

**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**

**Shrinivas Sherikar – ADT23SOCB1610**

**Under the Guidance of Prof. Rohini Bhosale**

**T.Y. 2025-2026 (SEM – V)**

**COURSE STRUCTURE**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Full Stack Front-End Development**  **SEMESTER – V** | | | | | |
| **Course Code:** | | **23IT3001** | **Course Credits:** | **04** | |
| **Teaching Hours / Week (L:T:P):** | | **2:0:4** | **CA Marks:** | **50** | |
| **Total Number of Teaching Hours:** | | **60** | **END-SEM Marks:** | **50** | |
| **Course Prerequisites:** Web Programming | | | | | |
| **Course Description:**  This course equips students with skills in responsive web design using Bootstrap and Flexbox, and dynamic interface development using React. It emphasizes component-based architecture, state management, and interactive user experiences for modern web applications. | | | | | |
| **Course Learning Objectives:** This course will enable the students to:   1. Demonstrate the fundamental concepts of responsive web design with grid structures and Bootstrap components. 2. Design and build clear, easy-to-use web forms using Bootstrap and Flexbox to improve user experience and accessibility. 3. Set up and use a React development environment, showing skill in basic project setup, JSX, and building modular apps. 4. Develop proficiency in constructing modular and reusable React components using JSX and props. 5. Develop the capability to execute interactivity and state management in React applications through the utilization of Hooks and event handling. | | | | | |
| **Course Outcome:** The students will be able to:   1. Illustrate responsive design principles using Bootstrap components and grid systems for adaptive web interfaces. 2. Design accessible, structured web forms using Bootstrap utilities and Flexbox for intuitive user experiences. 3. Configure and deploy a React development environment to scaffold and run modular front-end applications. 4. Analyze and construct reusable React components by leveraging JSX and data binding through props. 5. Create interactive React applications using Hooks and event-handling techniques for dynamic state management. | | | | | |
| **UNIT – I** | **Responsive Interfaces with Bootstrap Components** | | | | **9 Hours** |
| Module 1: Introduction to Media Queries, Responsive Web Design Using Media Queries, Syntax, Uses of Media Queries, Benefits of Responsive Design, Elements of a Responsive Design, Working of Responsive Design  Module 2: Introduction to Bootstrap: Introduction to Bootstrap: Bootstrap Concept, Uses of Bootstrap, Bootstrap library source, Bootstrap Basics- container, jumbotron, page header, buttons, tables;  Module 3: Bootstrap Grid System | | | | | |
| **Pedagogy** | **ICT Teaching / PowerPoint Presentation and Videos** | | | | |
| **Self-study / Do it yourself /** | | | | |
| **Experiential Learning Topics:** | | | | |
| **Case Study / PBL - Project Based Learning** | | | | |
| **UNIT – II** | **Form Design and Layouts using Bootstrap and Flexbox** | | | | **9 Hours** |
| Module 1: Bootstrap Forms- stacked form, textarea, form row/grid (inline forms), form control size, color picker, select menu, select menu size, data list, checkboxes, radio buttons, toggle switches;    Module 2: Bootstrap Flexbox- horizontal direction, vertical direction. | | | | | |
| **Pedagogy** | **ICT Teaching / PowerPoint Presentation and Videos** | | | | |
| **Self-study / Do it yourself** | | | | |
| **Experiential Learning Topics:** | | | | |
| **Case Study / PBL - Project Based Learning** | | | | |
| **UNIT – III** | **Introduction to React and Development Environment Setup** | | | | **9 Hours** |
| Module 1 - Introduction to React: What is React? Advantages of React for dynamic web and mobile applications;  Module 2 - Environment Setup: Installing Node.js and npm, Creating a React project using create-react-app; Overview of folder structure and development tools; Running the development server and understanding hot reloading;  Module 3- Writing Your First React Application: Building a simple "Hello, World!" React app; Modifying App.js to understand component structure; | | | | | |
| **Pedagogy** | **ICT Teaching / PowerPoint Presentation and Videos** | | | | |
| **Self-study / Do it yourself /** | | | | |
| **Experiential Learning Topics:** | | | | |
| **Case Study / PBL - Project Based Learning** | | | | |
| **UNIT – IV** | **Core React Concepts: JSX and Components** | | | | **9 Hours** |
| Module 1 - JSX Basics: Writing JSX syntax. Embedding expressions in JSX; Rules of JSX (e.g., className vs class)  Module 2 - Components and Props: Understanding Functional and Class Components; Creating reusable components; Passing data using props | | | | | |
| **Pedagogy** | **ICT Teaching / PowerPoint Presentation and Videos** | | | | |
| **Self-study / Do it yourself** | | | | |
| **Experiential Learning Topics:** | | | | |
| **Case Study / PBL - Project Based Learning** | | | | |
| **UNIT – V** | **State Management and Interactivity in React** | | | | **9 Hours** |
| Module 1: Component State Management - Introduction to useState hook for managing internal state, Updating state and understanding re-renders, Conditional rendering using state, Managing form data using component state  Module 2: Event Handling and User Interactions, Handling user events such as clicks, form submissions, Binding event handlers in functional and class components, Preventing default form behavior, Best practices for clean and maintainable event logic | | | | | |
| **Pedagogy** | **ICT Teaching / PowerPoint Presentation and Videos** | | | | |
| **Self-study / Do it yourself** | | | | |
| **Experiential Learning Topics:** | | | | |
| **Case Study / PBL - Project Based Learning** | | | | |

**Text Books:**

1. Snig Bhaumik, |Bootstrap Essentials|, Packt Publishing Limited, ISBN-10: ‎178439517X, ISBN-13: ‎ 978-1784395179
2. Alex Banks, |Learning React|, O'Reilly Media, ISBN-10 : ‎ 1492051721, ISBN-13 : ‎ 978-1492051725
3. Stoyan Stefanov, |React - Up & Running|, O'Reilly Media, Inc, USA, ISBN-10 : ‎ 1491931825, ISBN-13 : ‎ 978-1491931820

**Reference Books:**

1. Frahaan Hussain, |Mastering Bootstrap 5: From Basics to Expert Projects|, ISBN-13: ‎ 979-8871360415
2. Robin Wieruch, |The Road to React|, Zaccheus Entertainment, ISBN-10 : ‎ 9781720043997, ISBN-13 : ‎ 978-1720043997

**URLs (Optional) - List of Online Courses**

1. **Full-Stack Web Development with React Specialization – Coursera (by The Hong Kong University of Science and Technology)** Covers: Node.js, Express.js, MongoDB, React, REST APIs  
    Platform: Coursera  
    Level: Intermediate – Project-based
2. **The Complete Node.js Developer Course (3rd Edition) – Udemy (by Andrew Mead & Rob Percival)** Covers: Node.js, Express, MongoDB, REST APIs, Authentication  
    Platform: Udemy  
    Level: Beginner to Advanced
3. **Web Development with Node.js and Express – Coursera (by University of London)** Covers: Node.js, Express.js, REST APIs  
    Platform: Coursera  
    Level: Intermediate
4. **MySQL for Data Analytics and Business Intelligence – Udemy** Covers: MySQL Basics to Advanced (joins, CRUD, queries)  
    Platform: Udemy  
    Level: Beginner to Intermediate
5. **Full Stack Open 2024 – University of Helsinki (Free)** Covers: Modern web app development with React, Node.js, REST APIs, MongoDB  
    Platform: Independent (Free)  
    Level: Advanced

**List of Assignments:**

|  |  |
| --- | --- |
| **Laboratory Assignments: Project based Assignment** | |
| **Project-I** | Project Title: ReactJS E-Commerce Website with Page RoutingProject Statement: Objective: Develop a fully functional E-commerce website using ReactJS that includes page routing functionality for smooth navigation between different sections like Home, Product Listing, Product Details, Cart, and Checkout. Project Features:  1. Navigation Bar with Routing:    * Create a Navigation Bar at the top of the website with links to:      + Home: Displays featured products and general information about the store.      + Products: Lists all available products for purchase.      + Cart: Displays the products added to the shopping cart.      + Checkout: Allows the user to review and complete the purchase.      + Profile: Displays user account information and order history.    * Implement React Router for routing between these pages. 2. Home Page:    * The Home page should showcase highlighted products or categories.    * Implement a product carousel or featured section.    * Include a search bar to allow users to search for products directly. 3. Product Listing Page:    * Create a Product Listing page that fetches products dynamically from an API or static JSON file.    * Each product in the list should show:      + Product image.      + Product name.      + Price.      + A button to add the product to the cart.    * Implement React Router links on product items to navigate to the Product Details page. 4. Product Details Page:    * When a user clicks on a product, navigate them to the Product Details page.    * The Product Details page should display:      + Full-size product image.      + Detailed product description.      + Price.      + Add to Cart button.    * Allow users to select product quantity before adding it to the cart. 5. Cart Page:    * Implement a Cart page that displays products added to the cart.    * Each cart item should include:      + Product name.      + Product quantity.      + Price per item and total cost for the item.      + A button to remove items from the cart.    * Display a total price for all items in the cart.    * Include buttons for Continue Shopping and Proceed to Checkout. 6. Checkout Page:    * Create a Checkout page where users can review their cart, enter shipping information, and confirm the order.    * Display an order summary with total cost, shipping details, and payment options.    * Provide a Place Order button to finalize the transaction. 7. User Profile Page:    * Create a Profile page where users can view their account information and order history.    * Include a form to update the profile details (name, email, shipping address). 8. State Management with Context API/Redux:    * Use React Context API or Redux to manage the global state of the cart, allowing users to add, remove, and update quantities of products across different pages.    * Ensure that the cart data is preserved during navigation between pages. 9. Responsive Design:    * Ensure the website is responsive and works well on both desktop and mobile devices using CSS Flexbox or CSS Grid.  Technical Requirements:ReactJS: For building the user interface.React Router: For implementing page routing functionality.State Management: Use React Context API or Redux to manage cart state and other global data.API Integration: Fetch products from a static JSON file or a mock API (e.g., JSONPlaceholder or a custom API).Responsive Styling: Use CSS/SCSS to ensure the website is mobile-friendly.Optional: Add localStorage or sessionStorage to persist cart data between page reloads.Expected Outcome: By completing this project, you will:   * Learn how to implement page routing in a ReactJS application using React Router. * Build a dynamic E-commerce website with features like product listings, cart management, and checkout functionality. * Gain experience with state management and ensuring the cart data persists across different pages. * Develop a responsive, user-friendly interface that functions well on both mobile and desktop screens.   Assignment list:   1. React Project Setup and Navigation  * Create a new React project using create-react-app. * Set up routing using react-router-dom. * Create a top navigation bar with links to: Home, Products, Cart, Checkout, and Profile.  1. Home Page Development  * Design the Home page layout. * Include featured products or categories. * Add a functional search bar to filter products by name or category.  1. Fetch product data from array or static JSON file  * Display product image, name, and price. * Include an Add to Cart button. * Use React Router links for navigation to individual product detail pages.  1. Product Details Page  * On clicking a product, route to the product detail page. * Show full-size image, detailed description, price, quantity selector, and Add to Cart functionality.  1. Cart Functionality  * Create a Cart page to display products added to the cart. * Allow users to:  1. Change quantities 2. Remove products 3. View total cost  * Provide "Continue Shopping" and "Proceed to Checkout" buttons.  1. Checkout Page  * Build a Checkout page with:  1. Form to capture user shipping details 2. Order summary section 3. Place Order button to confirm purchase 4. User Profile Page  * Create a Profile page to:  1. Display user account information 2. Show order history (dummy or stored locally) 3. Include a form to update profile details like name, email, and address 4. State Management Using Context API / Redux  * Implement cart state using React Context API or Redux. * Enable global add, remove, and update actions for cart items. * Maintain state across all page navigations.  1. Routing and Component Structure  * Use React Router to manage navigation between pages. * Break the UI into reusable components (e.g., Navbar, ProductCard, CartItem). * Ensure smooth navigation and modular code structure.  1. Final Integration  * Integrate all previous assignments into one working React E-Commerce App. * Ensure: a. Routing works properly b. State is maintained c. Pages are connected and functional * Create a GitHub repo and Submit a complete codebase   Additional Assignments:  Assignment 1: Create a Static Homepage Layout Using Bootstrap   * Design a responsive homepage using Bootstrap components like container, jumbotron, headers, and buttons. * Use the grid system to create a structured and adaptive layout suitable for any web app.   Assignment 2: Design a Grid-Based Information Page   * Develop a multi-column content layout using Bootstrap Grid. * Display items such as cards, product previews, or blog entries with consistent alignment and spacing.   Assignment 3: Bootstrap Form and Table Integration   * Redesign the Checkout and Profile forms using Bootstrap forms:  1. Use components like form-group, form-control, form-row, and validation classes.  * Display the Cart and Order History using Bootstrap tables:  1. Include features like striped rows, borders, and responsive layout.   Assignment 4: Responsive Design Using Bootstrap Grid and Utilities   * Apply Bootstrap Grid System to all pages for layout consistency. * Use Bootstrap utilities like:  1. d-flex, justify-content-\*, align-items-\* for alignment 2. m-, p-, text- classes for spacing and text formatting 3. Test and optimize responsiveness across mobile, tablet, and desktop views. |

|  |  |  |
| --- | --- | --- |
| ASSESSMENT AND EVALUATION PATTERN | | |
| WEIGHTAGE | Continuous Assessment (CA) | End Semester Assessment (ESA) |
| CA - Assignments (PBL / Case Study / Presentation / Poster Presentation / Seminar / Group Discussions / Quiz / Test / Open Book Examination / Book or Article Review / Jury / Concept Map / Mind Maps | | |
| ESA - Bloom’s Taxonomy Levels: Remembering, Understanding, Applying, Analyzing, Evaluating, and Creating | | |
| Experiential Learning - Internship (Summer / Winter) / Industry Visit / Site Visit / Field Visit | | |
| Project Based Learning – Mini (Minor) Project / Major Project / Website Making / e-Portfolios | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Course Articulation Matrix (CO-PO Mapping) [Full Stack Front End Development - 23IT3001]** | | | | | | | | | | | | | | | |
| COs | PO-1 | PO-2 | PO-3 | PO-4 | PO-5 | PO-6 | PO-7 | PO-8 | PO-9 | PO-10 | PO-11 | PO-12 | PSO-1 | PSO-2 | Relevance |
| CO-1 | 3.0 | 3.0 | 2.0 | 3.0 | 3.0 |  |  |  |  | 3.0 |  | 3.0 | 2.0 |  | National/Global |
| CO-2 | 3.0 | 3.0 | 2.0 | 3.0 | 3.0 |  |  |  |  | 2.0 |  | 3.0 | 2.0 |  | National/Global |
| CO-3 | 3.0 | 2.0 |  |  | 2 |  |  |  |  | 3.0 |  | 3.0 | 2.0 |  | National/Global |
| CO-4 | 3.0 | 2.0 |  |  | 2 |  |  |  |  | 3.0 |  | 3.0 | 2.0 |  | National/Global |
| CO-5 | 3.0 | 2.0 |  |  | 2 |  |  |  |  | 3.0 |  | 3.0 | 2.0 |  | National/Global |
| **3** – HIGH, **2** – MEDIUM, **1** - LOW | | | | | | | | | | | | | | | |

## **Experiment 1: Navigation Bar with Routing**

### **Title:**

Navigation Bar with Routing in *DrawKit – Collaborative Whiteboarding Tool*

### **Objective:**

To design and implement a responsive navigation bar using React Router that enables seamless navigation between different sections of the DrawKit collaborative whiteboarding tool.

### **Theory:**

A **Navigation Bar (Navbar)** is a fundamental component of modern web applications, allowing users to move between different sections easily. In a single-page application (SPA) like **DrawKit**, routing is handled on the client side using **React Router**, ensuring smooth transitions without reloading the page.

**React Router** is a powerful library used to manage navigation in React applications. It helps define routes that map URL paths to specific components. When users click a navigation link, React Router updates the view dynamically, maintaining the SPA structure.

In the context of **DrawKit**, a collaborative whiteboarding tool, the navigation bar serves as the primary interface for accessing different modules such as brainstorming boards, wireframe tools, flowchart creators, and user profiles. A well-structured navbar improves usability, enhances productivity, and provides quick access to essential features.

The **Navbar** typically contains the following components:

1. **Home:** Provides an overview of the platform, featured tools, and updates.
2. **Whiteboards / Products:** Lists available collaborative boards or drawing tools.
3. **Team / Cart:** Displays active collaborations or shared whiteboards.
4. **Workspace / Checkout:** Allows users to start a new project or manage existing boards.
5. **Profile:** Displays user information, saved work, and settings.

**Routing Concepts Used:**

* **BrowserRouter:** Wraps the entire app to enable routing.
* **Routes & Route:** Define each navigable path.
* **Link / NavLink:** Creates clickable navigation links.
* **useNavigate Hook:** Enables programmatic navigation between routes.

**Advantages of Implementing Routing in DrawKit:**

* Enhances user experience by avoiding page reloads.
* Keeps the interface organized and consistent.
* Supports modular structure for easy scaling.
* Allows for dynamic rendering of collaborative workspaces.

### **Tools and Technologies Used:**

* **Frontend:** ReactJS
* **Routing Library:** React Router DOM
* **Styling:** CSS / Tailwind CSS
* **IDE:** Visual Studio Code
* **Browser:** Google Chrome

### **Procedure / Tasks Performed:**

1. Initialize a React project using npx create-react-app drawkit.

Install React Router using the command:  
  
 npm install react-router-dom

1. Create the following components:  
   * Home.js
   * Products.js
   * Cart.js
   * Checkout.js
   * Profile.js
2. Create a Navbar.js component and add navigation links using <NavLink>.
3. Wrap the app in <BrowserRouter> inside App.js.
4. Define routes inside <Routes> for each component.
5. Apply CSS or Tailwind styling to design a responsive top navigation bar.
6. Run the application using npm start and verify smooth navigation between pages.

