Part 1: Navigation Bar with Routing

Aim:

To implement a navigation bar in the **ToyKart ReactJS e-commerce website** that enables smooth routing between pages like **Home**, **Products**, **Cart**, **Checkout**, and **Profile** using **React Router**.

Theory

In modern web development, most websites are built as **Single Page Applications (SPAs)** — meaning all page transitions occur dynamically without reloading the entire browser window. This approach provides a fast, smooth, and app-like experience.
For the **ToyKart** online toy store, React Router plays a vital role in managing navigation between different sections such as the home page, product listings, cart, checkout, and user profile.

React Router is a powerful library in **ReactJS** that enables **client-side routing**, where navigation between pages happens by rendering specific React components rather than fetching entire new HTML pages from the server. This makes the user experience faster and more seamless — which is essential for e-commerce platforms like **ToyKart**, where users often browse multiple pages before making a purchase.

React Router enables **client-side routing**, meaning navigation between pages happens instantly by switching components instead of reloading from the server. This ensures a **faster and smoother user experience** — crucial for e-commerce and interactive web apps like the one we're developing.

The **Navigation Bar (Navbar)** acts as the main interface element that allows users to move between pages such as *Home*, *Products*, *Cart*, *Checkout*, and *Profile*. Using React Router, each of these pages is represented as a **Route**, and the Navbar links correspond to different **paths (URLs)**.

React Router uses components like:

- <BrowserRouter> – Wraps the entire app for routing support.
- <Routes> and <Route> – Define each route (path) and the component to render.
- <Link> or <NavLink> – Create navigation links that change the route without reloading the page.

This routing mechanism helps maintain the application's state, especially the cart or logged-in user details, while navigating between pages.

Advantages of Using React Router in ToyKart:

Seamless User Experience: Instant transitions between pages without page reloads.

Preserved Application State: Shopping cart and user data remain intact during navigation.

Dynamic Routing**:** Routes can change based on user login or role (e.g., guest vs registered user).

Organized Structure**:** Code for each page is modular and easier to maintain.

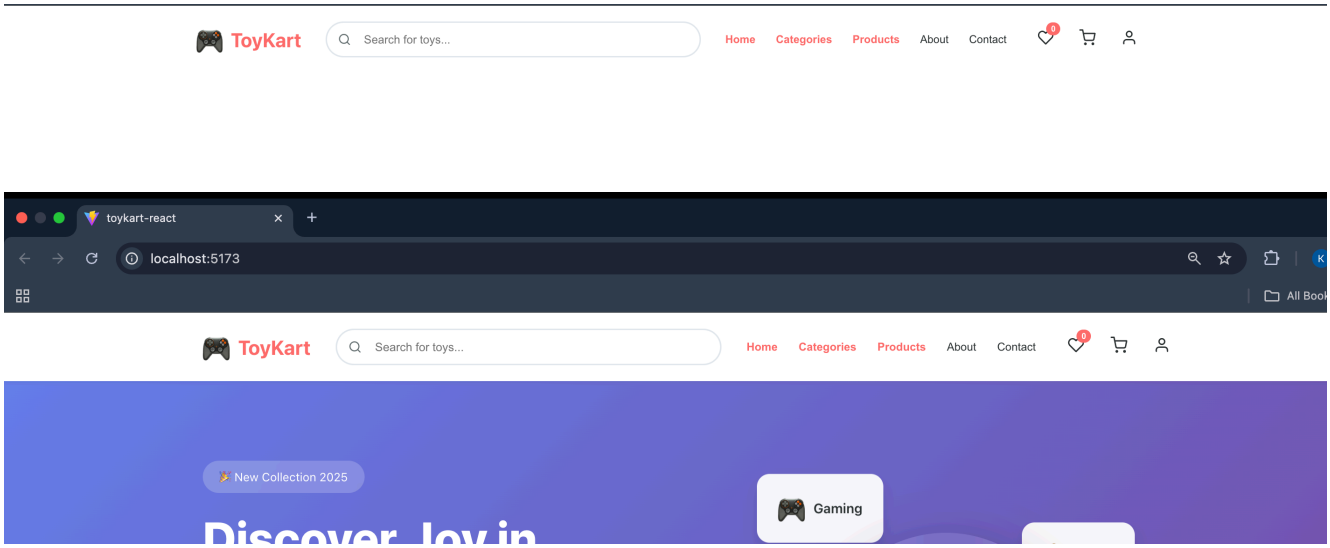
Consistent Design: The Navbar stays fixed at the top of every page, creating a unified shopping experience.

