-3" onChange={handleChange} />
        <input name="email" type="email" placeholder="Email" className="form-control mb-3"
onChange={handleChange} />
        <input name="password" type="password" placeholder="Password" className="form-control mb-3"
onChange={handleChange} />
        <button className="btn btn-primary w-100">Sign Up</button>
      </form>
    </div>
  );
}
export default Register;
import React, { useState } from "react";
import { useNavigate } from "react-router-dom";
```

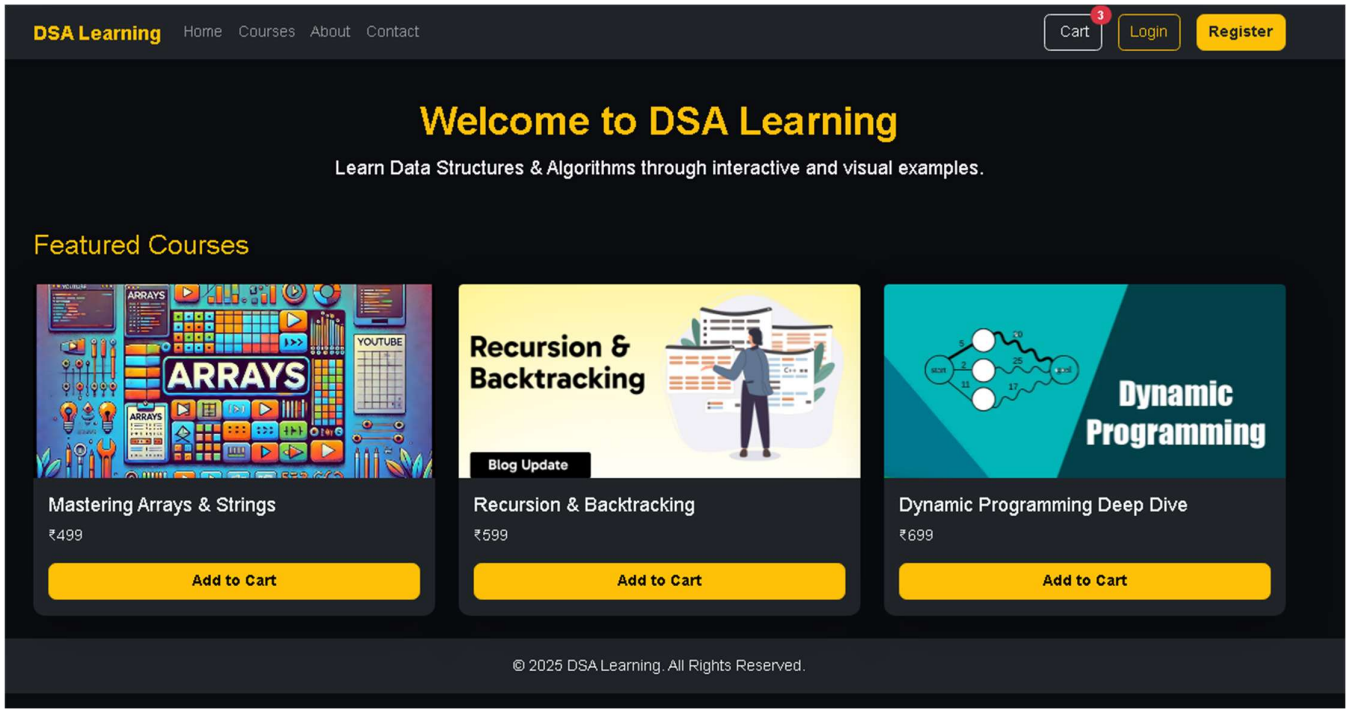
```
function Login() {
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");
  const navigate = useNavigate();