### **Step 4: Setting Up Routing in App.js**

import Header from './components/Header'

import Footer from './components/Footer'

import { BrowserRouter, Route, Routes } from 'react-router-dom'

import Home from './pages/Home'

import About from './pages/About'

import Login from './pages/Login'

import Signup from './pages/Signup'

import Contact from './pages/Contact'

import Cart from './pages/Cart'

import ProtectedRoute from './components/ProtectedRoute'

import PublicOnlyRoute from './components/PublicOnlyRoute'

import Board from './pages/Board'

export default function App() {

  return (

    <BrowserRouter>

      <div className="flex flex-col min-h-screen">

         <div

        className="absolute inset-x-0 top-0 z-0 pointer-events-none"

        style={{

          height: '100vh',

          background: "radial-gradient(ellipse 80% 60% at 50% 0%, rgba(168, 165, 255, 0.20), transparent 70%)",

        }}

      />

        <Header />

        <main className='flex-grow'>

          <Routes>

            <Route path='/' element={<Home />} />

            <Route path='/about' element={<About />} />

            <Route path='/contact' element={<Contact />} />

            <Route path='/cart' element={<Cart />} />

            <Route

              path='/login'

              element={

                <PublicOnlyRoute>

                  <Login />

                </PublicOnlyRoute>

              }

            />

            <Route

              path='/register'

              element={

                <PublicOnlyRoute>

                  <Signup />

                </PublicOnlyRoute>

              }

            />

            <Route

              path='/board'

              element={

                <ProtectedRoute>

                  <Board />

                </ProtectedRoute>

              }

            />

          </Routes>

        </main>

        <Footer />

      </div>

    </BrowserRouter>

  )

}

### **Step 5: Creating the Navigation Bar**

import React from "react";

import { Link, useNavigate } from "react-router-dom";

import { useAuth } from "../hooks/useAuth"; // Import the auth hook

function Header() {

  const [isMenuOpen, setIsMenuOpen] = React.useState(false);

  const { isLoggedIn, logout } = useAuth();

  const navigate = useNavigate();

  const handleLogout = () => {

    logout();

    navigate('/');

  };

  return (

    <nav className="bg-dark-custom fixed top-0 w-full z-50  bg-transparent">

      <div className="container mx-auto px-4">

        <div className="flex justify-between items-center h-16">

          <Link className="text-2xl font-bold text-white" to="/">DrawKit</Link>

          <div className="hidden md:flex items-center space-x-6">

            <Link to="/" className="text-gray-300 hover:text-white transition-colors">Home</Link>

            <Link to="/about" className="text-gray-300 hover:text-white transition-colors">About Us</Link>

            <Link to="/contact" className="text-gray-300 hover:text-white transition-colors">Contact</Link>

            {isLoggedIn && (

              <Link to="/board" className="text-gray-300 hover:text-white transition-colors">My Board</Link>

            )}

            <div className="pl-4">

              {isLoggedIn ? (

                <button

                  onClick={handleLogout}

                  className="py-2 px-4 bg-primary-custom text-black rounded hover:bg-opacity-90 transition-colors text-sm font-medium"

                >

                  Logout

                </button>

              ) : (

                <>

                  <Link

                    to="/login"

                    className="py-2 px-4 border border-gray-600 text-gray-300 rounded hover:bg-gray-700 transition-colors text-sm"

                  >

                    Login

                  </Link>

                  <Link

                    to="/register"

                    className="ml-2 py-2 px-4 bg-primary-custom text-black rounded hover:bg-opacity-90 transition-colors text-sm font-medium"

                  >

                    Sign Up

                  </Link>

                </>

              )}

            </div>

          </div>

          <div className="md:hidden">

            <button onClick={() => setIsMenuOpen(!isMenuOpen)}>

              <svg className="w-6 h-6 text-white" fill="none" stroke="currentColor" viewBox="0 0 24 24" xmlns="http://www.w3.org/2000/svg">

                <path strokeLinecap="round" strokeLinejoin="round" strokeWidth="2" d={isMenuOpen ? "M6 18L18 6M6 6l12 12" : "M4 6h16M4 12h16m-7 6h7"}></path>

              </svg>

            </button>

          </div>

        </div>

        <div className={`${isMenuOpen ? 'block' : 'hidden'} md:hidden pb-4`}>

          <Link to="/" className="block py-2 text-gray-300 hover:text-white">Home</Link>

          <Link to="/about" className="block py-2 text-gray-300 hover:text-white">About Us</Link>

          <Link to="/contact" className="block py-2 text-gray-300 hover:text-white">Contact</Link>

          <Link to="/cart" className="block py-2 text-gray-300 hover:text-white">Cart</Link>

          {isLoggedIn && (

            <Link to="/board" className="block py-2 text-gray-300 hover:text-white">My Board</Link>

          )}

          <div className="flex flex-col space-y-2 mt-4">

            {isLoggedIn ? (

              <button

                onClick={handleLogout}

                className="py-2 px-4 bg-primary-custom text-black rounded hover:bg-opacity-90 transition-colors text-center font-medium"

              >

                Logout

              </button>

            ) : (

              <>

                <Link

                  to="/login"

                  className="py-2 px-4 border border-gray-600 text-gray-300 rounded hover:bg-gray-700 transition-colors text-center"

                >

                  Login

                </Link>

                <Link

                  to="/register"

                  className="py-2 px-4 bg-primary-custom text-black rounded hover:bg-opacity-90 transition-colors text-center font-medium"

                >

                  Sign Up

                </Link>

              </>

            )}

          </div>

        </div>

      </div>

    </nav>

  );

}

export default Header;

The <Link> components replace traditional anchor (<a>) tags to enable navigation **without page reloads**.

**OUTPUT :**

****

### **Step 6: Styling the Navigation Bar**

In App.css (or TailwindCSS classes):

import React from "react";

import { Link, useNavigate } from "react-router-dom";

import { useAuth } from "../hooks/useAuth"; // Import the auth hook

function Header() {

  const [isMenuOpen, setIsMenuOpen] = React.useState(false);

  const { isLoggedIn, logout } = useAuth();

  const navigate = useNavigate();

  const handleLogout = () => {

    logout();

    navigate('/');

  };

  return (

    <nav className="bg-dark-custom fixed top-0 w-full z-50  bg-transparent">

      <div className="container mx-auto px-4">

        <div className="flex justify-between items-center h-16">

          <Link className="text-2xl font-bold text-white" to="/">DrawKit</Link>

          <div className="hidden md:flex items-center space-x-6">

            <Link to="/" className="text-gray-300 hover:text-white transition-colors">Home</Link>

            <Link to="/about" className="text-gray-300 hover:text-white transition-colors">About Us</Link>

            <Link to="/contact" className="text-gray-300 hover:text-white transition-colors">Contact</Link>

            {isLoggedIn && (

              <Link to="/board" className="text-gray-300 hover:text-white transition-colors">My Board</Link>

            )}

            <div className="pl-4">

              {isLoggedIn ? (

                <button

                  onClick={handleLogout}

                  className="py-2 px-4 bg-primary-custom text-black rounded hover:bg-opacity-90 transition-colors text-sm font-medium"

                >

                  Logout

                </button>

              ) : (

                <>

                  <Link

                    to="/login"

                    className="py-2 px-4 border border-gray-600 text-gray-300 rounded hover:bg-gray-700 transition-colors text-sm"

                  >

                    Login

                  </Link>

                  <Link

                    to="/register"

                    className="ml-2 py-2 px-4 bg-primary-custom text-black rounded hover:bg-opacity-90 transition-colors text-sm font-medium"

                  >

                    Sign Up

                  </Link>

                </>

              )}

            </div>

          </div>

          <div className="md:hidden">

            <button onClick={() => setIsMenuOpen(!isMenuOpen)}>

              <svg className="w-6 h-6 text-white" fill="none" stroke="currentColor" viewBox="0 0 24 24" xmlns="http://www.w3.org/2000/svg">

                <path strokeLinecap="round" strokeLinejoin="round" strokeWidth="2" d={isMenuOpen ? "M6 18L18 6M6 6l12 12" : "M4 6h16M4 12h16m-7 6h7"}></path>

              </svg>

            </button>

          </div>

        </div>

        <div className={`${isMenuOpen ? 'block' : 'hidden'} md:hidden pb-4`}>

          <Link to="/" className="block py-2 text-gray-300 hover:text-white">Home</Link>

          <Link to="/about" className="block py-2 text-gray-300 hover:text-white">About Us</Link>

          <Link to="/contact" className="block py-2 text-gray-300 hover:text-white">Contact</Link>

          <Link to="/cart" className="block py-2 text-gray-300 hover:text-white">Cart</Link>

          {isLoggedIn && (

            <Link to="/board" className="block py-2 text-gray-300 hover:text-white">My Board</Link>

          )}

          <div className="flex flex-col space-y-2 mt-4">

            {isLoggedIn ? (

              <button

                onClick={handleLogout}

                className="py-2 px-4 bg-primary-custom text-black rounded hover:bg-opacity-90 transition-colors text-center font-medium"

              >

                Logout

              </button>

            ) : (

              <>

                <Link

                  to="/login"

                  className="py-2 px-4 border border-gray-600 text-gray-300 rounded hover:bg-gray-700 transition-colors text-center"

                >

                  Login

                </Link>

                <Link

                  to="/register"

                  className="py-2 px-4 bg-primary-custom text-black rounded hover:bg-opacity-90 transition-colors text-center font-medium"

                >

                  Sign Up

                </Link>

              </>

            )}

          </div>

        </div>

      </div>

    </nav>

  );

}

export default Header;

**OUTPUT :**

****

## **Code:**

Home page:

import { Link } from 'react-router-dom';

import FeatureCarousel from '../components/FeatureCarousel';

import FAQ from '../components/FAQ';

const HeroImg = "https://placehold.co/600x400/121212/a8a5ff/png?text=DrawKit+App";

function Home() {

  return (

    <div className="text-white pt-16">

      <section className="relative z-10 pt-14 mt-14 flex flex-col justify-center items-center gap-5 container mx-auto px-4">

        <div className="text-2xl sm:text-4xl md:text-5xl font-bold px-1 text-center mt-10 md:mt-0">

          <p>Unleash Your Ideas, Together.</p>

          <p>

            <span className="text-primary-custom">DrawKit</span> is Your Smart Investment

          </p>

        </div>

        <div className="text-center max-w-xl text-gray-400 text-sm md:text-base">

          <p>

            DrawKit is a real-time collaborative whiteboard for brainstorming,

            wireframing, and bringing your team's vision to life. Effortlessly.

          </p>

        </div>

        <div className="space-y-1 flex flex-col">

          <Link

            to="/register"

            className="bg-primary-custom text-black py-2 md:py-3 px-2 md:px-5 rounded-md hover:bg-opacity-90 transition-colors md:text-base text-sm font-medium"

          >

            Start Drawing Now

          </Link>

          <p className="text-[0.7rem] text-gray-400 text-center">

            No credit card required

          </p>

        </div>

        <div className="max-w-6xl mt-5">

          <img

            src={HeroImg}

            alt="DrawKit App"

            className="rounded-lg shadow-2xl"

          />

        </div>

      </section>

      <section className="relative z-10 flex flex-col justify-center items-center pt-24 gap-5 container mx-auto px-4">

        <div className="inline-block px-4 py-1.5 rounded-full bg-primary-custom/10 text-primary-custom text-sm font-medium">

          <span>Features</span>

        </div>

        <div className="text-3xl md:text-4xl text-center font-semibold">

          <p>

            Why teams{" "}

            <span className="text-primary-custom bg-primary-custom/10 px-2">

              love

            </span>{" "}

            DrawKit

          </p>

        </div>

        <div className="max-w-xl md:max-w-md text-gray-400 text-center text-sm md:text-base">

          <p>

            DrawKit offers comprehensive tools, combining real-time collaboration

            with powerful analytics to keep your team's ideas flowing.

          </p>

        </div>

        <FeatureCarousel />

      </section>

      <section className="relative z-10 flex flex-col lg:flex-row justify-center items-center pt-24 pb-20 gap-5 container mx-auto px-4">

        <FAQ />

      </section>

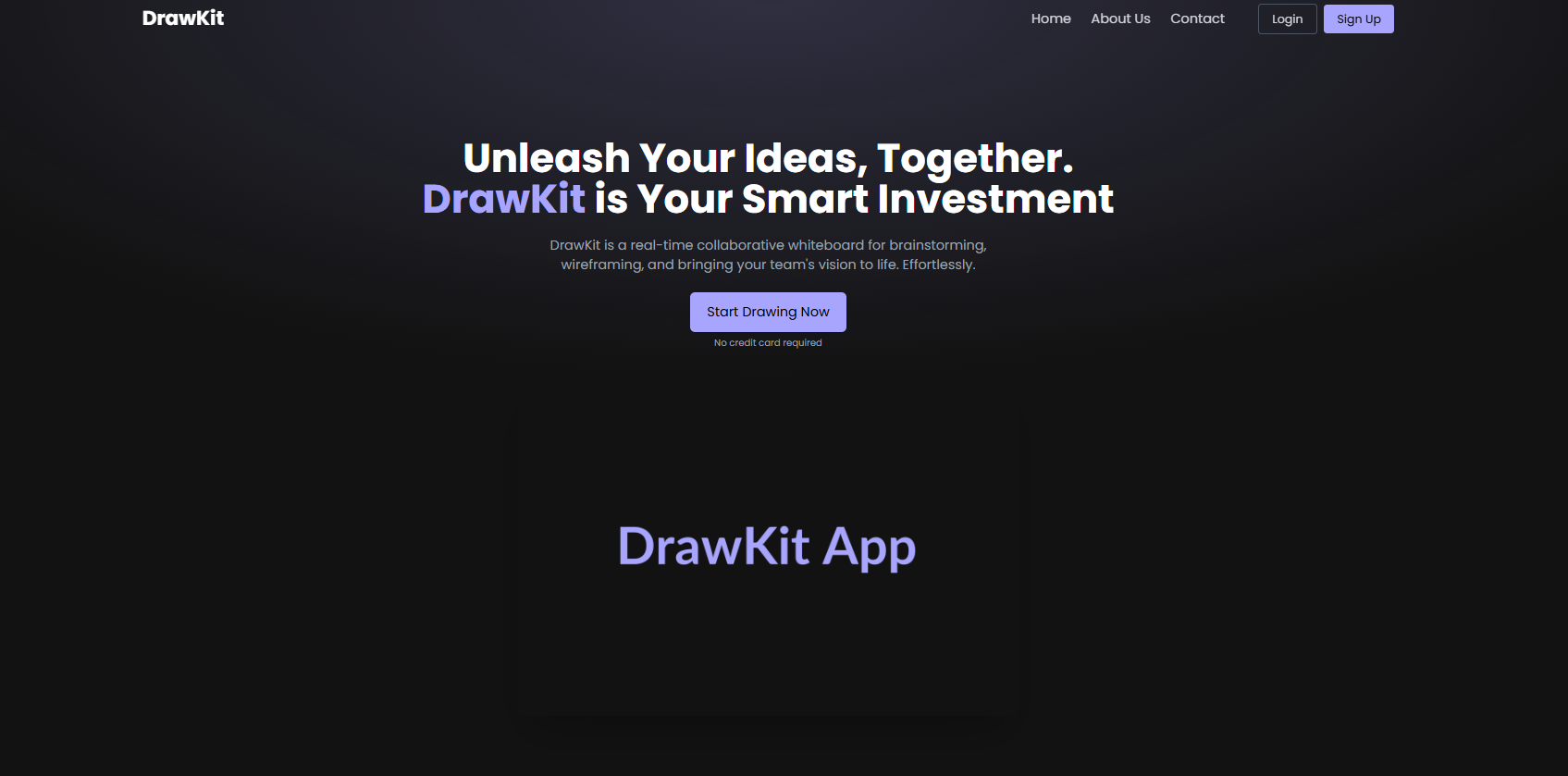
    </div>

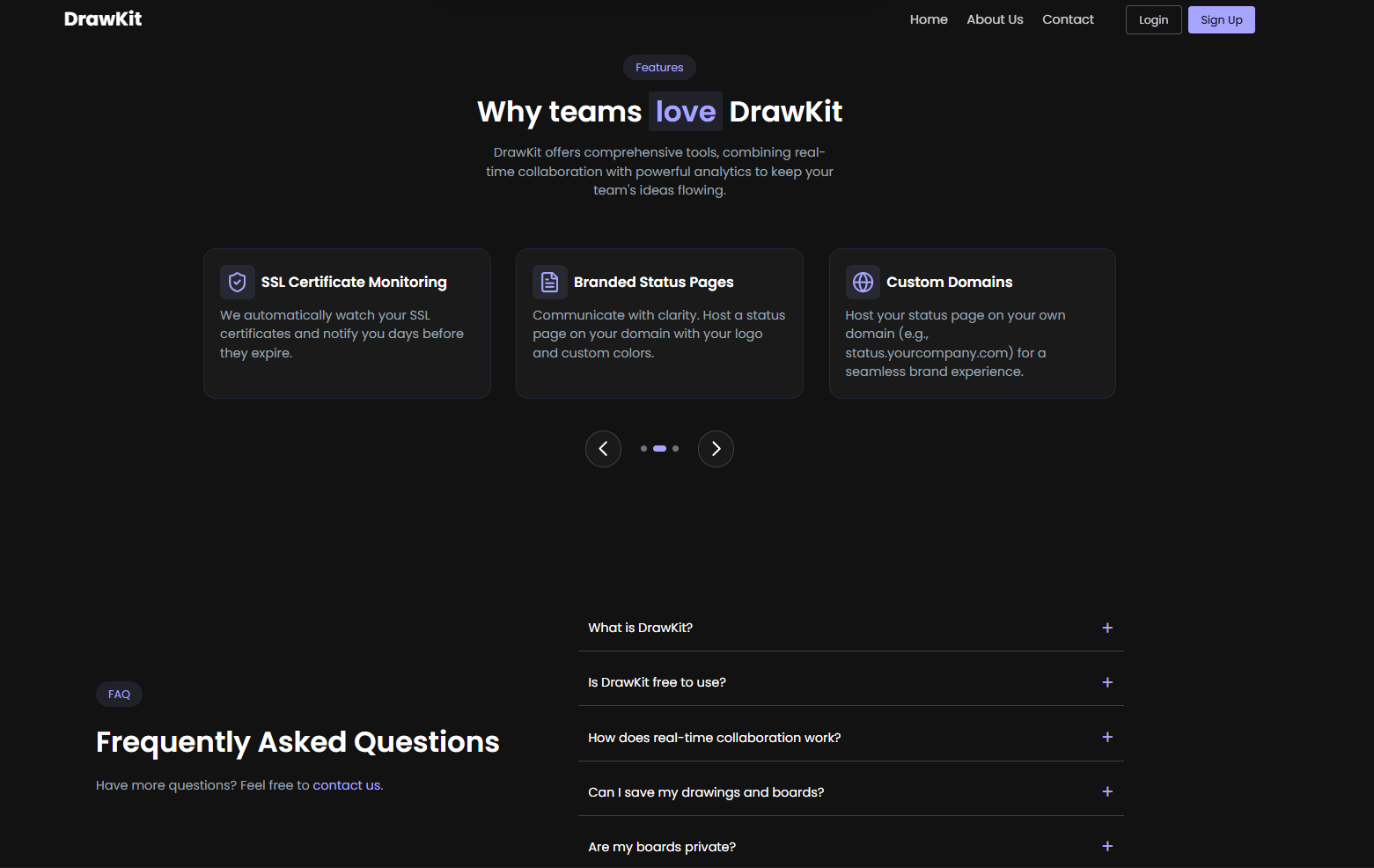
  )