Code:

```
import { FiShoppingCart, FiHeart, FiUser } from 'react-icons/
fi';
import { useCart } from '../../context/CartContext';

const Navigation = ({ currentPage, setCurrentPage }) => {
  const { getCartItemCount } = useCart();
  const cartCount = getCartItemCount();

  const navLinks = [
    { id: 1, name: "Home", page: "home" },
       { id: 2, name: "Categories", page: "home", scrollTo:
"categories" },
        { id: 3, name: "Products", page: "home", scrollTo:
"products" },
    { id: 4, name: "About", page: "about" },
    { id: 5, name: "Contact", page: "contact" }
  ];

  const handleNavClick = (link) => {
    if (link.scrollTo) {
      setCurrentPage('home');
      setTimeout(() => {
                                      const element =
document.getElementById(link.scrollTo);
        if (element) {
          element.scrollIntoView({ behavior: 'smooth' });
        }
      }, 100);
    } else {
      setCurrentPage(link.page);
      window.scrollTo({ top: 0, behavior: 'smooth' });
    }
  };
```

```
  return (
    <>
      <nav className="nav-links">
        {navLinks.map(link => (
          <button
            key={link.id}
            onClick={() => handleNavClick(link)}
            className={`nav-link ${currentPage === link.page ?
'active' : ''}`}
          >
            {link.name}
          </button>
        ))}
      </nav>

      <div className="nav-icons">
        <button className="icon-btn">
          <FiHeart />
          <span className="badge">0</span>
        </button>
        <button
          className="icon-btn"
          onClick={() => {
            setCurrentPage('cart');
            window.scrollTo({ top: 0, behavior: 'smooth' });
          }}
        >
          <FiShoppingCart />
                              {cartCount > 0 && <span
className="badge">{cartCount}</span>}
        </button>
        <button className="icon-btn">
          <FiUser />
        </button>
      </div>
    </>
  );
};

export default Navigation;
```

Output:



Output Preview:

A purple navigation bar with white text links for *Home*, *Products*, *Cart*, *Checkout*, and *Profile*. Clicking a link changes the route instantly without reloading the page.

Conclusion:

The implementation of a **Navigation Bar with React Router** in the **ToyKart website** provides users with an intuitive, smooth, and responsive browsing experience. It enhances the website's usability and design consistency while ensuring that navigation between sections like Home, Products, Cart, Checkout, and Profile happens efficiently without page reloads. This makes the ToyKart application behave like a modern, high-performance e-commerce platform.

Part 2 – Home Page:

Aim:

To design and implement an attractive and responsive **Home Page** for the **ToyKart** e-commerce website using **ReactJS**, showcasing featured toys, promotional banners, and category highlights to engage users and encourage product exploration.

Theory:

The **Home Page** serves as the **entry point** of the **ToyKart** website and plays a crucial role in creating the **first impression** for users. It provides an overview of the store's offerings and acts as a gateway to other key sections such as **Products**, **Cart**, and **Profile**.

In modern e-commerce applications, the home page is designed to be **dynamic and user-centric**, combining **visual appeal** with **functional navigation**. ReactJS helps achieve this by structuring the page into **modular reusable components**, allowing each part — such as banners, toy listings, and category cards — to be developed and managed independently.