  const handleLogin = (e) => {
    e.preventDefault();
    const stored = JSON.parse(localStorage.getItem("userData"));
    if (!stored) {
      alert("No user found. Please register first!");
      navigate("/register");
      return;
    }
    if (stored.email === email && stored.password === password) {
      localStorage.setItem("isLoggedIn", "true");
      alert("Login successful!");
      navigate("/profile");
    } else {
      alert("Invalid credentials!");
    }
  };

  return (
    <div className="container mt-5 pt-5">
      <h2 className="text-center fw-bold text-success mb-4">Login</h2>
      <form className="w-50 mx-auto shadow p-4 rounded" onSubmit={handleLogin}>
        <input
```

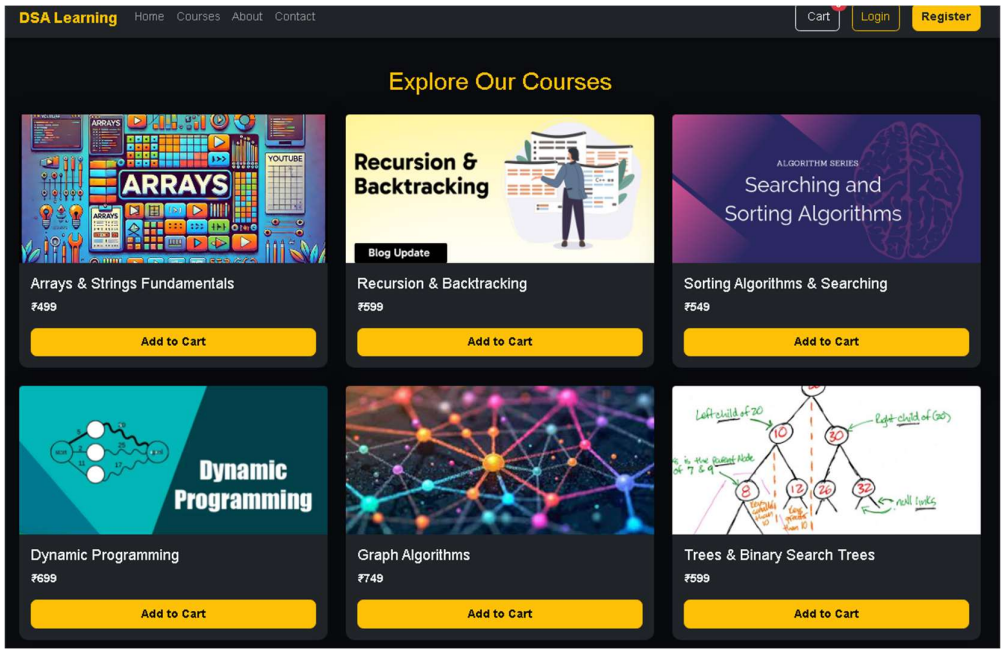
```
          type="email"
          placeholder="Email"
          className="form-control mb-3"
          onChange={(e) => setEmail(e.target.value)}
        />
        <input
          type="password"
          placeholder="Password"
          className="form-control mb-3"
          onChange={(e) => setPassword(e.target.value)}
        />
        <button className="btn btn-success w-100">Login</button>
      </form>
    </div>
  );
}
export default Login;
import React from "react";
import { useNavigate } from "react-router-dom";

function Profile() {
  const user = JSON.parse(localStorage.getItem("userData"));
  const navigate = useNavigate();

  const handleLogout = () => {
    localStorage.removeItem("isLoggedIn");
    alert("Logged out successfully!");
```

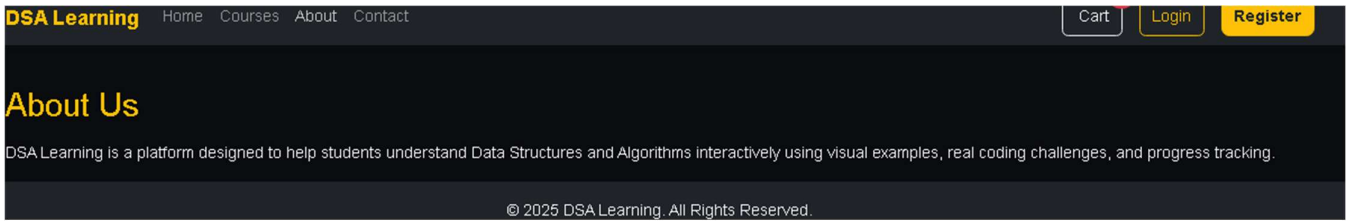
```
    navigate("/");
  };