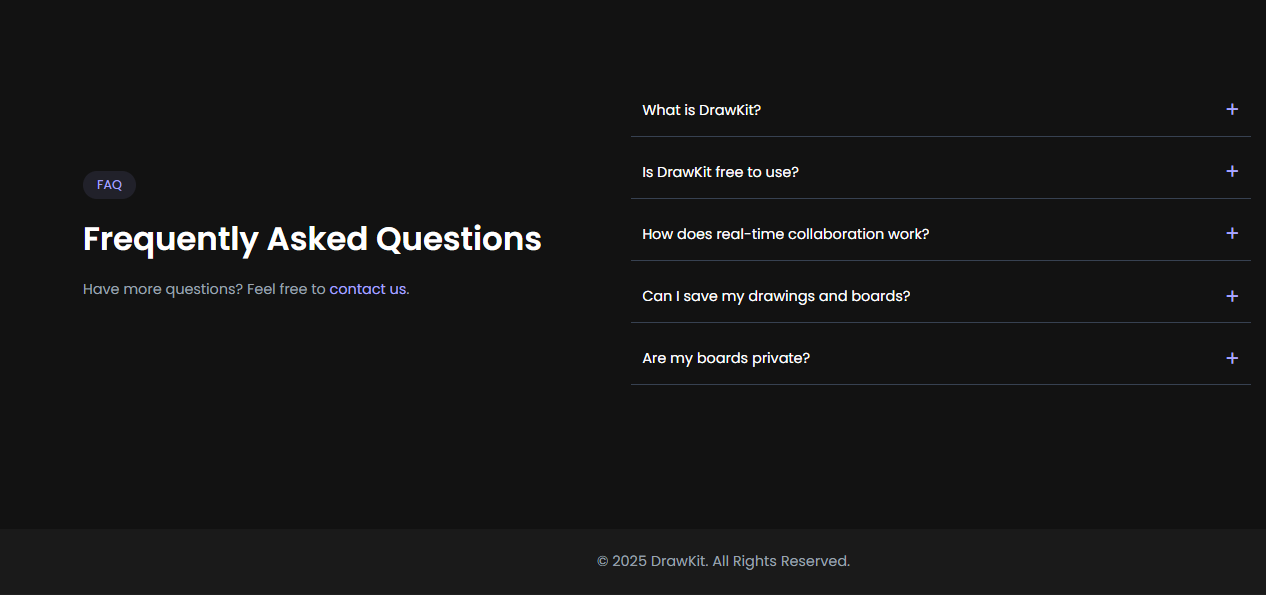
}

export default Home;

## **Output:**







## **Code:**

B. cart page:

import PricingSection from '../components/PricingSection'

function Cart() {

  return (

    <section className="relative z-10 flex flex-col justify-center items-center pt-24 gap-5 container mx-auto px-4 text-white">

        <div className="text-3xl md:text-4xl text-center font-semibold space-y-2">

          <p>Simple, transparent pricing</p>

          <p>

            that{" "}

            <span className="text-primary-custom bg-primary-custom/10 px-2">

              scales

            </span>{" "}

            with you.

          </p>

        </div>

        <div className="max-w-xs md:max-w-md text-gray-400 text-center text-sm md:text-base">

          <p>

            Get started for free. No credit card required.

          </p>

        </div>

        <PricingSection />

      </section>

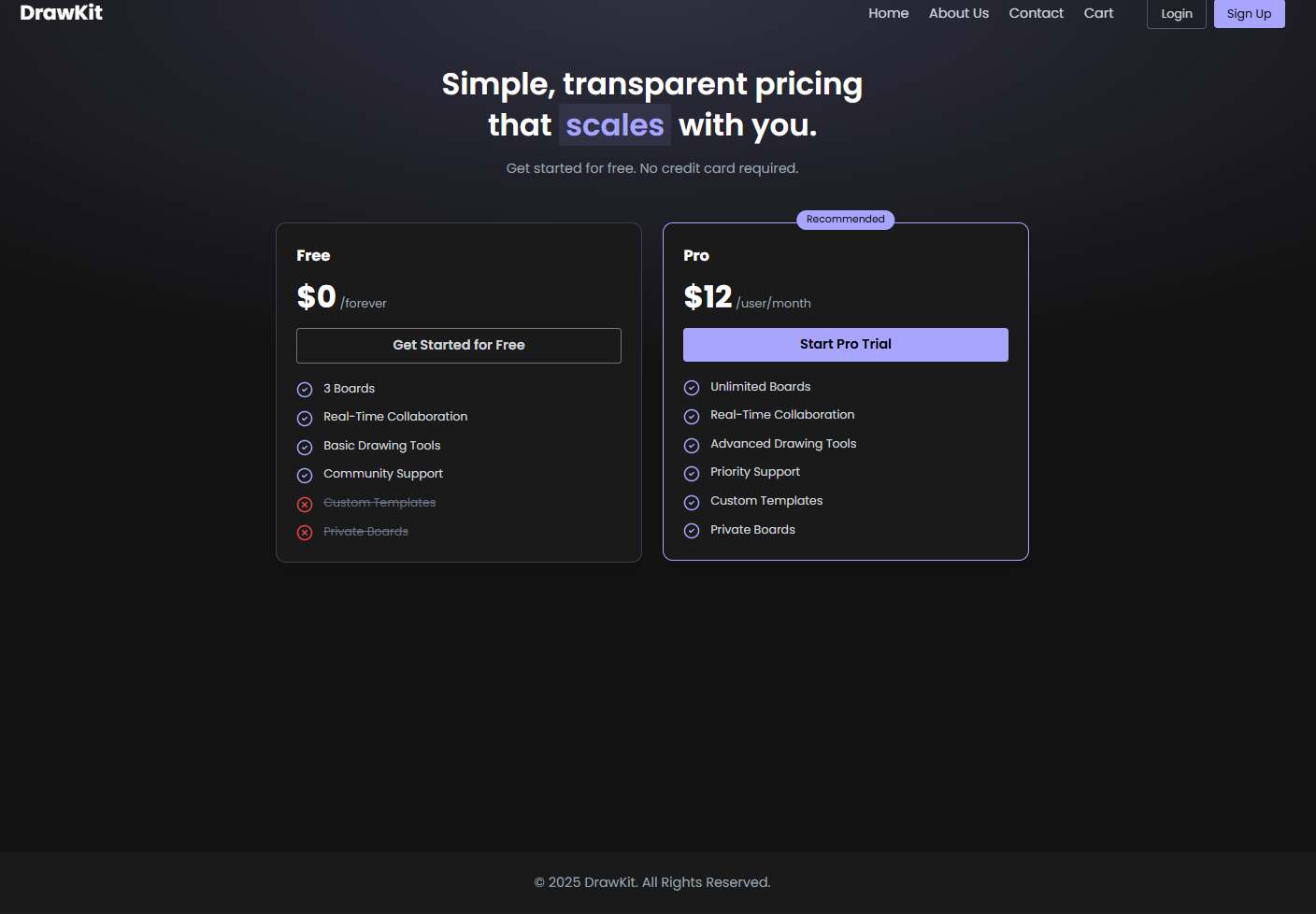
  )

}

export default Cart

## **Output:**

B. menu/room page output:



## **Code:**

contact page:

function Contact() {

  return (

    <section className="py-20 pt-36 container mx-auto px-4">

      <div className="text-center mb-12">

        <h2 className="text-3xl md:text-4xl font-bold text-white" >Get in Touch</h2>

        <p className="text-lg text-gray-400">We'd love to hear from you.</p>

      </div>

      <div className="max-w-3xl mx-auto">

        <form className="space-y-6">

          <div className="grid md:grid-cols-2 gap-6">

            <div>

              <label htmlFor="fullName" className="block text-sm font-medium text-gray-300 mb-2">Full Name</label>

              <input

                type="text"

                id="fullName"

                className="w-full px-4 py-3 bg-dark-custom border border-gray-700 rounded-md text-white focus:outline-none focus:ring-2 focus:ring-primary-custom"

                required

              />

            </div>

            <div>

              <label htmlFor="email" className="block text-sm font-medium text-gray-300 mb-2">Email Address</label>

              <input

                type="email"

                id="email"

                className="w-full px-4 py-3 bg-dark-custom border border-gray-700 rounded-md text-white focus:outline-none focus:ring-2 focus:ring-primary-custom"

                required

              />

            </div>

          </div>

          <div>

            <label htmlFor="department" className="block text-sm font-medium text-gray-300 mb-2">Department</label>

            <select

              id="department"

              className="w-full px-4 py-3 bg-dark-custom border border-gray-700 rounded-md text-white focus:outline-none focus:ring-2 focus:ring-primary-custom"

            >

              <option>Choose...</option>

              <option value="1">Sales</option>

              <option value="2">Support</option>

              <option value="3">General Inquiry</option>

            </select>

          </div>

          <div>

            <label htmlFor="message" className="block text-sm font-medium text-gray-300 mb-2">Your Message</label>

            <textarea

              id="message"

              rows={5}

              className="w-full px-4 py-3 bg-dark-custom border border-gray-700 rounded-md text-white focus:outline-none focus:ring-2 focus:ring-primary-custom"

              required

            ></textarea>

          </div>

          <div className="flex items-center">

            <input

              id="newsletterSwitch"

              type="checkbox"

              className="w-4 h-4 text-primary-custom bg-gray-700 border-gray-600 rounded focus:ring-primary-custom"

              defaultChecked

            />

            <label htmlFor="newsletterSwitch" className="ml-3 text-sm font-medium text-gray-300">

              Subscribe to our newsletter

            </label>

          </div>

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

            <button

              type="submit"

              className="py-3 px-8 bg-primary-custom text-black rounded-md hover:bg-opacity-90 transition-colors font-medium text-lg"

            >

              Submit Form

            </button>

          </div>

        </form>

      </div>

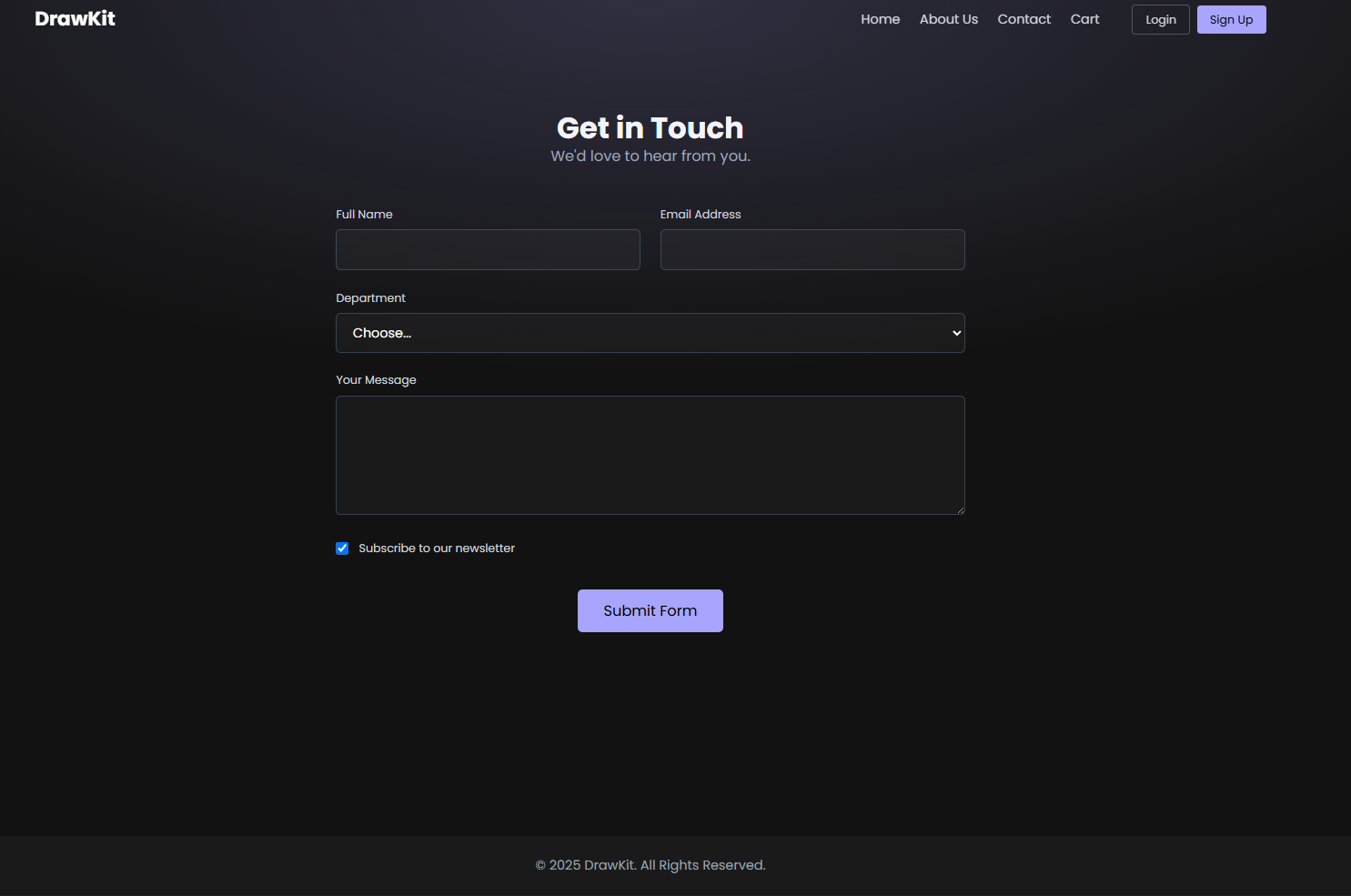
    </section>

  )

}

export default Contact

## **Output:**



## **Code:**

About us page:

import { Link } from 'react-router-dom'

import { Users, Eye, Lightbulb, Rocket } from 'lucide-react'

const teamMembers = [

  {

    name: "Shrinivas Sherikar",

    role: "Lead Developer, Desginer ",

    img: "https://placehold.co/400x400/1a1a1a/a8a5ff/png?text=AJ"

  }

];

function About() {

  return (

    <div className="pt-36 pb-20 container mx-auto px-4 text-white">

      <div className="max-w-4xl mx-auto">

        <div className="text-center mb-16">

          <h1 className="text-4xl md:text-5xl font-bold mb-4">About DrawKit</h1>

          <p className="text-xl text-gray-300">

            We're on a mission to make collaboration visual, intuitive, and accessible for everyone.

          </p>

        </div>

        <div className="grid md:grid-cols-2 gap-12 items-center mb-20">

          <div>

            <h2 className="text-3xl font-semibold text-primary-custom mb-4">Our Story</h2>

            <p className="text-gray-400 mb-4">

              DrawKit started as a simple idea: what if teams could brainstorm as freely online as they do on a physical whiteboard? Frustrated by clunky and restrictive tools, our founders set out to build a platform that was fluid, fast, and, most importantly, collaborative at its core.

            </p>

            <p className="text-gray-400">

              Today, DrawKit is used by thousands of teams—from startups to Fortune 500 companies—to bring their best ideas to life.

            </p>

          </div>

          <div>

            <img

              src="https://placehold.co/500x350/1a1a1a/a8a5ff/png?text=Our+Team"

              className="rounded-lg shadow-2xl"

              alt="Our team collaborating"

            />

          </div>

        </div>

        <div className="text-center mb-20">

          <h2 className="text-3xl font-semibold text-primary-custom mb-12">Our Core Values</h2>

          <div className="grid md:grid-cols-3 gap-8">

            <div className="bg-dark-custom p-8 rounded-lg shadow-lg border border-gray-800">

              <Eye size={40} className="text-primary-custom mx-auto mb-4" />

              <h3 className="text-2xl font-bold mb-3">Simplicity</h3>

              <p className="text-gray-400">

                Great tools should be easy to use. We focus on an intuitive experience that gets out of your way.

              </p>

            </div>

            <div className="bg-dark-custom p-8 rounded-lg shadow-lg border border-gray-800">

              <Users size={40} className="text-primary-custom mx-auto mb-4" />

              <h3 className="text-2xl font-bold mb-3">Collaboration</h3>

              <p className="text-gray-400">

                We believe the best ideas are built together. Our platform is designed for seamless teamwork.

