ROOTED Hair Store
CIT 313 Final Project Documentation
Reactive Perspectives
Course: CIT 31300 - Commercial Website Design
Student: Ruth Makanjuola
Project Title: ROOTED Hair Store

1. Project Overview

ROOTED is a React-based commercial website designed for a professional hair store. The application presents premium hair extensions, hair products, styling tools, and consultation services using modern React development techniques. The project was built to demonstrate the required concepts from CIT 313, including page routing, reusable components, React hooks, responsive design, and effective data presentation.

2. Page Routing Implementation

Page routing was implemented using React Router DOM. The application uses:

- BrowserRouter to wrap the application
- Routes and Route to define individual virtual pages
- Link components for navigation within the Navbar

Each page has its own route, and content is dynamically rendered as React components. The application includes the following virtual pages:

- Home
- Shop
- Services
- About
- Contact

Navigation between pages does not trigger a full page reload, instead a smooth user experience. All pages share a common navigation bar, header, and footer.

3. Components and Component Relationships

The application follows a component-based architecture. Components are organized into Pages and Components.

Core Components:

- Navbar (shared across all pages)
- Header (page-specific content)
- Footer (shared across all pages)
- ProductCard
- Modal
- Carousel
- Accordion
- SectionTitle (additional custom component)

Components are designed to be reusable and performant, with data passed through props. This allows the same component to be reused with different data across the application.

4. Data Handling and Props

Application data is stored in external JavaScript files rather than embedded directly in components. Examples include:

- Product data (products.js)
- FAQ data (faqs.js)
- Testimonial data (testimonials.js)

This data is imported and passed to components using props, that shows separation of concerns and making the components flexible and reusable. Data is then manipulated and displayed using JSX on the appropriate pages.

5. React Hooks Usage

React hooks were used to manage state and user interaction throughout the application.
Examples include:

- useState for managing modal visibility
- useState for handling selected products and services
- useState for form input handling on the Contact page
- useMemo for optimizing derived data where applicable

Hooks is used to generate visible changes in the application, such as opening modals, updating UI content, and responding to user events like button clicks.

6. Modal, Carousel, and Accordion Components

The project includes all required interactive components:

- Modal: Used on the Shop and Services pages to display additional information based on application state.
- Carousel: Used on the Home page to display testimonials in a rotating format.
- Accordion: Used on the Home page to present FAQs in a collapsible, user-friendly layout.

Each of these components responds to user interaction and is controlled through React state rather than static content.

7. Responsive Design and Hamburger Menu

The application is fully responsive and adapts to different screen sizes using Tailwind CSS utility classes.
A hamburger menu is implemented for smaller screen sizes, ensuring that navigation remains accessible and readable on mobile devices. Layouts adjust appropriately between mobile and desktop breakpoints, meeting the responsiveness requirements of the project.

8. Styling Approach

Styling was implemented using Tailwind CSS, with additional custom configuration for brand colors and spacing.
Tailwind utility classes were used to:

- Maintain consistent spacing and typography
- Implement responsive layouts
- Apply a professional olive-green and off-white brand theme

The final application renders professionally and maintains visual consistency across all pages.

9. Challenges and Problem Solving

During development, challenges included:

- Managing image paths correctly between the public and src directories
- Ensuring modals responded correctly to state changes
- Implementing a functional hamburger menu without breaking desktop navigation

These issues were resolved through debugging, restructuring component logic, and refining state management to ensure predictable UI behavior.