  if (!user) {
    return <p className="text-center mt-5 pt-5">No user data found.</p>;
  }

  return (
    <div className="container mt-5 pt-5">
      <h2 className="text-center fw-bold text-primary mb-4">My Profile</h2>
      <div className="card shadow p-4">
        <p><strong>Name:</strong> {user.name}</p>
        <p><strong>Email:</strong> {user.email}</p>
        <button className="btn btn-danger w-25" onClick={handleLogout}>Logout</button>
      </div>
    </div>
  );
}
export default Profile;
import React from "react";

function Contact() {
  return (
    <div className="container mt-5 pt-5 text-center">
      <h2 className="text-primary fw-bold mb-3">Contact Us</h2>
      <p>Email: support@dsalearning.com</p>
      <p>Phone: +91 98765 43210</p>
```

```
    </div>
  );
}
export default Contact;
```

Output :

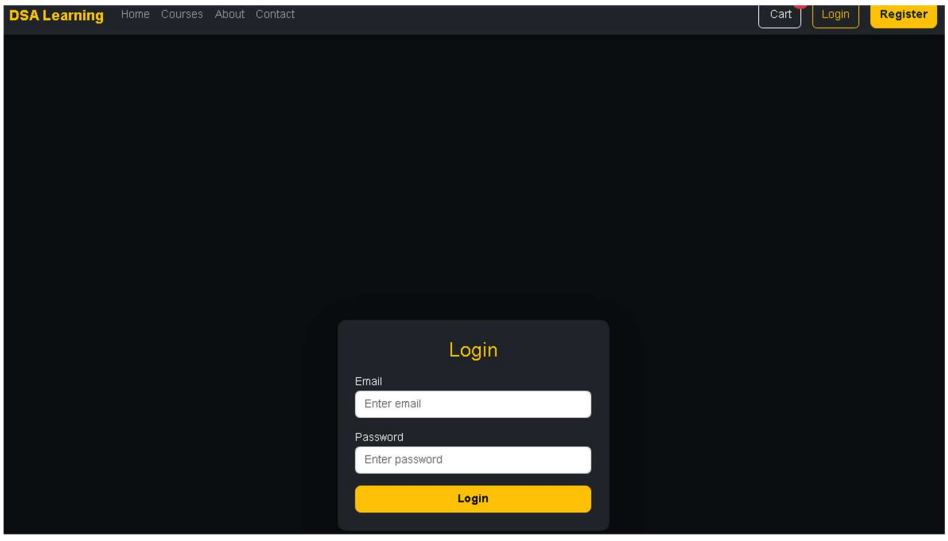


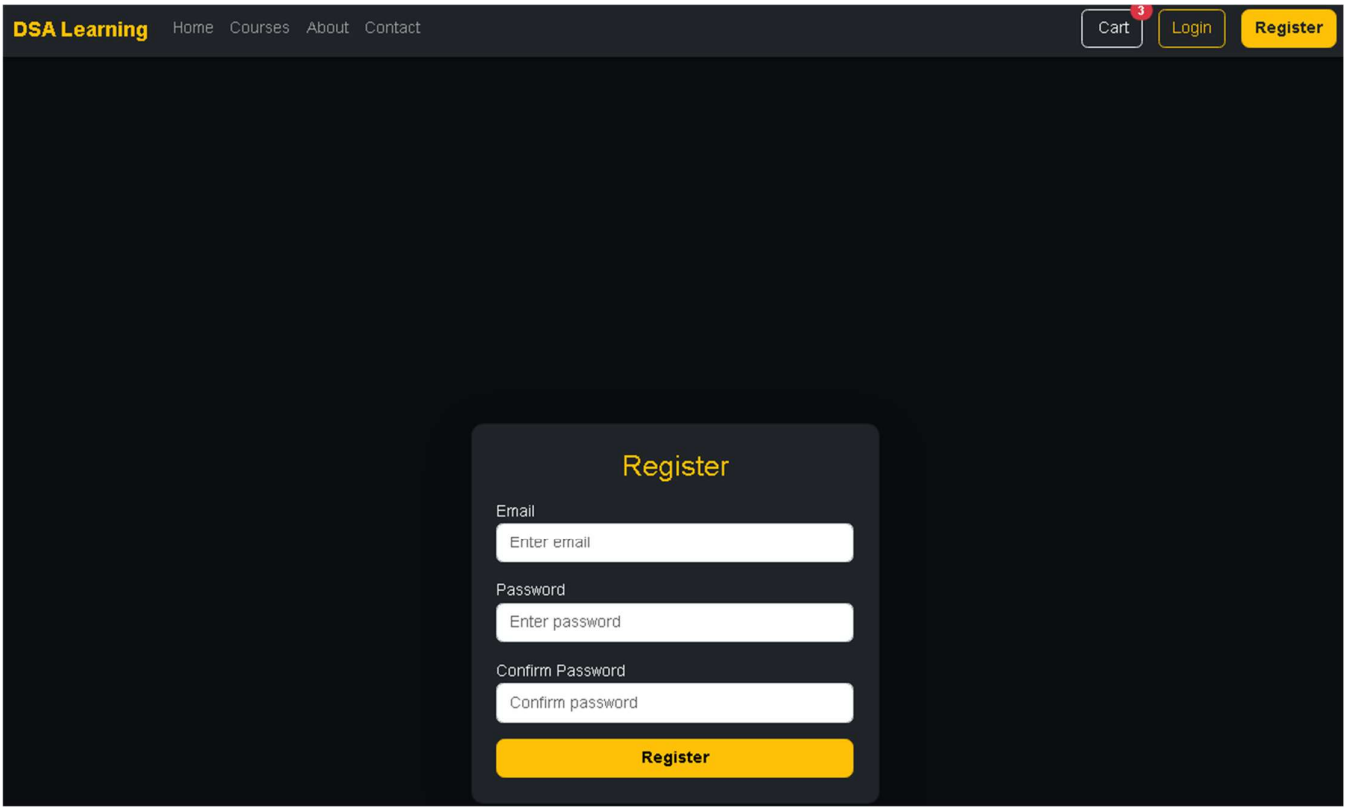



DSA Learning
Home
Courses
About
Contact
Cart
Login
Register
About Us
DSA Learning is a platform designed to help students understand Data Structures and Algorithms interactively using visual examples, real coding challenges, and progress tracking.
© 2025 DSA Learning. All Rights Reserved.


DSA Learning
Home
Courses
About
Contact
Cart
Login
Register
Contact Us
Name
Email
Message
Send
© 2025 DSA Learning. All Rights Reserved.


DSA Learning
Home
Courses
About
Contact
Cart
3
Login
Register
Your Cart
Mastering Arrays & Strings
₹499
Remove
Recursion & Backtracking
₹599
Remove
Dynamic Programming Deep Dive
₹699
Remove
© 2025 DSA Learning. All Rights Reserved.


DSA Learning
Home
Courses
About
Contact
Cart
Login
Register
Login
Email
Enter email
Password
Enter password
Login



Conclusion :

We successfully developed a **complete React single-page application (SPA)** for the DSA Learning portal. It demonstrates:

- Modular React structure
- React Router DOM navigation
- Context API state management
- Dynamic cart functionality
- LocalStorage-based authentication
- Form handling & validation

This assignment combines all the key concepts of modern React development into one cohesive project.