              </p>

            </div>

            <div className="bg-dark-custom p-8 rounded-lg shadow-lg border border-gray-800">

              <Lightbulb size={40} className="text-primary-custom mx-auto mb-4" />

              <h3 className="text-2xl font-bold mb-3">Innovation</h3>

              <p className="text-gray-400">

                We're constantly pushing boundaries to create new and better ways for you to visualize and create.

              </p>

            </div>

          </div>

        </div>

        <div className="text-center mb-20">

          <h2 className="text-3xl font-semibold text-primary-custom mb-12">Meet the Team</h2>

          <div className="grid">

            {teamMembers.map((member) => (

              <div key={member.name} className="bg-dark-custom p-6 rounded-lg shadow-lg border border-gray-800">

                <img

                  src={member.img}

                  alt={member.name}

                  className="w-32 h-32 rounded-full mx-auto mb-4 border-4 border-gray-700"

                />

                <h3 className="text-xl font-bold text-white">{member.name}</h3>

                <p className="text-primary-custom">{member.role}</p>

              </div>

            ))}

          </div>

        </div>

        <div className="text-center bg-dark-custom border border-gray-800 p-10 rounded-lg">

           <Rocket size={48} className="text-primary-custom mx-auto mb-4" />

          <h2 className="text-3xl font-bold mb-4">Ready to Start Drawing?</h2>

          <p className="text-lg text-gray-300 mb-8 max-w-lg mx-auto">

            Join thousands of teams building their next big idea on DrawKit.

            Sign up for free, no credit card required.

          </p>

          <Link

            to="/register"

            className="py-3 px-8 bg-primary-custom text-black rounded-md hover:bg-opacity-90 transition-colors font-medium text-lg"

          >

            Start for Free

          </Link>

        </div>

      </div>

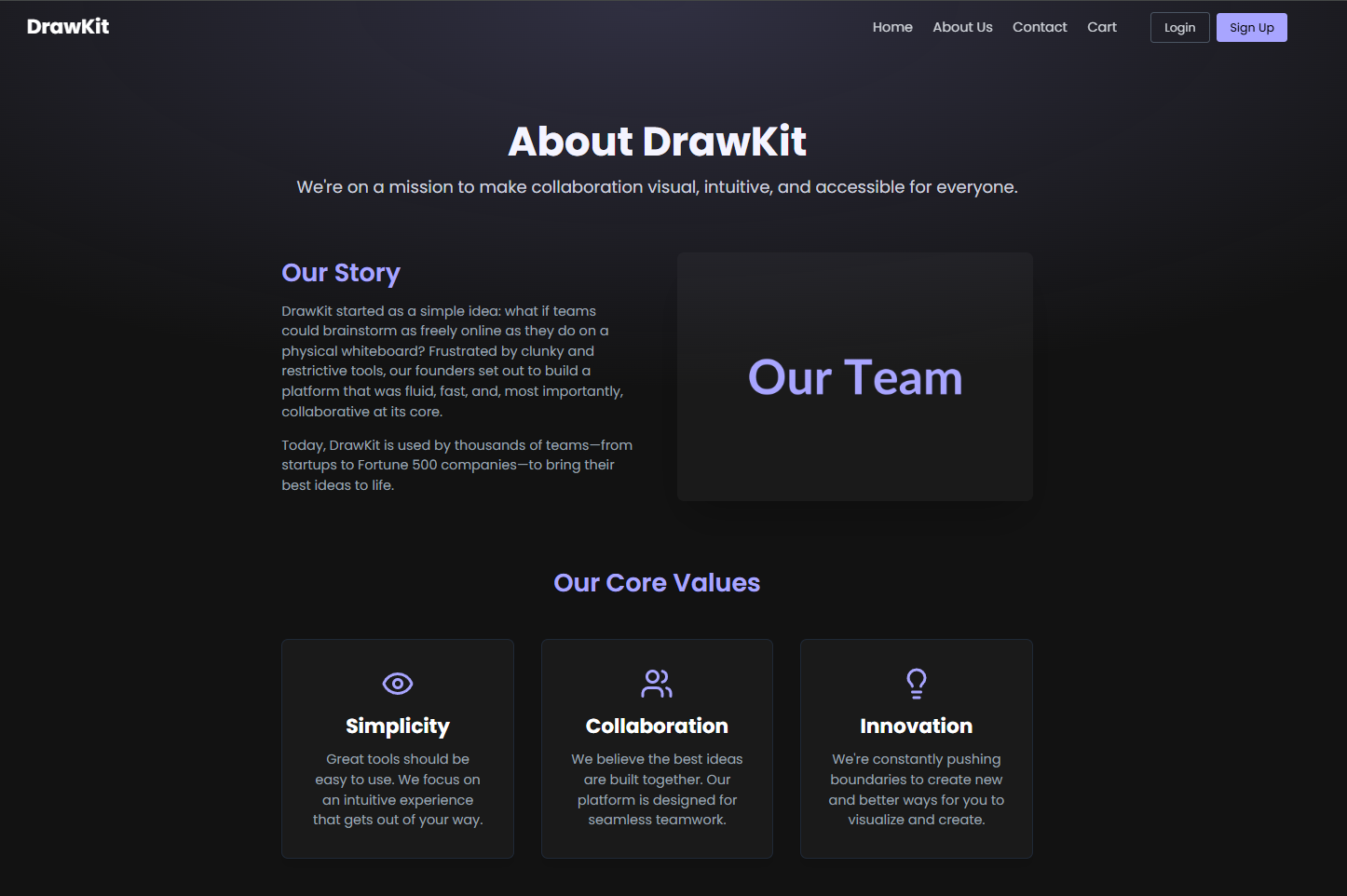
    </div>

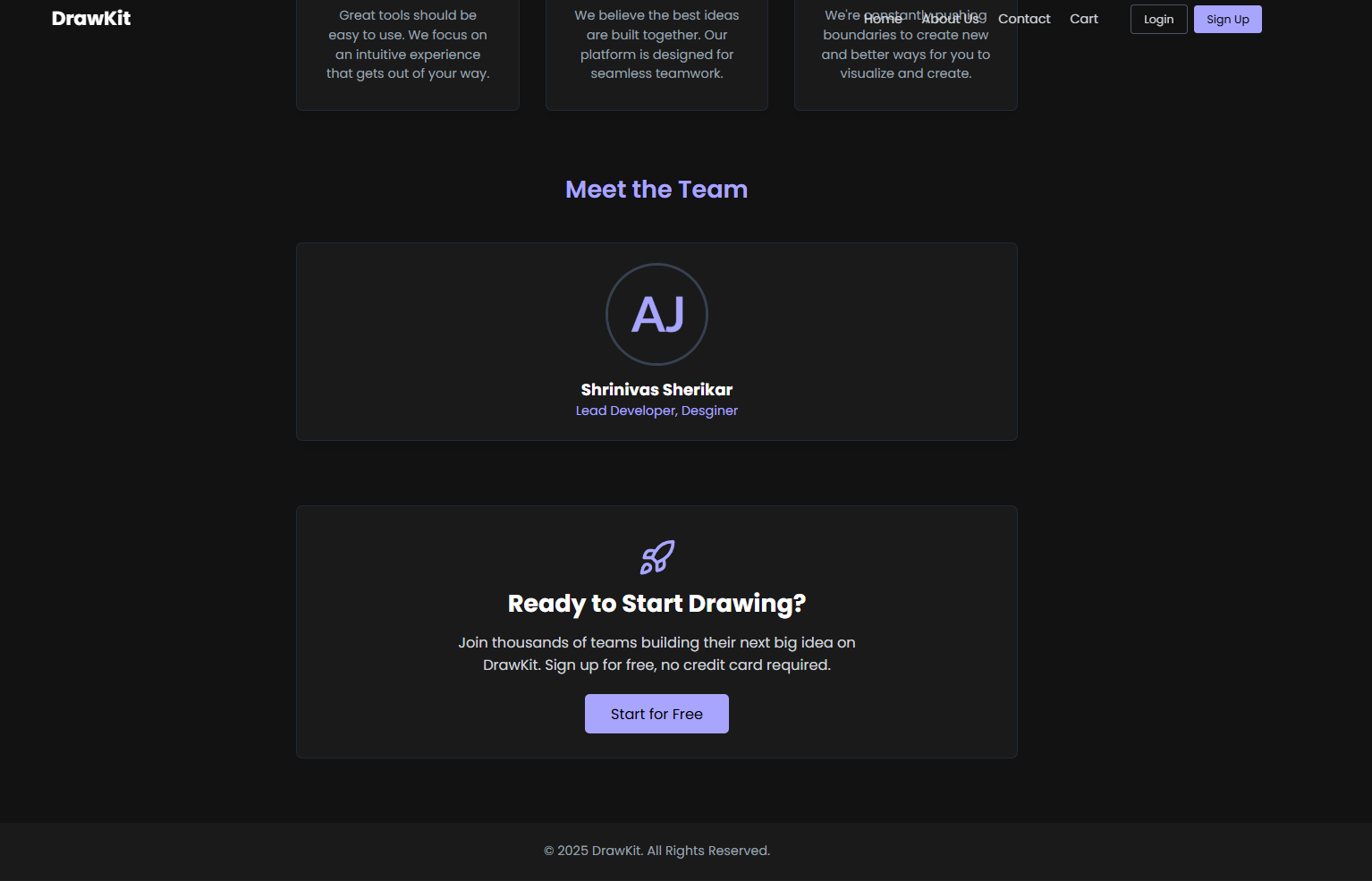
  )

}

export default About

**OUTPUT :**





**Code :**

Login page

import React from 'react'

import { Link, useNavigate } from 'react-router-dom'

function Login() {

  const navigate = useNavigate();

  const handleDemoLogin = () => {

    localStorage.setItem('userAuthToken', 'demo-user-token');

    localStorage.setItem('username', 'Demo User');

window.location.reload()

    navigate('/board');

  };

  const handleSubmit = (e: React.FormEvent) => {

    e.preventDefault();

    handleDemoLogin();

  };

  return (

    <div className="flex items-center justify-center min-h-screen pt-16">

      <div className="w-full max-w-md p-8 space-y-8 bg-dark-custom rounded-lg shadow-2xl border border-gray-800 mx-4">

        <h2 className="text-3xl font-bold text-center text-white">

          Welcome Back

        </h2>

        <form className="space-y-6" onSubmit={handleSubmit}>

          <div>

            <label htmlFor="email" className="block text-sm font-medium text-gray-300 mb-2">

              Email Address

            </label>

            <input

              type="email"

              id="email"

              className="w-full px-4 py-3 bg-darker-custom border border-gray-700 rounded-md text-white focus:outline-none focus:ring-2 focus:ring-primary-custom"

              required

              defaultValue="demo@user.com"

            />

          </div>

          <div>

            <label htmlFor="password"className="block text-sm font-medium text-gray-300 mb-2">

              Password

            </label>

            <input

              type="password"

              id="password"

              className="w-full px-4 py-3 bg-darker-custom border border-gray-700 rounded-md text-white focus:outline-none focus:ring-2 focus:ring-primary-custom"

              required

              defaultValue="password123"

            />

          </div>

          <div className="flex items-center justify-between">

            <div className="flex items-center">

              <input

                id="remember-me"

                type="checkbox"

                className="w-4 h-4 text-primary-custom bg-gray-700 border-gray-600 rounded focus:ring-primary-custom"

              />

              <label

                htmlFor="remember-me"

                className="ml-2 block text-sm text-gray-400"

              >

                Remember me

              </label>

            </div>

            <div className="text-sm">

              <a href="#" className="font-medium text-primary-custom hover:underline">

                Forgot password?

              </a>

            </div>

          </div>

          <div>

            <button

              type="submit"

              className="w-full py-3 px-4 bg-primary-custom text-black rounded-md hover:bg-opacity-90 transition-colors font-medium text-lg"

            >

              Login

            </button>

          </div>

          <div className="flex items-center">

            <div className="flex-grow border-t border-gray-700"></div>

            <span className="flex-shrink mx-4 text-gray-400">OR</span>

            <div className="flex-grow border-t border-gray-700"></div>

          </div>

          <div>

            <button

              type="button"

              onClick={handleDemoLogin}

              className="w-full py-3 px-4 bg-darker-custom text-primary-custom border border-primary-custom rounded-md hover:bg-primary-custom hover:text-black transition-colors font-medium text-lg"

            >

              Login as Demo User

            </button>

          </div>

        </form>

        <p className="text-sm text-center text-gray-400">

          Don't have an account?{' '}

          <Link to="/register" className="font-medium text-primary-custom hover:underline">

            Sign Up

          </Link>

        </p>

      </div>

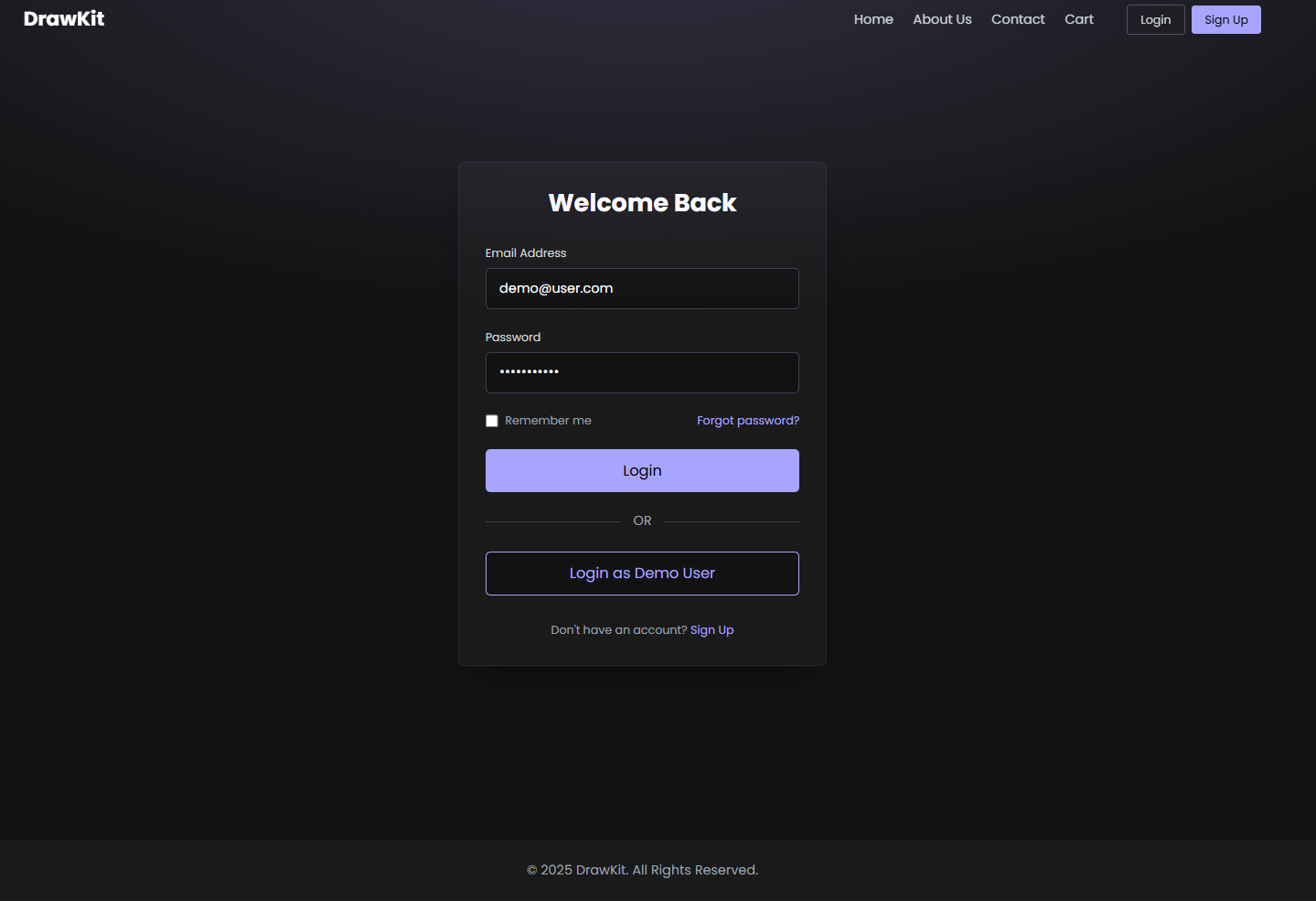
    </div>

  )

}

export default Login

**OUTPUT :**



**Code:**

Register page

import { Link } from 'react-router-dom'

function Signup() {

  return (

    <div className="flex items-center justify-center min-h-screen pt-24 pb-10">

      <div className="w-full max-w-md p-8 space-y-8 bg-dark-custom rounded-lg shadow-2xl border border-gray-800 mx-4">

        <h2 className="text-3xl font-bold text-center text-white">

          Create Your Account

        </h2>

        <form className="space-y-6">

          <div>

            <label htmlFor="name" className="block text-sm font-medium text-gray-300 mb-2">

              Full Name

            </label>

            <input

              type="text"

              id="name"

              className="w-full px-4 py-3 bg-darker-custom border border-gray-700 rounded-md text-white focus:outline-none focus:ring-2 focus:ring-primary-custom"

              required

            />

          </div>

          <div>

            <label htmlFor="email" className="block text-sm font-medium text-gray-300 mb-2">

              Email Address

            </label>

            <input

              type="email"

              id="email"

              className="w-full px-4 py-3 bg-darker-custom border border-gray-700 rounded-md text-white focus:outline-none focus:ring-2 focus:ring-primary-custom"

              required

            />

          </div>

          <div>

            <label htmlFor="password"className="block text-sm font-medium text-gray-300 mb-2">

              Password

            </label>

            <input

              type="password"

              id="password"

              className="w-full px-4 py-3 bg-darker-custom border border-gray-700 rounded-md text-white focus:outline-none focus:ring-2 focus:ring-primary-custom"

              required

            />

          </div>

          <div>

            <button

              type="submit"

              className="w-full py-3 px-4 bg-primary-custom text-black rounded-md hover:bg-opacity-90 transition-colors font-medium text-lg"