The ToyKart Home Page is implemented as a **React functional component** that includes:

- A **Header or Hero Banner** displaying the brand name and tagline (e.g., "Welcome to ToyKart – Where Fun Begins!")
- **Featured Toy Collections** highlighting best-selling or trending toys.
- **Category Sections** for quick browsing (e.g., Cars, Dolls, Educational Toys, Soft Toys).
- A **Call to Action (CTA)** such as "Shop Now" or "View All Products."
- **Footer or Contact Info** for easy access to support and social links.

Working Concept:

When the user visits **ToyKart's Home Page**, ReactJS loads the Home.js component through **React Router**, mapped to the / route.

This page is visually engaging, containing images, text, and buttons that link to other sections like **Products** or **Offers**.

The content is structured using **React components** such as:

- Banner.js – Displays promotional graphics and welcome messages.
- CategoryCard.js – Represents different toy categories with images and titles.

- FeaturedProducts.js – Showcases popular toys from the products dataset.
- Footer.js – Contains brand information and links to contact or social media.

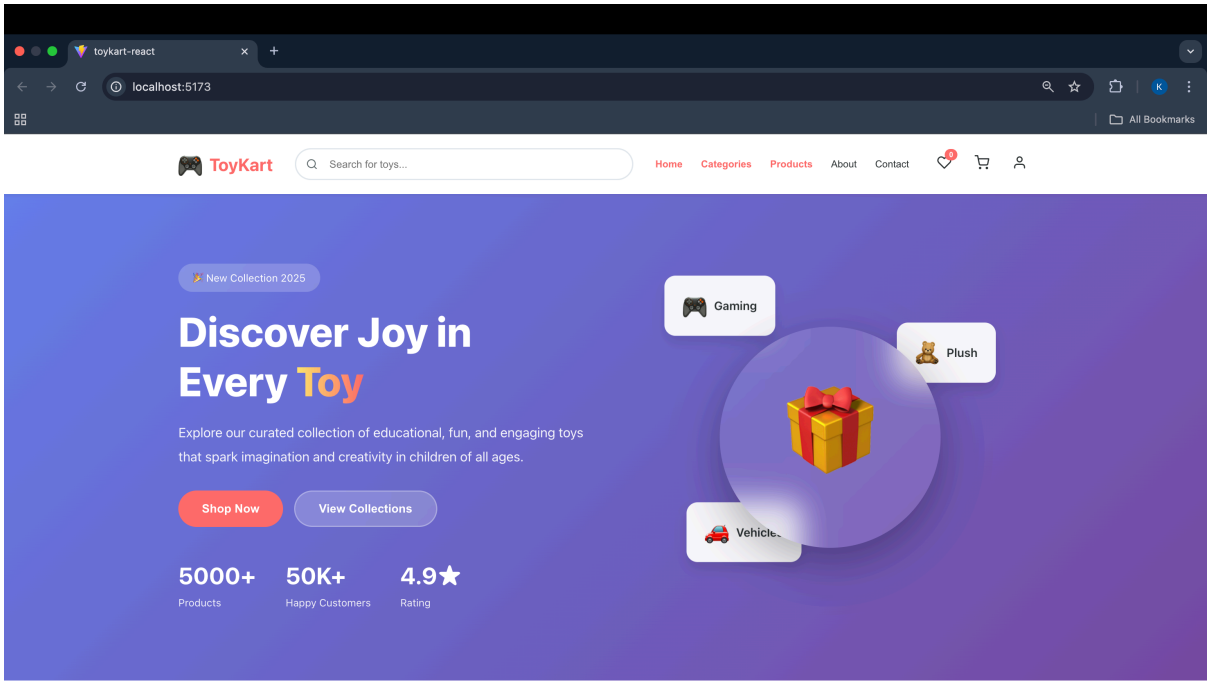
All components are styled using **CSS modules** or **styled-components**, ensuring responsiveness across devices (desktop, tablet, and mobile).

Advantages of a Dynamic Home Page:

- **Attractive Interface:** Captures user attention through bright visuals and interactive design.
- **Improved Navigation:** Direct links guide users to key pages like Products and Offers.
- **Reusable Components:** Modular design enhances maintainability and scalability.
- **Fast Loading:** React efficiently renders only required components.
- **Engagement Boost:** Strategic placement of promotions and offers increases user interaction.

