CIT 31300 Final Project Documentation

Lexus Luxury Collection: React Multi-Page Website

Project Overview

The Lexus Luxury Collection is a multi-page React web application designed to showcase Lexus vehicles using modern UI patterns, reusable components, and responsive design. The project implements:

- React Router for smooth, client-side navigation
- Reusable component architecture such as Header, Footer, CarCard, Carousel, Accordion, Hero
- Tailwind CSS + custom CSS for styling and responsive layout
- Dynamic data rendering using props
- Hooks like useState, useEffect, and useCallback to manage interaction

The result is a clean, professional, and highly interactive website that meets all the requirements for the React final project development.

Component Architecture

The project is built around independent, reusable components, which is one of the main strengths of React. Each component focuses on one responsibility, making the app easy to maintain, extend, and test.

Key Components

Header

Displays the gold Lexus-themed banner. Accepts headingText as a prop, making it flexible for all pages.

Footer

A configurable footer component that accepts props such as title, bgColor, and text color.

Hero

Large wallpaper section introducing the brand visually.

Carousel

Auto-rotating Lexus image slider using hooks (useState, useEffect, useCallback). The component takes an array of Lexus car objects as a prop, making it fully reusable.

Accordion

FAQ section using advanced animation + icon rotation.Takes in title and content as props which demonstrate component decoupling.

CarCard

One of the strongest componnets used in this project.

Use:

- useState for opening/closing modal
- Props (car object to dynamically display each model)
- Tailwind styling for a smooth, modern modal UI

Page Components

- Home.js: Combines Hero, Carousel, Accordion

- About.js: Responsive Lexus info layout
- Contact.js: Fully designed form + contact grid
- DisplayCars.js: Dynamic rendering of all car cards
- Layout.js: Navigation + route wrapper

Responsiveness (Mobile-First Design)

The project uses Tailwind CSS, which provides built-in responsive breakpoints (md:, lg:, xl:).

Examples of responsive features:

- Home Page Layout
  - Carousel and accordion stack vertically on mobile (flex-col), side-by-side on medium screens (md:flex-row).
- About Page
  - Two-column grid (md:grid-cols-2) that collapses into a single column on small screens.
- Car Cards and Modal
  - Cards wrap automatically using flex-wrap.
  - Modal centers properly on all screen sizes.
- Navigation
  - Hamburger-style toggle in Layout.js using useState.

This ensures the site works beautifully on phones, tablets, and desktops.

Performant Components (Using Props Correctly)

Across the project, props are used to make components reusable, flexible, and efficient**.** As a result, I don't use hard coding in any components.

Examples:

- Header
  - <Header headingText="About Lexus Collection" />
  - The text is injected using props, allowing one component to serve all pages.
- CarCard
  - <CarCard car={car} />
  - The entire card receives a car object as props, which results in better flexibility and scalability.
- Carousel
  - <Carousel items={CarData} />
  - This allow carousel to display car image and title from the dataset.

- Accordion
  - <Accordion title={item.title} content={item.content} />
  - Minimizes duplication

Hooks Used in the Project

React's useState hook is used throughout the project to manage component-specific state, meaning each component can store and update its own values without affecting the rest of the application. This allows the UI to respond dynamically to user actions.

Examples:

- Controlling Modal Visibility(showModal)
- Used in CarCard.js to show or hide the vehicle information message box.
    - const [showModal, setShowModal] = useState(false);
    - Tracks whether the modal should be displayed. False : modal or hidden,true :modal shown. This creates an interactive, user-driven interface.
- Accordion Open/Close State (isOpen)
    - const [isOpen, setIsOpen] = useState(false);
    - Stores whether each accordion section is expanded.
- Carousel Slide Position (currentIndex)
    - const [currentIndex, setCurrentIndex] = useState(0);
    - This code keeps track of the active slide to show the correct image.
- Navigation Menu Toggle (isOpen) in Layout
    - const [isOpen, setISOpen] = useState(false);
    - this code is to control the Opens and close the mobile menu

Using showModal to Control Interactive Message Box / Modal

In the project, the modal windows used for displaying detailed car information are controlled using the React useState hook. This allows the component to show or hide a message box dynamically based on user interaction.

Example:

In the Carcard.js, a state variable is created:

- const [showModal, setShowModal] = useState(false);
- showModal = false: Modal is hidden
- showModal = true : modal become invisible
- This state determines whether the full vehicle information dialog is rendered on the screen.

Conditional Rendering

The modal only exists in the DOM when showModal is true

Example:

```
{showModal && (
  <div className="modal-overlay" onClick={handleCloseModal}>
    <div className="modal" onClick={(e) => e.stopPropagation()}>
      ...
    </div>
  </div>
)}
```

This demonstrates performance conditional rendering, because React does not create or update the modal unless it is needed.

Summary

The Lexus Luxury Collection is a multi-page React web application designed to showcase Lexus vehicles through a modern, responsive, and component-driven interface. The project demonstrates strong use of React fundamentals, including reusable components, props, routing, conditional rendering, and state management with hooks.

The website includes several key pages such as Home, About, Contact, and Display Cars, each built using shared components such as the Header, Footer, Hero, Carousel, Accordion, and CarCard. Data for both vehicles and FAQs is stored in separate JavaScript modules, allowing the UI to be populated dynamically and efficiently.

Responsiveness is achieved through Tailwind CSS, ensuring layouts adjust smoothly across screen sizes. Interactive features such as the rotating carousel, expandable accordion, and modal pop-ups are powered by hooks like useState, useEffect, and useCallback. These hooks manage component-specific state, automate slide transitions, and improve performance by reducing unnecessary re-renders.

Overall, the project highlights a strong understanding of modern frontend development practices. By combining clean design with reusable components and responsive layouts, the application delivers an elegant Lexus-themed user experience which fulfills the final project requirements.