            >

              Sign Up

            </button>

          </div>

        </form>

        <p className="text-sm text-center text-gray-400">

          Already have an account?{' '}

          <Link to="/login" className="font-medium text-primary-custom hover:underline">

            Login

          </Link>

        </p>

      </div>

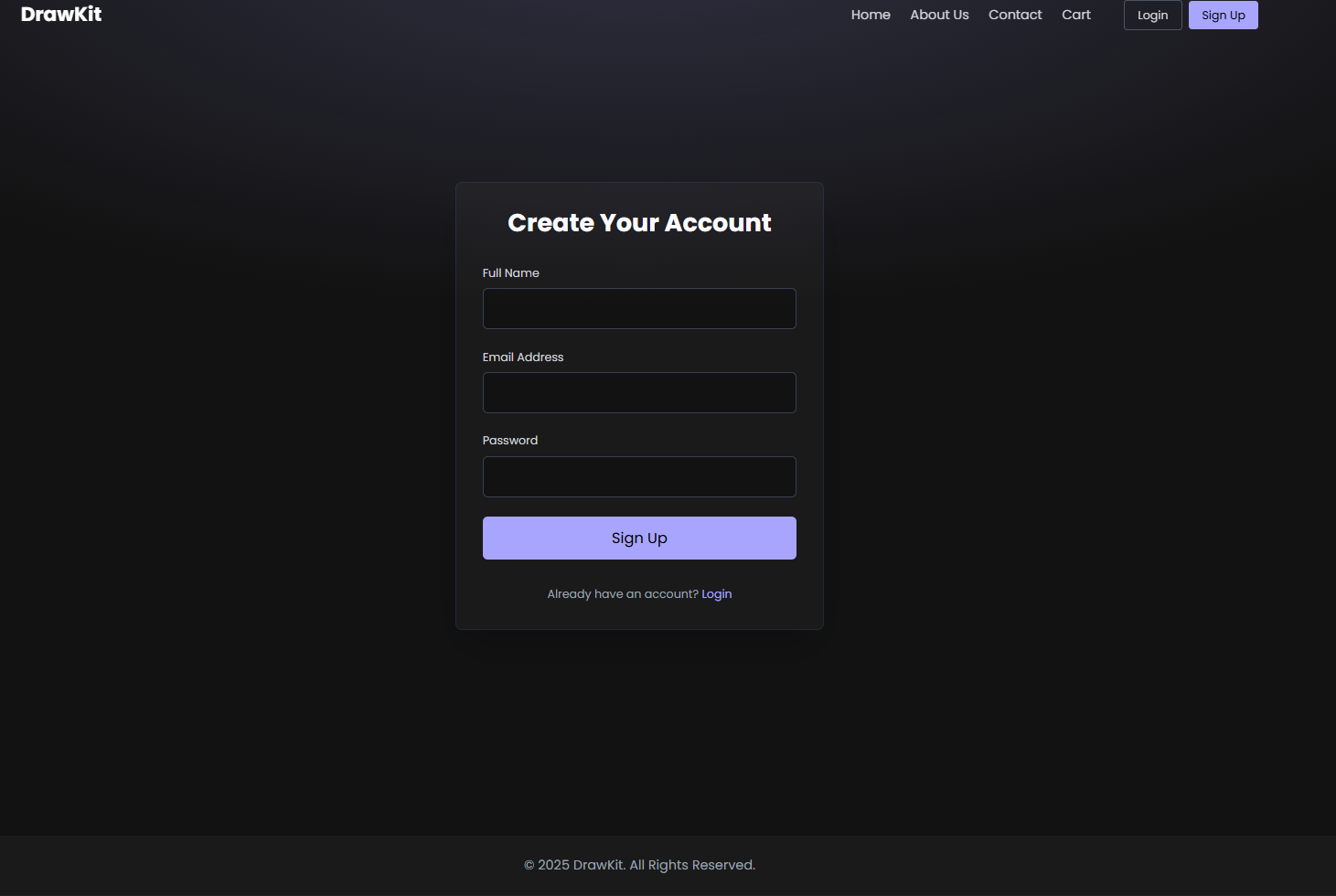
    </div>

  )

}

export default Signup

**OUTPUT :**



**CONCLUSION :**

In this experiment, we successfully set up a ReactJS environment and implemented a basic navigation system for a web-based music listening platform. By using **React Router DOM**, we enabled seamless transitions between multiple pages — such as Home, Explore, Create Room, Join Room, and Profile — without reloading the browser, demonstrating the concept of a **Single Page Application (SPA)**.

We learned how to structure a React project effectively, organize components and pages, and integrate a responsive **navigation bar** using Bootstrap/Tailwind for styling. This hands-on setup laid the foundation for developing a dynamic, scalable, and interactive music-sharing website where users can create or join listening rooms in real-time.

**Experiment 2: Home Page Implementation**

### **Title:**

### Design and Implementation of the Home Page in *DrawKit – Collaborative Whiteboarding Tool*

### **Objective:**

To design a responsive and user-friendly **Home Page** that showcases highlighted features or tools, implements a product (or feature) carousel, and includes a search bar for easy navigation and accessibility.

### **Theory:**

The **Home Page** serves as the entry point and the most important part of any web application. It provides the first impression and directs users to different functionalities of the system. In the context of **DrawKit**, which is a collaborative whiteboarding tool, the home page acts as a dashboard for users to explore different collaborative features such as sketching, brainstorming, and wireframing.

A well-designed **Home Page** in a React application ensures:

* **Quick access** to primary tools or features.
* **Visual appeal** through the use of carousels or banners.
* **Ease of navigation** using search bars and category highlights.
* **Interactive UI** to engage users effectively.

The **Product Carousel** (or Feature Showcase) is an interactive slider that allows users to view multiple items (such as templates, boards, or design tools) in a compact space. It enhances user engagement and makes the interface dynamic.

A **Search Bar** enables users to quickly find specific boards, templates, or tools by typing keywords. This improves the efficiency of user interaction and enhances the accessibility of the system.

**Key UI/UX Components of the Home Page:**

1. **Header Section:** Includes logo and navbar links.
2. **Carousel / Featured Section:** Displays highlighted tools or templates (e.g., “Brainstorming Board”, “Wireframe Creator”, “Flowchart Builder”).
3. **Search Bar:** Allows users to search for a board, tool, or team workspace.
4. **Category Highlights:** Cards or tiles representing different whiteboarding categories.
5. **Footer Section:** Contains quick links and contact information.

**Advantages of a Well-Designed Home Page:**

* Enhances the user’s first impression.
* Provides quick navigation to main tools.
* Improves usability and accessibility.
* Encourages user engagement with visual elements.

### **Tools and Technologies Used:**

* **Frontend:** ReactJS
* **Styling:** CSS / Tailwind CSS / Bootstrap
* **Carousel:** React Slick / Swiper.js
* **Search Functionality:** JavaScript (filter-based search)
* **IDE:** Visual Studio Code
* **Browser:** Google Chrome

### **Procedure / Tasks Performed:**

1. Create a new component named Home.js in the src folder.
2. Import necessary libraries for carousel functionality (e.g., React Slick).
3. Design a **carousel section** displaying featured products or tools.
4. Add a **search bar** at the top of the home page using an <input> field.
5. Implement a search function to filter displayed items based on user input.
6. Add **category cards** below the carousel to represent various sections.
7. Apply CSS or Tailwind styling to ensure a visually appealing, responsive layout.
8. Test the page in a browser using npm start to ensure proper functionality.

### **Code Snippet:**

import { Link } from 'react-router-dom';

import FeatureCarousel from '../components/FeatureCarousel';

import FAQ from '../components/FAQ';

const HeroImg = "https://placehold.co/600x400/121212/a8a5ff/png?text=DrawKit+App";

function Home() {

  return (

    <div className="text-white pt-16">

      <section className="relative z-10 pt-14 mt-14 flex flex-col justify-center items-center gap-5 container mx-auto px-4">

        <div className="text-2xl sm:text-4xl md:text-5xl font-bold px-1 text-center mt-10 md:mt-0">

          <p>Unleash Your Ideas, Together.</p>

          <p>

            <span className="text-primary-custom">DrawKit</span> is Your Smart Investment

          </p>

        </div>

        <div className="text-center max-w-xl text-gray-400 text-sm md:text-base">

          <p>

            DrawKit is a real-time collaborative whiteboard for brainstorming,

            wireframing, and bringing your team's vision to life. Effortlessly.

          </p>

        </div>

        <div className="space-y-1 flex flex-col">

          <Link

            to="/register"

            className="bg-primary-custom text-black py-2 md:py-3 px-2 md:px-5 rounded-md hover:bg-opacity-90 transition-colors md:text-base text-sm font-medium"

          >

            Start Drawing Now

          </Link>

          <p className="text-[0.7rem] text-gray-400 text-center">

            No credit card required

          </p>

        </div>

        <div className="max-w-6xl mt-5">

          <img

            src={HeroImg}

            alt="DrawKit App"

            className="rounded-lg shadow-2xl"

          />

        </div>

      </section>

      <section className="relative z-10 flex flex-col justify-center items-center pt-24 gap-5 container mx-auto px-4">

        <div className="inline-block px-4 py-1.5 rounded-full bg-primary-custom/10 text-primary-custom text-sm font-medium">

          <span>Features</span>

        </div>

        <div className="text-3xl md:text-4xl text-center font-semibold">

          <p>

            Why teams{" "}

            <span className="text-primary-custom bg-primary-custom/10 px-2">

              love

            </span>{" "}

            DrawKit

          </p>

        </div>

        <div className="max-w-xl md:max-w-md text-gray-400 text-center text-sm md:text-base">

          <p>

            DrawKit offers comprehensive tools, combining real-time collaboration

            with powerful analytics to keep your team's ideas flowing.

          </p>

        </div>

        <FeatureCarousel />

      </section>

      <section className="relative z-10 flex flex-col lg:flex-row justify-center items-center pt-24 pb-20 gap-5 container mx-auto px-4">

        <FAQ />

      </section>

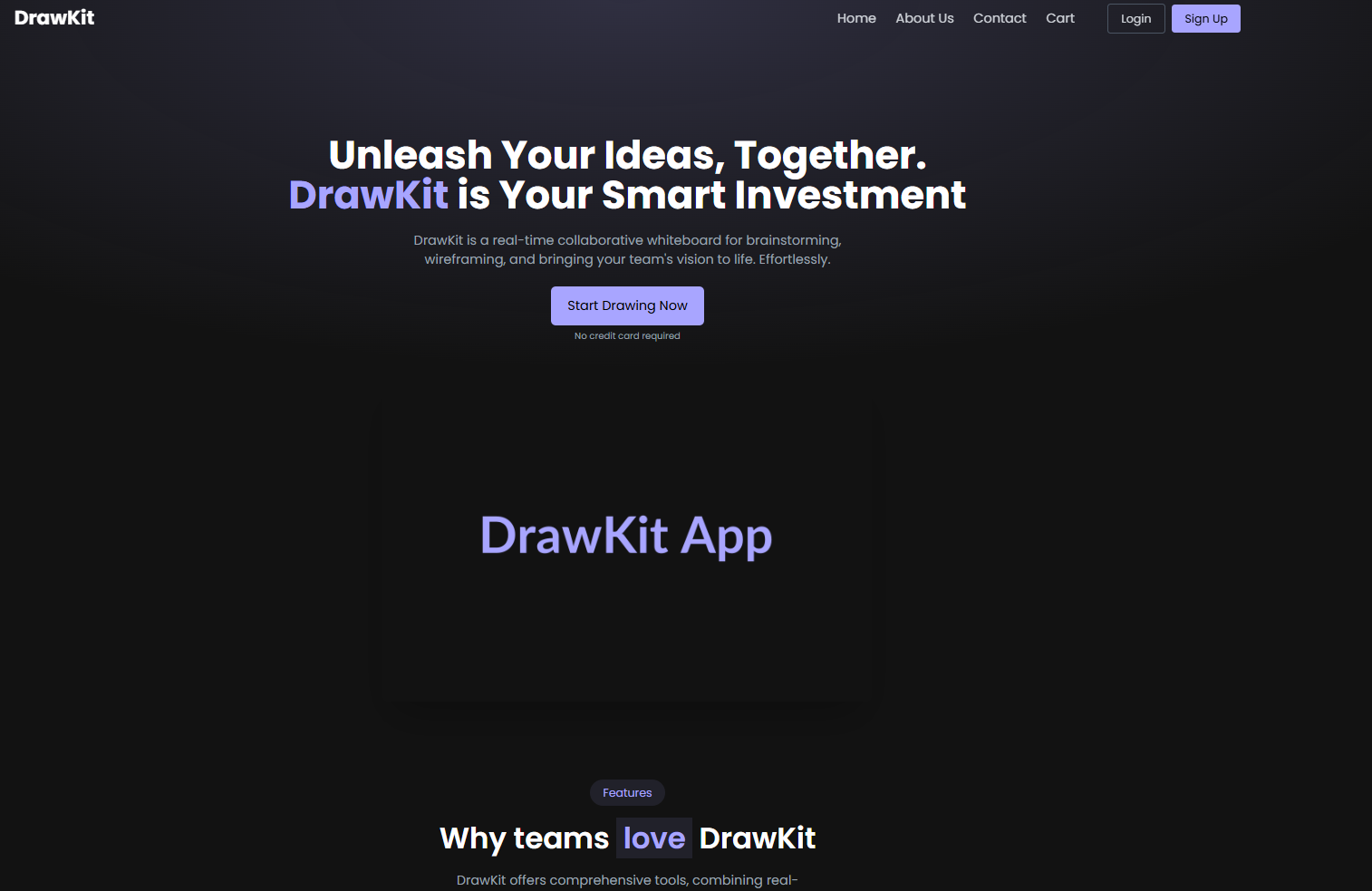
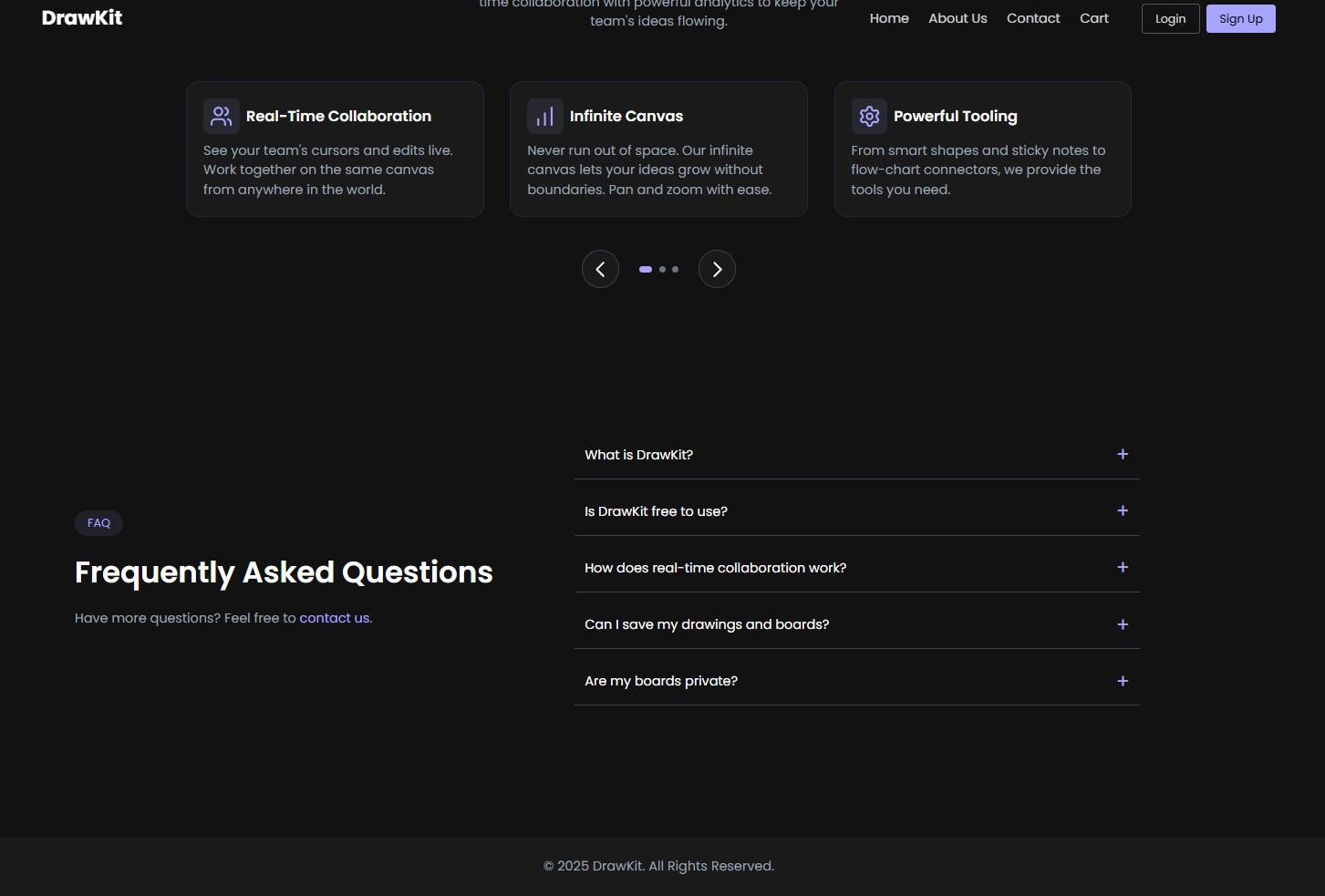
    </div>

  )

}

export default Home;

**Output:**

**** ****

### **CONCLUSION:**

Successfully designed and implemented a **Home Page** featuring a dynamic carousel, a search bar, and category highlights for the *DrawKit collaborative whiteboarding tool*. The page provides an engaging interface and enhances user experience.