Code:

```
import HeroSection from '../components/Hero/HeroSection';
import FeaturesSection from '../components/Features/
FeaturesSection';
import CategoriesSection from '../components/Categories/
CategoriesSection';
import ProductsSection from '../components/Products/
ProductsSection';
import NewsletterSection from '../components/Newsletter/
NewsletterSection';

const Home = () => {
  return (
    <>
      <HeroSection />
      <FeaturesSection />
      <CategoriesSection />
      <ProductsSection />
      <NewsletterSection />
    </>
  );
};

export default Home;
```



Conclusion:

The **Home Page** of the **ToyKart ReactJS website** acts as the face of the online store, presenting users with a visually appealing and well-structured introduction to the brand. It enhances engagement through featured toys and category displays while maintaining smooth navigation via React Router.By combining aesthetics, responsiveness, and functionality, the ToyKart Home Page ensures a delightful and seamless shopping experience for users.

fetching and user dashboards — all built upon the solid base of this well-structured and interactive React home interface.

Part 3 – Product Listing Page

Aim:

To design and implement a **Product Listing Page** in the Toykart web application that displays a collection of toys with essential details such as name, price , description.. The page should allow users to view available medicines, understand their usage, and interact with the data dynamically using ReactJS state and props.To develop a **Product Listing Page** in the **ToyKart** ReactJS e-commerce website that displays all available toys with options for browsing, filtering, and adding items to the cart using a component-based and dynamic data-driven structure.

Theory :

A The **Product Listing Page (PLP)** is one of the most essential pages in any e-commerce application. It allows users to **explore the range of products** offered by the store and serves as the central hub for shopping activity.

In the **ToyKart** website, the Product Listing Page displays all toy items — such as cars, dolls, puzzles, and soft toys — in a well-organized and visually appealing grid format.

Using **ReactJS**, this page is built using **functional components** that dynamically render products from a **data source (array or API)**. Each product is represented by a **Product Card** component that displays:

- **Product Image**
- **Product Name**
- **Price**
- **“Add to Cart” button**

React’s reusability and state management capabilities make it easy to update the product list instantly when users interact with filters or sorting options — without refreshing the page.

Working Concept:

The Product Listing Page is linked to the /products route in React Router.

When a user clicks “Products” in the navigation bar, the React Router dynamically loads the **Products.js** component.

This page typically includes:

1. **Search Bar:** Allows users to search for specific toys by name or category.
2. **Filter/Sort Options:** Lets users filter toys by price, category, or age group.
3. **Product Grid:** Displays all toy items in a responsive card layout.
4. **Add to Cart Functionality:** Clicking the button triggers a context or state update to add the selected toy to the user's cart.

Each Product Card component (ProductCard.js) receives its details via props from a parent component (Products.js), which maps through the products array and renders cards dynamically.

Advantages of a Dynamic Product Listing Page:

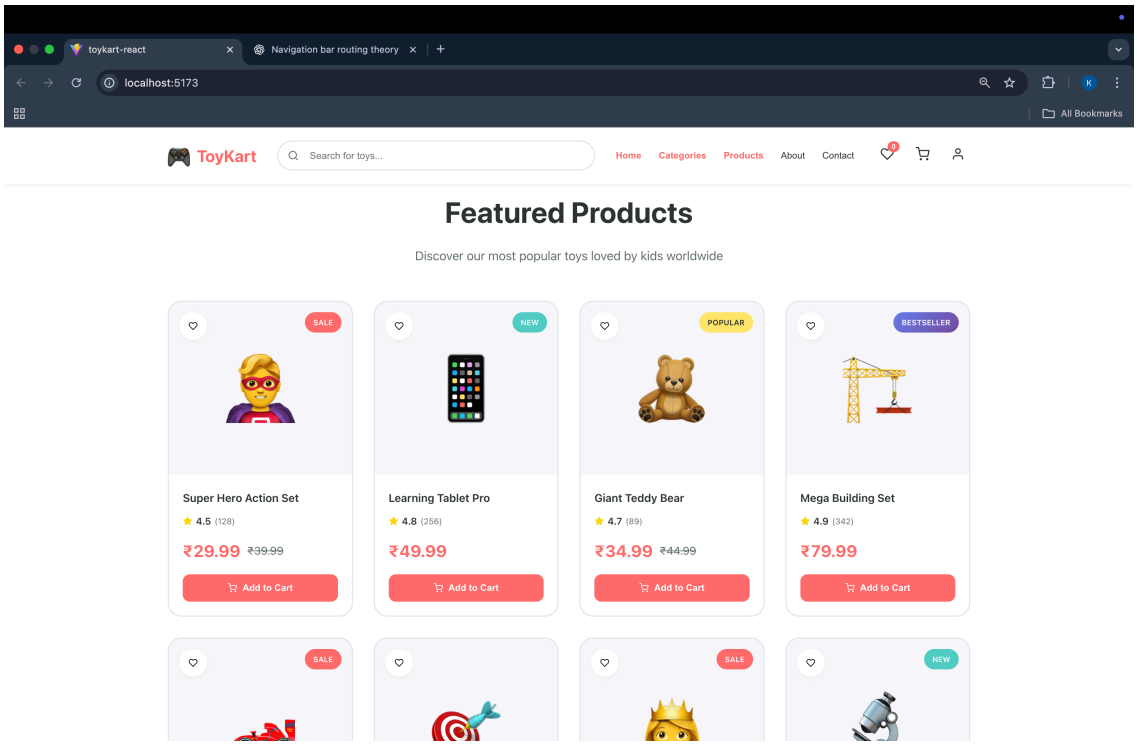
- **Real-Time Rendering:** Instantly updates UI when filters or cart actions are applied.
- **Improved User Experience:** Smooth navigation and responsive product display.
- Maintainability: Components like ProductCard and FilterBar can be reused across the app.
- **Scalability:** Easy to expand product catalog by updating the dataset or connecting to an API.
- **E-Commerce Ready:** Integrated "Add to Cart" and "Buy Now" actions enhance functionality.