## **Experiment 3: Product Listing Page Implementation**

### **Title:**

Design and Implementation of the Product Listing Page in *DrawKit – Collaborative Whiteboarding Tool*

**Objective:**

To create a dynamic **Product Listing Page** that fetches data from an API or static JSON file, displays products with essential details such as image, name, and price, and includes an **Add to Cart** button along with navigation to individual product detail pages using **React Router**.

### **Theory:**

A **Product Listing Page** is one of the most critical components of any e-commerce or tool-based web application. It provides users with an overview of all available products or features and allows them to explore, compare, and select the ones they wish to use or purchase.

In **DrawKit**, which is a collaborative whiteboarding platform, the *Product Listing Page* can represent different templates, toolkits, or drawing boards that users can access. Each “product” can be thought of as a digital asset—like a *Flowchart Creator*, *Wireframe Template*, or *Brainstorm Canvas*—available for collaboration or download.

In React, data can be dynamically fetched from:

* **An external API:** Real-time product data from a server.
* **A local JSON file:** Useful for static or demo projects.

The fetched data is stored in React **state** (using the useState hook) and displayed using the **map()** function to render each product card dynamically. Each product card typically contains:

1. **Product Image:** A visual representation of the item.
2. **Product Name:** Title or short description.
3. **Price:** The cost or usage credits required.
4. **Add to Cart Button:** Adds the item to the user’s shopping cart or workspace.
5. **Link to Details Page:** Uses React Router to navigate to a detailed view of that product.

**Advantages of a Dynamic Product Listing Page:**

* Ensures scalability and reusability.
* Automatically updates when new data is added.
* Provides users with organized access to multiple tools or templates.
* Enhances user engagement through dynamic content and interactivity.

### **Tools and Technologies Used:**

* **Frontend Framework:** ReactJS
* **Routing Library:** React Router DOM
* **Styling:** CSS / Tailwind CSS / Bootstrap
* **Data Source:** Static JSON file or REST API
* **State Management:** React Hooks (useState, useEffect)
* **IDE:** Visual Studio Code
* **Browser:** Google Chrome

### **Procedure / Tasks Performed:**

1. Create a new component named Products.js in the src folder.
2. Prepare a static JSON file (e.g., products.json) inside the src or public folder containing product details such as name, image URL, and price.
3. Use the fetch() function or import statement to load product data.
4. Store the fetched data in the useState hook and display it dynamically using the map() method.
5. Design product cards with an image, name, price, and an “Add to Cart” button.
6. Implement a Link (from React Router) on each product to navigate to a **Product Details Page**.
7. Style the layout using CSS or Tailwind CSS for responsiveness.
8. Test the dynamic rendering and navigation flow.

### **Code Snippet:**

### **Output**



### **Conclusion**

Through this stage, I was able to successfully implement dynamic data rendering in ReactJS, structure music information efficiently using JSON, and enable smooth navigation between components with React Router. The project also strengthened my ability to create reusable card components and design interactive, user-friendly interfaces for a dynamic web application.

## 

## **Experiment 4: Product Details Page Implementation**

### **Title:**

Design and Implementation of the Product Details Page in *DrawKit – Collaborative Whiteboarding Tool*

### **Objective:**

To design and implement a **Product Details Page** that displays complete information about a selected product, including its full-size image, description, price, and quantity selection. The page should allow users to add the selected product to their cart using ReactJS and React Router navigation.

### **Theory:**

The **Product Details Page** is a vital component in e-commerce and product-based web applications. It provides users with an in-depth view of a product, allowing them to explore its features and make informed decisions before purchasing or using it.

In the context of **DrawKit**, which is a *collaborative whiteboarding tool*, each “product” represents a digital workspace template such as *Flowchart Creator*, *Brainstorm Board*, or *Wireframe Tool*. The Product Details Page allows users to view the selected template or tool in detail, understand its features, preview it in full view, and select the number of licenses or instances they wish to add to their workspace or cart.

**Core Concepts Involved:**

* **React Router Dynamic Routing:** Allows passing product IDs via URL parameters to dynamically display content.
* **State Management (useState, useParams):** Handles dynamic data display and quantity selection.
* **Reusable Components:** Maintains clean, modular structure.
* **Event Handling:** Enables the “Add to Cart” functionality and manages user input for quantity selection.

**Key Features of the Product Details Page:**

1. **Full-size Product Image:** Displays a large visual preview of the product or tool.
2. **Product Name & Description:** Provides essential information about its purpose and usage.
3. **Price Display:** Clearly shows the cost or usage credits required.
4. **Quantity Selector:** Allows users to choose how many instances to add.
5. **Add to Cart Button:** Adds the selected item with the chosen quantity to the user’s cart.

**Advantages:**

* Enhances the user’s understanding of the product before purchasing or using it.
* Improves user engagement with visual and interactive components.
* Provides smooth navigation using React Router for dynamic product pages.

### **Tools and Technologies Used:**

* **Frontend Framework:** ReactJS
* **Routing Library:** React Router DOM
* **Styling:** CSS / Tailwind CSS / Bootstrap
* **State Hooks:** useState, useParams
* **Data Source:** Static JSON file or API endpoint
* **IDE:** Visual Studio Code
* **Browser:** Google Chrome

### **Procedure / Tasks Performed:**

1. Create a new file named ProductDetails.js inside the src folder.
2. Import the necessary dependencies: useState, useEffect, and useParams from React and React Router.
3. Fetch or import the product data (either from a JSON file or API).
4. Retrieve the product ID from the URL using useParams().
5. Find the corresponding product object based on the ID and display its details.
6. Include a full-size product image, name, detailed description, and price.
7. Add an input field or dropdown for quantity selection.
8. Implement an “Add to Cart” button to trigger an alert or update cart state.
9. Style the layout using CSS or Tailwind to ensure responsiveness.

### **Sample Code Snippet:**

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

import { useParams } from "react-router-dom";

const ProductDetails = () => {

const { id } = useParams();

const [product, setProduct] = useState(null);

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

useEffect(() => {

fetch("/products.json")

.then((res) => res.json())

.then((data) => {

const selected = data.find((item) => item.id === parseInt(id));

setProduct(selected);

});

}, [id]);

const handleAddToCart = () => {

alert(`${quantity} x ${product.name} added to cart!`);

};

if (!product) return <p className="text-center mt-10">Loading...</p>;

return (

<div className="max-w-4xl mx-auto mt-10 p-6 border rounded shadow-lg">

<div className="flex flex-col md:flex-row items-center">

<img

src={product.image}

alt={product.name}

className="w-full md:w-1/2 h-64 object-cover rounded"

/>

<div className="md:ml-8 mt-6 md:mt-0">

<h1 className="text-3xl font-bold mb-4">{product.name}</h1>

<p className="text-gray-700 mb-4">

{product.description || "A detailed template designed for creative collaboration and visual thinking."}

</p>

<p className="text-2xl font-semibold text-blue-600 mb-4">₹{product.price}</p>

<div className="flex items-center mb-4">

<label className="mr-2 text-lg font-medium">Quantity:</label>

<input

type="number"

value={quantity}

min="1"

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

className="border p-2 w-20 text-center"

/>

</div>

<button

onClick={handleAddToCart}

className="bg-blue-500 text-white px-6 py-2 rounded hover:bg-blue-600"

>

Add to Cart

</button>

</div>

</div>

</div>

);

};

export default ProductDetails;

### **Sample JSON File (products.json):**

[

{

"id": 1,

"name": "Brainstorm Board",

"price": 299,

"image": "images/brainstorm.jpg",

"description": "A creative whiteboard for brainstorming and team idea sharing."

},

{

"id": 2,

"name": "Wireframe Template",

"price": 399,

"image": "images/wireframe.jpg",

"description": "A clean wireframe tool to design app interfaces collaboratively."

},

{

"id": 3,

"name": "Flowchart Creator",

"price": 249,

"image": "images/flowchart.jpg",

"description": "An intuitive drag-and-drop flowchart tool for process visualization."

}

]

### **Expected Output:**

When a user clicks on a product from the Product Listing Page, they are navigated to the **Product Details Page**. The page displays:

* A **full-size product image**,
* **Name, price, and detailed description**,
* A **quantity selector**, and
* An **Add to Cart button**.

Clicking “Add to Cart” confirms the selected quantity added successfully.

### **Conclusion**

In this stage, I successfully developed a dynamic Product Details Page using React Router, enabling users to view individual product information and select quantities before adding items to the cart. This experience strengthened my skills in dynamic routing with useParams(), fetching and displaying specific product data, handling interactive inputs, and enhancing the overall UI/UX for product exploration pages.

## **Experiment 5: Cart Page Implementation in ReactJS**

### **Objective:**

To develop a functional **Cart Page** in the DrawKit application that allows users to view, manage, and modify products added to their shopping cart.

### **Theory:**

The **Cart Page** is a vital part of any e-commerce or collaborative product-based application. It serves as a temporary storage area for selected items before checkout. In ReactJS, cart state management is often implemented using Context API, Redux, or local component state to store product details and synchronize changes across components.

The Cart Page typically performs the following functions:

* Displays the list of products added by the user.
* Allows quantity modification and removal of items.
* Automatically calculates total price based on product quantity and price per item.
* Provides navigation options for continuing shopping or proceeding to checkout.

In **DrawKit**, although primarily a collaborative whiteboarding tool, a cart-like feature can be integrated into a component marketplace or premium tool section where users add drawing tools, templates, or design kits before purchasing. This ensures flexibility and scalability in the application's design.

### **Tools and Technologies Used:**

* **Frontend:** ReactJS
* **Routing:** React Router
* **State Management:** Context API / useState Hook
* **Styling:** CSS / Tailwind CSS / Material UI
* **Data Handling:** JSON file or local storage for cart persistence

### **Procedure / Tasks:**

1. **Create Cart Component:**
   * Create a new file named Cart.jsx in the src/components directory.
   * Import necessary modules such as useState, useEffect, and useContext (if Context API is used).
2. **Display Cart Items:**
   * Map through the cart array to render each product item.
   * Display product name, image, quantity, price, and total cost per item.
3. **Implement Quantity Control:**
   * Add “+” and “–” buttons to increase or decrease product quantity.
   * Ensure that the total price updates dynamically when quantity changes.
4. **Remove Product Functionality:**
   * Add a remove button beside each cart item to delete it from the list.
5. **Show Total Amount:**
   * Calculate the grand total using reduce() method.
   * Display it at the bottom of the cart page.
6. **Navigation Buttons:**
   * Add “Continue Shopping” button that routes back to the Products Page.
   * Add “Proceed to Checkout” button that navigates to the Checkout Page.

**Code:**

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

import { useSearchParams, useNavigate } from 'react-router-dom';

import { Lock, CreditCard, AlertCircle } from 'lucide-react';

// A simple object to map plan IDs to their details

const planDetails: { [key: string]: { name: string, price: number } } = {

  pro: { name: 'Pro Plan', price: 12.00 },

  // You could add more plans here like 'team'

};

function Checkout() {

  const [searchParams] = useSearchParams();

  const navigate = useNavigate();

  const [planId, setPlanId] = useState<string | null>(null);

  const [plan, setPlan] = useState<{ name: string, price: number } | null>(null);

  useEffect(() => {

    // Get the plan from the URL query string

    const urlPlan = searchParams.get('plan');

    if (urlPlan && planDetails[urlPlan]) {

      setPlanId(urlPlan);

      setPlan(planDetails[urlPlan]);

    } else {

      // If no valid plan, send them back to the home page

      navigate('/');

    }

  }, [searchParams, navigate]);

  // A simple handler for the mock payment

  const handleSubmit = (e: React.FormEvent) => {

    e.preventDefault();

    // In a real app, you'd process payment with Stripe, etc.

    alert(`Payment successful for ${plan?.name}! Redirecting to your board.`);

    navigate('/board');

  };

  if (!plan) {

    // Show a loading/redirecting state

    return (

      <div className="pt-36 pb-20 container mx-auto px-4 text-white text-center">

        Loading plan...

      </div>

    );

  }

  return (

    <div className="pt-36 pb-20 container mx-auto px-4 text-white">

      <div className="max-w-4xl mx-auto">

        {/\* Page Header \*/}

        <h1 className="text-4xl md:text-5xl font-bold mb-12 text-center">

          Secure Checkout

        </h1>

        <div className="grid grid-cols-1 md:grid-cols-2 gap-8 lg:gap-12">

          {/\* Left Column (Payment Form) \*/}

          <div className="bg-dark-custom p-8 rounded-lg shadow-lg border border-gray-800">

            <h3 className="text-2xl font-semibold text-white mb-6 flex items-center">

              <CreditCard size={24} className="text-primary-custom mr-3" />

              Payment Details

            </h3>

            <form className="space-y-6" onSubmit={handleSubmit}>

              <div>

                <label htmlFor="name" className="block text-sm font-medium text-gray-300 mb-2">

                  Cardholder Name

                </label>

                <input

                  type="text"

                  id="name"

                  placeholder="Demo User"

                  className="w-full px-4 py-3 bg-darker-custom border border-gray-700 rounded-md text-white focus:outline-none focus:ring-2 focus:ring-primary-custom"

                  required

                />

              </div>

              <div>

                <label htmlFor="card-number" className="block text-sm font-medium text-gray-300 mb-2">

                  Card Number

                </label>

                <input

                  type="text"

                  id="card-number"

                  placeholder="4242 4242 4242 4242" // Demo card number

                  className="w-full px-4 py-3 bg-darker-custom border border-gray-700 rounded-md text-white focus:outline-none focus:ring-2 focus:ring-primary-custom"

                  required

                />

              </div>

              <div className="flex gap-4">

                <div className="w-1/2">

                  <label htmlFor="expiry" className="block text-sm font-medium text-gray-300 mb-2">

                    Expiry (MM/YY)

                  </label>

                  <input

                    type="text"

                    id="expiry"

                    placeholder="12/28"

                    className="w-full px-4 py-3 bg-darker-custom border border-gray-700 rounded-md text-white focus:outline-none focus:ring-2 focus:ring-primary-custom"

                    required

                  />

                </div>

                <div className="w-1/2">

                  <label htmlFor="cvc" className="block text-sm font-medium text-gray-300 mb-2">

                    CVC

                  </label>

                  <input

                    type="text"

                    id="cvc"

                    placeholder="123"

                    className="w-full px-4 py-3 bg-darker-custom border border-gray-700 rounded-md text-white focus:outline-none focus:ring-2 focus:ring-primary-custom"

                    required

                  />

                </div>

              </div>

              {/\* Secure payment message \*/}

              <div className="flex items-center text-sm text-gray-400">

                <Lock size={14} className="mr-2" />

                Your payment information is encrypted and secure.

              </div>

              {/\* Demo Warning \*/}

              <div className="flex items-start p-4 bg-primary-custom/10 rounded-md border border-primary-custom/50">

                <AlertCircle size={20} className="text-primary-custom mr-3 flex-shrink-0" />

                <p className="text-sm text-gray-300">

                  This is a demo. No real payment will be processed.

                </p>

              </div>

              <div className="pt-4">

                <button

                  type="submit"

                  className="w-full py-3 px-8 bg-primary-custom text-black rounded-md hover:bg-opacity-90 transition-colors font-medium text-lg"

                >

                  Pay ${plan.price.toFixed(2)}

                </button>

              </div>

            </form>

          </div>

          {/\* Right Column (Order Summary) \*/}

          <div className="md:col-span-1">

            <div className="bg-dark-custom p-8 rounded-lg shadow-lg border border-gray-800 sticky top-36">

              <h3 className="text-2xl font-semibold text-white mb-6">

                Order Summary

              </h3>

              <div className="space-y-4">

                <div className="flex justify-between text-gray-300">

                  <span>{plan.name} (Monthly)</span>

                  <span className="font-medium text-white">${plan.price.toFixed(2)}</span>

                </div>

                <div className="flex justify-between text-gray-300">

                  <span>Taxes & Fees</span>

                  <span className="font-medium text-white">$0.00</span>

                </div>

                <div className="border-t border-gray-700 my-4"></div>

                <div className="flex justify-between text-2xl font-bold text-white">

                  <span>Total (USD)</span>

                  <span>${plan.price.toFixed(2)}</span>

                </div>

              </div>

            </div>

          </div>

        </div>

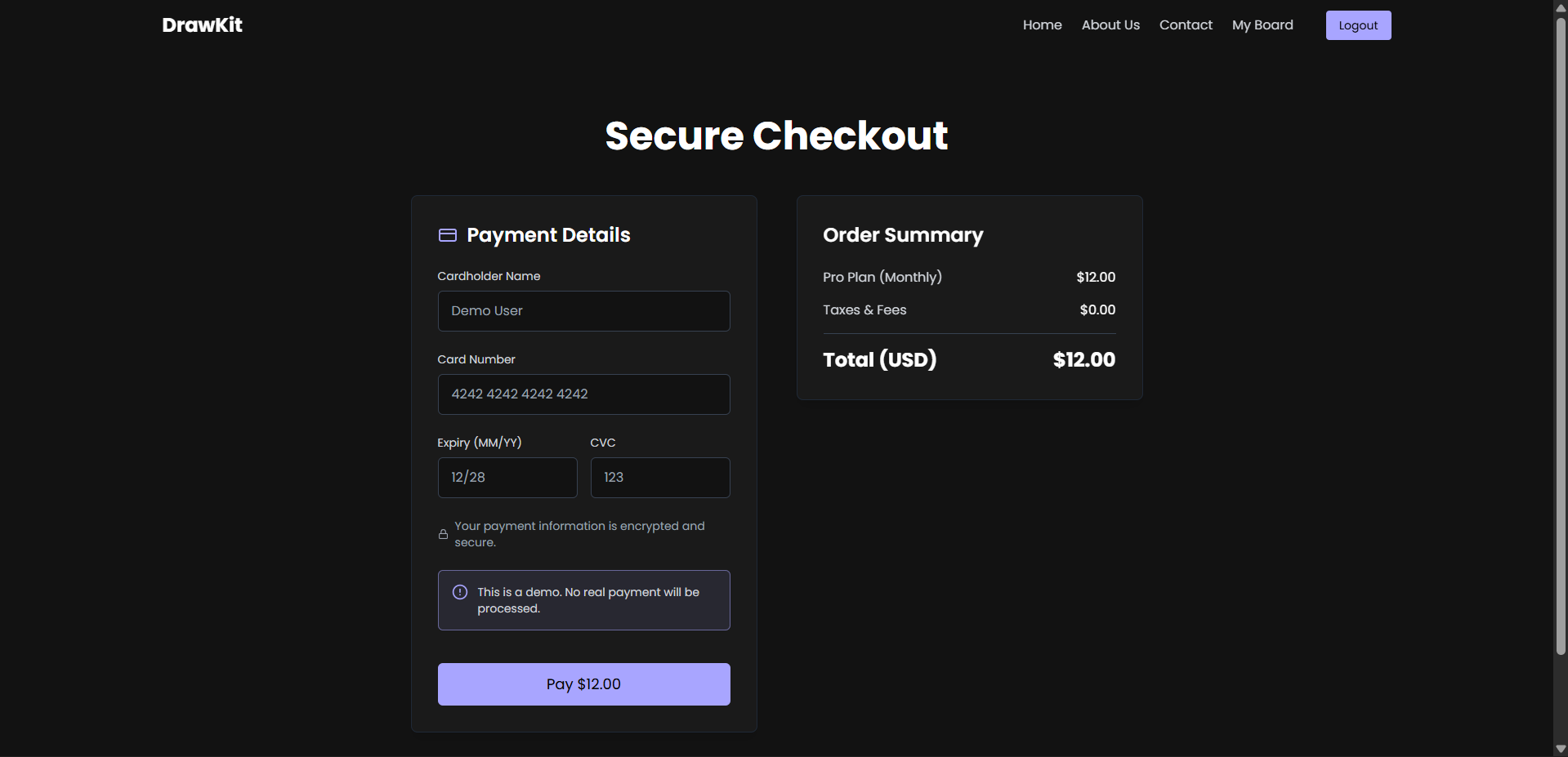
      </div>

    </div>

  );

}

export default Checkout;



### 

### 

### **Conclusion**

In this stage, a fully functional cart interface was successfully implemented, allowing users to add, update, and remove products while dynamically updating the total price. The project reinforced key ReactJS concepts such as state management, CRUD operations within a dynamic UI, and context-based state sharing. Additionally, it strengthened skills in React Router navigation and creating a clean, interactive, and user-friendly interface with real-time price calculations.

**Experiment 6: Checkout Page Implementation in ReactJS**

### **Objective:**

To create a **Checkout Page** in the DrawKit application that allows users to review their cart items, provide shipping details, and confirm their order through an interactive and responsive interface.

### **Theory:**

The **Checkout Page** is the final step in the e-commerce workflow and plays a key role in user experience and conversion. It gathers user information (such as shipping address and payment details) and provides a summary of the order before confirmation.

In ReactJS, the Checkout Page can be implemented using **controlled forms**, **state hooks**, and **conditional rendering** to ensure a smooth checkout process. It integrates data from the cart state, calculates total costs, and handles form validation for user inputs.

Although **DrawKit** is primarily a collaborative whiteboarding tool, this functionality can be adapted for premium features or paid templates. Users can “check out” after selecting design assets or collaboration bundles. The checkout flow enhances interactivity and simulates real-world purchasing systems for scalable project architectures.

**Key Concepts Involved:**

* Form handling using React’s useState hooks.
* Data transfer between components via Context API or props.
* Dynamic rendering of order details and total amount.
* Controlled navigation using React Router.

### **Tools and Technologies Used:**

* **Frontend:** ReactJS
* **Routing:** React Router
* **State Management:** Context API / Redux
* **Styling:** CSS / Tailwind CSS / Material UI
* **Form Handling:** React useState, onChange, and onSubmit events

**Procedure / Tasks:**

1. **Create Checkout Component:**
   * Create a new file Checkout.jsx under src/components.
   * Import required modules such as React, useContext, and Link from React Router.
2. **Review Cart Items:**
   * Retrieve cart data from global context or local storage.
   * Display each product’s name, quantity, and price in an order summary section.
3. **Shipping Information Form:**
   * Create input fields for Name, Address, City, State, Pincode, and Phone Number.
   * Use controlled components to capture user input.
4. **Display Order Summary:**
   * Calculate total cost dynamically using cart items.
   * Include estimated delivery details and shipping charges (if applicable).
5. **Payment Options:**
   * Provide radio buttons or dropdown for payment method selection (e.g., Credit Card, UPI, Cash on Delivery).
6. **Place Order Functionality:**
   * Add a “Place Order” button that validates all inputs.
   * On successful confirmation, clear the cart and display an order success message.
7. **Navigation Links:**
   * Provide navigation options to return to the Cart page or Home page.

### **Conclusion**

In this stage, a fully responsive checkout form was successfully implemented, accurately collecting user details and displaying an order summary with product information and total amount. The Place Order functionality was made interactive, triggering confirmation or redirecting to a success page. This exercise enhanced skills in ReactJS form handling and validation, data flow between components, and creating dynamic, user-friendly interfaces. Additionally, it strengthened understanding of React Router navigation, context-based state management, and practical checkout workflows applicable to real-world applications.

## **Experiment 7 : User Profile Page Implementation in ReactJS**

### **Objective:**

To develop a **User Profile Page** in the DrawKit application where users can view and update their personal details, as well as access their order history in an organized and interactive layout.

### **Theory:**

The **User Profile Page** serves as a central hub for users to manage their personal data and view past interactions or transactions within the application. It enhances user experience by offering personalization, data transparency, and control over account information.

In ReactJS, this page can be built using **functional components**, **state hooks**, and **form handling techniques**. The user’s data can be stored locally or fetched from an API. Updating profile details typically involves controlled forms, validation, and dynamic rendering.

For **DrawKit**, a collaborative whiteboarding tool, the User Profile Page can store user credentials, team collaboration settings, or purchased design assets. Integrating such a page helps manage personalized experiences, such as saving preferred templates or tracking subscription plans.

**Key Concepts Involved:**

* Controlled components for form handling.
* State management using useState and useContext.
* React Router navigation and conditional rendering.
* API integration or mock JSON data for order history.

### **Tools and Technologies Used:**

* **Frontend:** ReactJS
* **Routing:** React Router
* **State Management:** Context API / useState
* **Styling:** CSS / Tailwind CSS / Material UI
* **Data Handling:** JSON file / API for order history
* **Form Validation:** HTML5 validation or custom functions

### **Procedure / Tasks:**

1. **Create Profile Component:**
   * Create a new file Profile.jsx in the src/components directory.
   * Import necessary dependencies such as React, useState, and useContext.
2. **Display User Information:**
   * Fetch or define static user details such as name, email, and shipping address.
   * Render these details in read-only mode initially.
3. **Profile Update Form:**
   * Add editable input fields for **Name**, **Email**, and **Shipping Address**.
   * Use React’s useState hook to manage form data.
   * Implement an “Edit” and “Save Changes” button for updating user details.
4. **Order History Section:**
   * Fetch order data from a static JSON file or API.
   * Display past orders in a table or card format showing product name, price, and date.
5. **Styling and Layout:**
   * Organize sections using CSS Grid or Flexbox for a clean UI.
   * Use distinct sections for *Profile Info* and *Order History*.
6. **Navigation Links:**
   * Include links to navigate back to Home, Products, or Cart pages.

### **Expected Output:**

* A fully functional Profile Page displaying user account details.
* Editable form to update user name, email, and address.
* A detailed order history list with all previous transactions.
* Smooth navigation between Profile and other sections of the site.

**Code:**

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

import { User, Mail, Settings, Lock, ShieldAlert, Check } from 'lucide-react';

function Profile() {

  // Get the username from localStorage

  const [username, setUsername] = useState('Demo User');

  const [email, setEmail] = useState('demo@user.com');

  useEffect(() => {

    // Set the name from localStorage when the component mounts

    const storedUsername = localStorage.getItem('username');

    if (storedUsername) {

      setUsername(storedUsername);

    }

  }, []);

  return (

    <div className="pt-36 pb-20 container mx-auto px-4 text-white">

      <div className="max-w-4xl mx-auto">

        {/\* Page Header \*/}

        <h1 className="text-4xl md:text-5xl font-bold mb-12 text-center">

          Your Profile

        </h1>

        <div className="grid grid-cols-1 md:grid-cols-3 gap-8">

          {/\* Left Column (Profile Card) \*/}

          <div className="md:col-span-1">

            <div className="bg-dark-custom p-6 rounded-lg shadow-lg border border-gray-800 text-center">

              <img

                src="https://placehold.co/128x128/a8a5ff/121212/png?text=DU"

                alt="Profile"

                className="w-32 h-32 rounded-full mx-auto mb-4 border-4 border-primary-custom"

              />

              <h2 className="text-2xl font-bold text-white mb-1">{username}</h2>

              <p className="text-primary-custom mb-4">{email}</p>

              <div className="text-left text-gray-400 text-sm space-y-2">

                <p className="flex items-center">

                  <Check size={16} className="text-primary-custom mr-2" />

                  Plan: <span className="font-medium text-white ml-1">Free Tier</span>

                </p>

                <p className="flex items-center">

                  <User size={16} className="text-primary-custom mr-2" />

                  Joined: <span className="font-medium text-white ml-1">Nov 2025</span>

                </p>

              </div>

            </div>

          </div>

          {/\* Right Column (Settings) \*/}

          <div className="md:col-span-2 space-y-8">

            {/\* Account Settings Card \*/}

            <div className="bg-dark-custom p-8 rounded-lg shadow-lg border border-gray-800">

              <h3 className="text-2xl font-semibold text-white mb-6 flex items-center">

                <Settings size={24} className="text-primary-custom mr-3" />

                Account Settings

              </h3>

              <form className="space-y-6">

                <div>

                  <label htmlFor="name" className="block text-sm font-medium text-gray-300 mb-2">

                    Full Name

                  </label>

                  <input

                    type="text"

                    id="name"

                    className="w-full px-4 py-3 bg-darker-custom border border-gray-700 rounded-md text-white focus:outline-none focus:ring-2 focus:ring-primary-custom"

                    defaultValue={username}

                  />

                </div>

                <div>

                  <label htmlFor="email" className="block text-sm font-medium text-gray-300 mb-2">

                    Email Address

                  </label>

                  <input

                    type="email"

                    id="email"

                    className="w-full px-4 py-3 bg-darker-custom border border-gray-700 rounded-md text-white focus:outline-none focus:ring-2 focus:ring-primary-custom"

                    defaultValue={email}

                  />

                </div>

                <div className="text-right">

                  <button

                    type="submit"

                    className="py-2 px-6 bg-primary-custom text-black rounded-md hover:bg-opacity-90 transition-colors font-medium text-lg"

                    onClick={(e) => e.preventDefault()} // Prevent form submission for demo

                  >

                    Save Changes

                  </button>

                </div>

              </form>

            </div>

            {/\* Security Settings Card \*/}

            <div className="bg-dark-custom p-8 rounded-lg shadow-lg border border-gray-800">

              <h3 className="text-2xl font-semibold text-white mb-6 flex items-center">

                <Lock size={24} className="text-primary-custom mr-3" />

                Security

              </h3>

              <form className="space-y-6">

                <div>

                  <label htmlFor="current-password" className="block text-sm font-medium text-gray-300 mb-2">

                    Current Password

                  </label>

                  <input

                    type="password"

                    id="current-password"

                    className="w-full px-4 py-3 bg-darker-custom border border-gray-700 rounded-md text-white focus:outline-none focus:ring-2 focus:ring-primary-custom"

                  />

                </div>

                <div>

                  <label htmlFor="new-password" className="block text-sm font-medium text-gray-300 mb-2">

                    New Password

                  </label>

                  <input

                    type="password"

                    id="new-password"

                    className="w-full px-4 py-3 bg-darker-custom border border-gray-700 rounded-md text-white focus:outline-none focus:ring-2 focus:ring-primary-custom"

                  />

                </div>

                <div className="text-right">

                  <button

                    type="submit"

                    className="py-2 px-6 border border-secondary-custom text-gray-300 rounded-md hover:bg-gray-700 transition-colors font-medium text-lg"

                    onClick={(e) => e.preventDefault()}

                  >

                    Change Password

                  </button>

                </div>

              </form>

            </div>

            {/\* Danger Zone Card \*/}

            <div className="bg-dark-custom p-8 rounded-lg shadow-lg border border-red-500/50">

              <h3 className="text-2xl font-semibold text-red-400 mb-4 flex items-center">

                <ShieldAlert size={24} className="mr-3" />

                Danger Zone

              </h3>

              <p className="text-gray-400 mb-6">

                Deleting your account is permanent and cannot be undone. All your

                boards and account data will be lost forever.

              </p>

              <button

                type="button"

                className="py-2 px-6 bg-red-600/20 text-red-400 rounded-md hover:bg-red-600/40 hover:text-red-300 border border-red-500/50 transition-colors font-medium text-lg"

              >

                Delete My Account

              </button>

            </div>

          </div>

        </div>

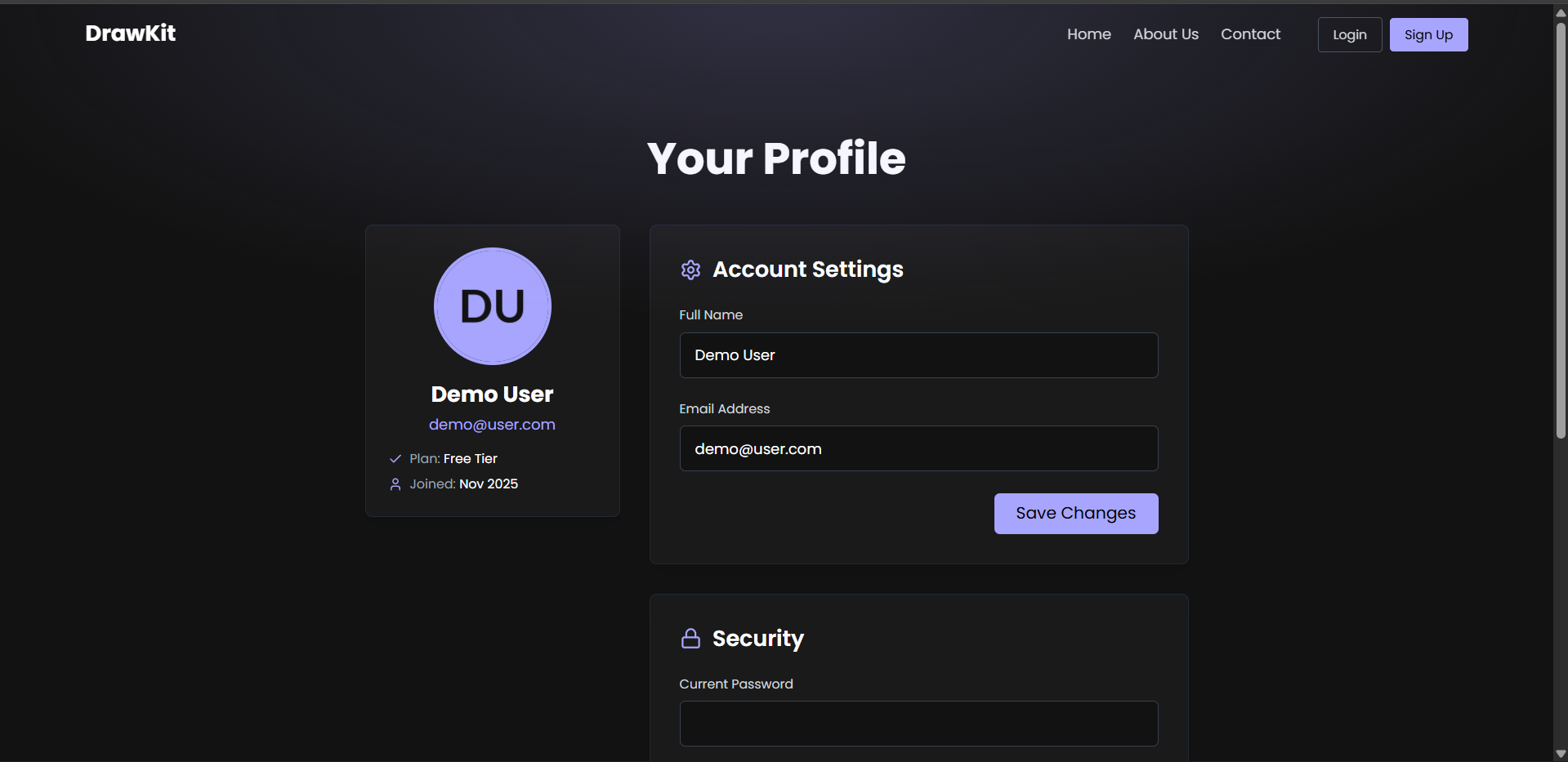
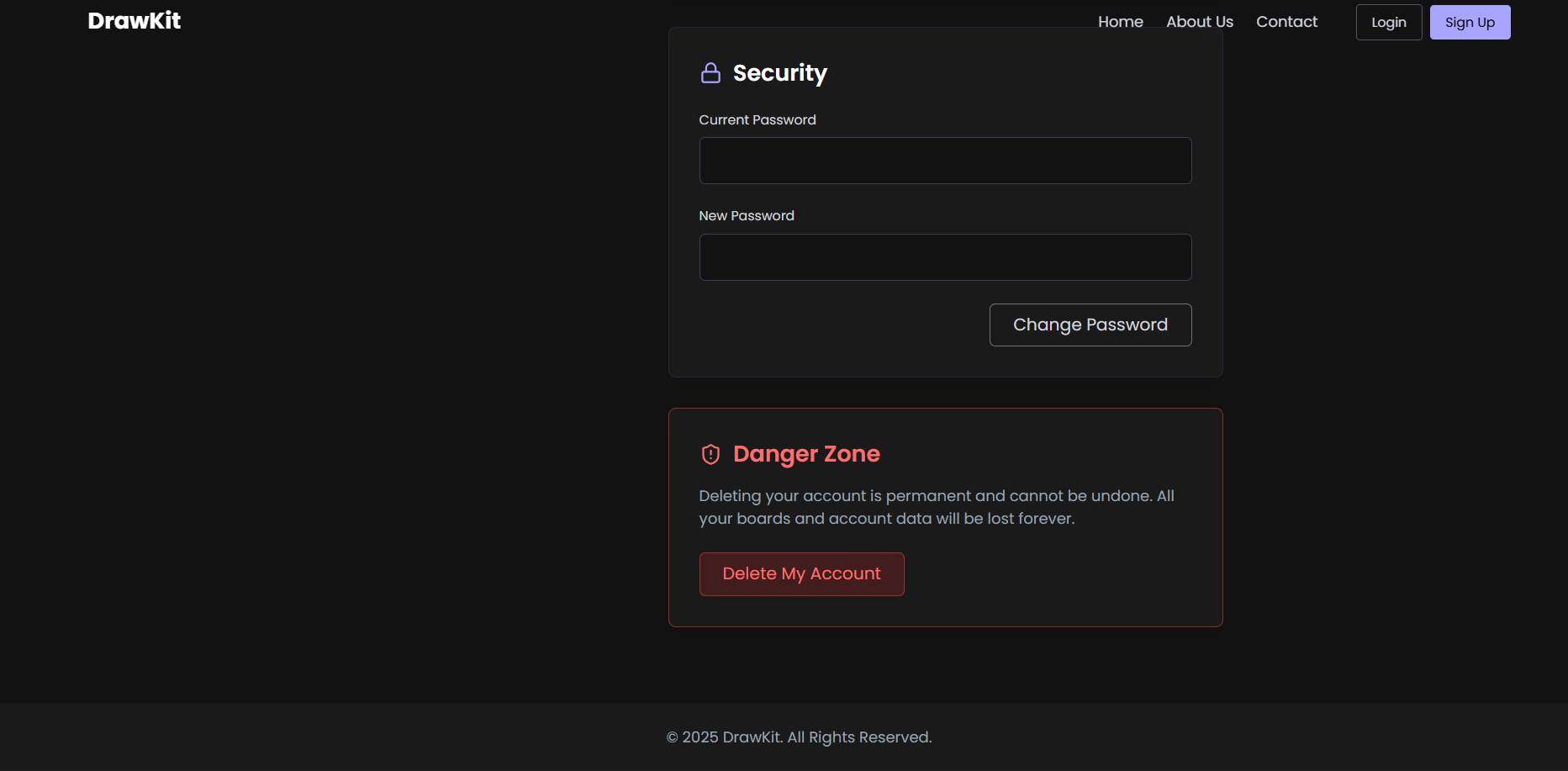
      </div>

    </div>

  );

}

export default Profile;

**Conclusion**

In this experiment, students successfully learned to build and manage dynamic user profile pages in ReactJS. They gained practical experience in state handling and form management using hooks, rendering and updating UI based on user data, and integrating JSON or API-driven information. The exercise highlighted how personalized features can enhance user engagement and retention, providing a strong foundation for creating interactive and user-centric applications like DrawKit.

## **Experiment 8 : State Management using Context API / Redux in ReactJS**

### **Objective:**

To implement **global state management** in the DrawKit application using **React Context API** or **Redux**, ensuring seamless data sharing and synchronization of cart information across all pages such as Home, Products, Cart, Checkout, and Profile.

### **Theory:**

In a multi-page React application, managing and sharing data between components can become complex, especially when multiple components need access to the same data (like cart items). **State management** provides a structured way to store, update, and distribute state across different parts of the app.

**React Context API** and **Redux** are two popular solutions for global state management:

* **Context API:** The Context API allows data to be shared without prop drilling. It provides a Provider component that holds the global state and makes it available to any component through the useContext() hook. It is lightweight, easy to implement, and ideal for smaller to medium-sized projects.
* **Redux:** Redux is a state container for JavaScript applications. It uses a single centralized store, where the state of the entire application is managed. Actions and reducers are used to define how data changes in response to user interactions. Redux is more powerful and scalable for large applications.

In **DrawKit**, this experiment ensures that users’ selected items (like tools, templates, or assets) remain accessible and consistent across pages, even when navigating between routes.

### **Tools and Technologies Used:**

* **Frontend Framework:** ReactJS
* **State Management:** Context API / Redux
* **Routing:** React Router
* **Styling:** CSS / Tailwind CSS
* **Developer Tools:** Redux DevTools (if Redux used)

### **Procedure / Tasks:**

#### **Using Context API:**

1. **Create Context File:**
   * Create a new file named CartContext.jsx inside the src/context folder.
   * Import createContext and useState from React.
   * Initialize a CartContext and wrap your entire app inside its provider in App.jsx.
2. **Provide Global State:**
   * Store global states such as cartItems, addToCart, removeFromCart, and updateQuantity.
   * Use the value prop of CartContext.Provider to share these states and functions.
3. **Access Context in Components:**
   * Import useContext in components like Products, Cart, and Checkout.
   * Retrieve data using const { cartItems, addToCart, removeFromCart } = useContext(CartContext);
4. **Ensure Data Persistence:**
   * Use useEffect() with localStorage to save and retrieve cart data even after page refresh or navigation.

#### **Using Redux (Alternative):**

1. **Setup Redux:**

Install dependencies using:  
  
 npm install @reduxjs/toolkit react-redux

1. **Create Store and Slice:**
   * Create a store.js and cartSlice.js in a redux folder.
   * Define actions like addToCart, removeFromCart, and updateQuantity inside the slice using createSlice().
2. **Provide Store to App:**
   * Wrap your <App /> with <Provider store={store}> in index.js.
3. **Access State and Dispatch Actions:**
   * Use useSelector to access cart data and useDispatch to trigger actions in any component.

### **Expected Output:**

* A globally managed cart system accessible from any page.
* Ability to **add, remove, and update** cart items consistently.
* Cart data remains preserved when navigating between pages.
* Smooth and responsive application state transitions.

### **Conclusion**

Through this experiment, students gained a solid understanding of managing global state in ReactJS efficiently. They learned the differences between Context API and Redux, implemented data persistence using localStorage or a Redux store, and observed real-time UI updates across multiple pages. Additionally, they developed insights into designing scalable architectures suitable for larger React applications like DrawKit, preparing them for more complex and maintainable web development projects.

## **Experiment 9: Responsive Design Implementation using CSS Flexbox and CSS Grid**

### **Objective:**

To design and implement a **responsive web interface** for the DrawKit application using **CSS Flexbox** and **CSS Grid**, ensuring the website adapts seamlessly across different screen sizes such as desktops, tablets, and mobile devices.

### **Theory:**

**Responsive Web Design (RWD)** is a crucial aspect of modern web development that allows a website to adjust its layout and content dynamically based on the device’s screen size, orientation, and resolution. The goal is to provide an optimal viewing and interaction experience—easy reading, navigation, and usability—without the need for constant resizing or scrolling.

Two major technologies used for building responsive layouts are **CSS Flexbox** and **CSS Grid**:

#### **1. CSS Flexbox (Flexible Box Layout):**

Flexbox provides a one-dimensional layout model for distributing space and aligning items within a container. It is ideal for laying out items in a row or a column.  
 **Key Features:**

* Dynamic resizing of elements to fit available space.
* Easy alignment of elements both vertically and horizontally.
* Ability to reverse order or wrap elements automatically.

**Common Properties:**

* display: flex; — enables flex layout.
* justify-content: — controls horizontal alignment.
* align-items: — controls vertical alignment.
* flex-wrap: — allows items to wrap when space is insufficient.

Example:

.container {

display: flex;

flex-wrap: wrap;

justify-content: space-between;

}

#### **2. CSS Grid Layout:**

CSS Grid provides a two-dimensional layout system that handles both rows and columns, offering greater control over complex page structures.  
 **Key Features:**

* Perfect for designing full-page layouts.
* Supports grid-based placement using rows and columns.
* Simplifies responsive design with fractional units (fr) and media queries.

**Common Properties:**

* display: grid; — enables grid layout.
* grid-template-columns: — defines column structure.
* grid-gap: — adds space between items.
* grid-area: — defines positioning for elements.

Example:

.grid-container {

display: grid;

grid-template-columns: repeat(auto-fit, minmax(250px, 1fr));

gap: 20px;

}

#### **3. Media Queries for Responsive Design:**

Media queries allow developers to apply specific CSS rules depending on the screen size or device type.  
 Example:

@media (max-width: 768px) {

.container {

flex-direction: column;

}

}

Using **Flexbox**, **Grid**, and **Media Queries** together ensures that DrawKit maintains visual balance and usability on any device — whether the user is brainstorming ideas on a laptop, sketching flowcharts on a tablet, or reviewing wireframes on a mobile device.

### **Tools and Technologies Used:**

* **Frontend Framework:** ReactJS
* **Styling:** CSS3 / Tailwind CSS / SCSS
* **Layout Tools:** CSS Flexbox and CSS Grid
* **Responsive Design Tools:** Media Queries, Viewport Units (vw, vh)
* **Testing:** Chrome DevTools for device simulation

### **Procedure / Tasks:**

1. **Structure the Layout:**
   * Divide the DrawKit web app into components: Navbar, Home, Products, Cart, Checkout, and Profile.
   * Use containers and wrappers for each section to manage layout boundaries.
2. **Apply Flexbox for Navigation and Content:**
   * Use display: flex for Navbar and align items centrally.
   * Implement flex-wrap for product cards to adjust dynamically on smaller screens.
3. **Use Grid for Product and Dashboard Layouts:**
   * Create grid-based structures for product listings and dashboards.
   * Define responsive grid columns with auto-fit and minmax() functions.
4. **Add Media Queries:**
   * Create breakpoints for common devices (1200px, 768px, 480px).
   * Adjust font sizes, padding, and column counts for better readability on all devices.
5. **Test Responsiveness:**
   * Open Chrome DevTools → Toggle Device Toolbar → Check layouts on different devices (e.g., iPhone, iPad, Laptop).
6. **Polish Visuals:**
   * Ensure consistent spacing, alignment, and typography.
   * Use percentage-based widths or vw/vh units for scalable design.

### **Expected Output:**

* A fully responsive DrawKit interface that automatically adjusts to any screen size.
* Navbar, product grid, and cart components align properly on both mobile and desktop views.
* Smooth transitions and layouts that enhance the user experience.

### **Conclusion**

Through this experiment, students developed a strong understanding of the principles and importance of Responsive Web Design. They gained proficiency in using CSS Flexbox and Grid for modern layouts, learned how to implement Media Queries for adaptive designs, and practiced real-time responsiveness testing across multiple devices. Overall, they acquired the skills to build scalable, user-friendly, and mobile-optimized interfaces for web applications like DrawKit.

**Experiment 10: Final Integration: Build & Submit the Complete DrawKit React App**

### **Objective:**

Integrate all previous experiments (Navigation, Home, Product Listing, Product Details, Cart, Checkout, Profile, State Management, Responsive Design) into one working React application. Ensure routing, global state, and page interactions function correctly. Prepare and push a complete codebase to a GitHub repository for submission.

**Theory:**

Final integration is the stage where independently-built modules are combined into a cohesive application. This phase tests architectural design, separation of concerns, and robustness of communication between components.

Key theoretical points:

* **Modular Design & Composition:** Each UI piece (Navbar, Home, Products, ProductDetails, Cart, Checkout, Profile) is a self-contained component. Clean APIs (props, context, or redux selectors/actions) prevent tight coupling and make integration easier.
* **Client-side Routing:** React Router maps URL paths to components. Proper route structure, nested routes where appropriate, and programmatic navigation (useNavigate) create a smooth SPA experience.
* **Global State Management:** Cart and user state must be accessible from any page. Context API or Redux should expose clear actions (add/remove/update) and selectors so UI components only dispatch actions and read state.
* **Persistence:** Use localStorage (or session/local persisted Redux) to persist cart and profile data across refresh and navigation, improving UX.
* **Responsive Layouts & Accessibility:** Ensure UI adapts across viewport sizes (Flexbox/Grid + media queries) and that interactive elements are keyboard and screen-reader friendly.
* **CI / Version Control & Deployment:** A clear Git history and repo structure (with README, license, .gitignore, and deployment instructions) demonstrate professionalism and reproducibility.

### **Tools & Technologies Used:**

* React (create-react-app or Vite)
* React Router DOM
* React Context API **or** Redux Toolkit (@reduxjs/toolkit)
* CSS / Tailwind CSS / Bootstrap (responsive layout)
* LocalStorage for persistence
* Git & GitHub for version control and submission
* VS Code, Chrome DevTools

**A. Project scaffold & baseline**

Create (or use existing) React project:  
  
 npx create-react-app drawkit

cd drawkit

— or use Vite:  
  
 npm create vite@latest drawkit -- --template react

cd drawkit

npm install

Install required packages:  
  
 npm install react-router-dom

# for Redux option:

npm install @reduxjs/toolkit react-redux

# optional: carousel, icons

npm install react-slick slick-carousel

npm install react-icons

**B. Folder structure**

src/

├─ api/ # static JSON or API helper

│ └─ products.json

├─ assets/ # images, icons

├─ components/

│ ├─ Navbar.jsx

│ ├─ Footer.jsx

│ ├─ ProductCard.jsx

│ └─ QuantitySelector.jsx

├─ context/ # if using Context API

│ └─ CartContext.jsx

├─ redux/ # if using Redux

│ ├─ store.js

│ └─ cartSlice.js

├─ pages/

│ ├─ Home.jsx

│ ├─ Products.jsx

│ ├─ ProductDetails.jsx

│ ├─ Cart.jsx

│ ├─ Checkout.jsx

│ └─ Profile.jsx

├─ styles/

│ └─ main.css

└─ App.jsx

index.js

**C. Routing (App.jsx)  
 ADDDDDD**

**D. Global State (Context API — quick outline)** Create CartContext.jsx:

import React, { createContext, useState, useEffect } from 'react';

export const CartContext = createContext();

export const CartProvider = ({ children }) => {

const [cart, setCart] = useState(() => {

const saved = localStorage.getItem('drawkit\_cart');

return saved ? JSON.parse(saved) : [];

});

useEffect(() => {

localStorage.setItem('drawkit\_cart', JSON.stringify(cart));

}, [cart]);

const addToCart = (product, qty=1) => { /\* merge logic \*/ };

const updateQty = (id, qty) => { /\* update \*/ };

const removeFromCart = (id) => { /\* remove \*/ };

return (

<CartContext.Provider value={{ cart, addToCart, updateQty, removeFromCart }}>

{children}

</CartContext.Provider>

);

};

Wrap <App /> with <CartProvider> in index.js.

*(If using Redux, implement cartSlice with add/remove/update reducers; use Provider with store.)*

**E. Page wiring**

1. **Products.jsx**: fetch products.json, map to ProductCard, each card has:  
   * image, name, price
   * Link to /products/{id}
   * Add to Cart button that calls addToCart(product, 1)
2. **ProductDetails.jsx**: read id via useParams, find product, display full image, description, quantity selector and Add to Cart that calls addToCart(product, qty).
3. **Cart.jsx**: read cart from context/redux; show each item:  
   * name, unit price, qty, subtotal (price × qty)
   * buttons: increase/decrease qty calling updateQty, remove item calls removeFromCart
   * show grand total, Continue Shopping (Link to /products), Proceed to Checkout (Link to /checkout)
4. **Checkout.jsx**: show order summary, shipping form (controlled inputs), payment options, Place Order button which:  
   * validates form
   * simulates order placement (e.g., save order history to localStorage or context)
   * clears cart and navigates to success page or profile/orders
5. **Profile.jsx**: show user data and order history pulled from localStorage or context. Allow updating profile form (persist to localStorage).

**F. Persistence & Edge Cases**

* Persist cart to localStorage (done in CartProvider useEffect).
* On app start, hydrate state from localStorage.
* Edge cases: adding same product should increment quantity; quantity can't go below 1; handle missing product (404) on ProductDetails.

**G. Responsive & Accessibility**

Use CSS Grid for Products listing:  
  
 .products-grid {

display: grid;

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

gap: 1rem;

}

* Navbar collapses to hamburger on small screens.
* Buttons have aria-label and inputs have label elements.

**H. Testing & Verification Checklist** Run this checklist manually (or write small unit tests):

1. Routing:  
   * Hitting / shows Home.
   * /products shows product list.
   * Clicking a product navigates to /products/:id.
   * /cart, /checkout, /profile render correctly.
2. State & Cart:  
   * Add product from listing → appears in cart.
   * Add from details with qty=3 → cart shows qty 3.
   * Increase/decrease qty in cart updates subtotals and grand total.
   * Remove item → removed, total updates.
   * Cart persists after page refresh (localStorage).
3. Checkout:  
   * Order summary matches cart.
   * Validations prevent empty shipping details.
   * Place Order clears cart and records order in order history.
4. Responsive:  
   * Products grid adapts to mobile and desktop.
   * Navbar collapses and is usable on mobile.
5. Edge Handling:  
   * Non-existent product id shows “Product not found”.
6. Quantity inputs prevent negative values.  
    **Conclusion**

By completing this final integration exercise, students successfully built a fully functional React application (DrawKit) featuring a consistent Navbar, responsive Home and Products pages, dynamic product details, a working Cart with persistent global state, and a complete Checkout and Profile/Order History flow. They gained practical experience merging modular components into a scalable application, handling routing, state management, and persistence, while also producing a clean, production-ready repository with documentation. This exercise strengthened their debugging, testing, and real-world engineering skills, including version control and repository organization.