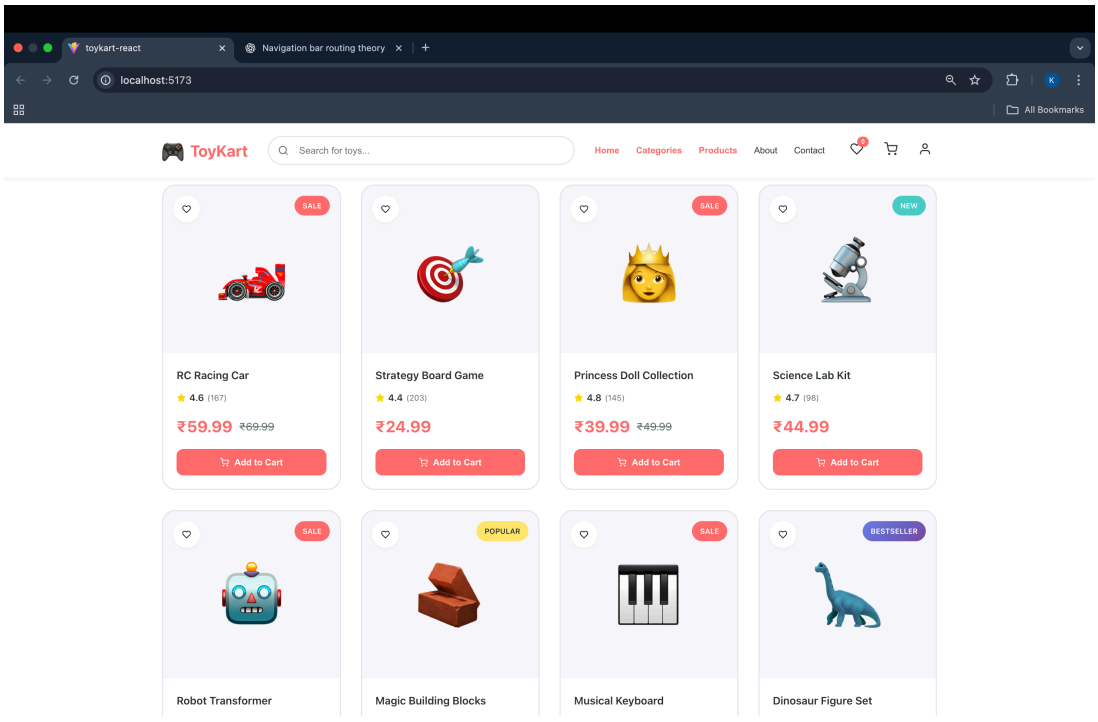
Code:

```
import { FiHeart, FiShoppingCart, FiStar } from 'react-icons/
fi';
import { useCart } from '../../context/CartContext';
import { useState } from 'react';

const ProductCard = ({ product }) => {
   const { name, price, originalPrice, rating, reviews, image,
badge } = product;
  const { addToCart } = useCart();
  const [isAdded, setIsAdded] = useState(false);

  const handleAddToCart = () => {
    addToCart(product);
    setIsAdded(true);

    setTimeout(() => {
      setIsAdded(false);
```

```
    }, 2000);
  };

  return (
    <div className="product-card">
            {badge && <span className={`product-badge badge-${
badge.toLowerCase()}`}>{badge}</span>}

      <div className="product-image">
        <span className="product-emoji">{image}</span>
        <button className="wishlist-btn">
          <FiHeart />
        </button>
      </div>

      <div className="product-info">
        <h3 className="product-name">{name}</h3>

        <div className="product-rating">
          <FiStar className="star-icon filled" />
          <span className="rating-value">{rating}</span>
          <span className="rating-reviews">({reviews})</span>
        </div>

        <div className="product-price">
          <span className="price-current">₹{price}</span>
          {originalPrice && (
              <span className="price-original">₹{originalPrice}
</span>
          )}
        </div>

        <button
            className={`add-to-cart-btn ${isAdded ? 'added' :
''}`}
          onClick={handleAddToCart}
        >
          <FiShoppingCart />
          <span>{isAdded ? 'Added!' : 'Add to Cart'}</span>
        </button>
      </div>
    </div>
  );
};

export default ProductCard;
```

Output:


toykart-react
Navigation bar routing theory
localhost:5173
All Bookmarks
ToyKart
Search for toys...
Home
Categories
Products
About
Contact
Featured Products
Discover our most popular toys loved by kids worldwide
SALE
NEW
POPULAR
BESTSELLER
Super Hero Action Set
4.5 (128)
₹29.99 ₹39.99
Add to Cart
Learning Tablet Pro
4.8 (256)
₹49.99
Add to Cart
Giant Teddy Bear
4.7 (89)
₹34.99 ₹44.99
Add to Cart
Mega Building Set
4.9 (342)
₹79.99
Add to Cart


toykart-react
Navigation bar routing theory
localhost:5173
All Bookmarks
ToyKart
Search for toys...
Home
Categories
Products
About
Contact
SALE
NEW
POPULAR
BESTSELLER
RC Racing Car
4.6 (167)
₹59.99 ₹69.99
Add to Cart
Strategy Board Game
4.4 (203)
₹24.99
Add to Cart
Princess Doll Collection
4.8 (145)
₹39.99 ₹49.99
Add to Cart
Science Lab Kit
4.7 (98)
₹44.99
Add to Cart
Robot Transformer
Magic Building Blocks
Musical Keyboard
Dinosaur Figure Set

Conclusion:

The **Product Listing Page** of the **ToyKart** website serves as the shopping hub where users can browse, compare, and select toys effortlessly.
By leveraging ReactJS concepts such as component reusability, state management, and dynamic rendering, this page ensures a **fast, interactive, and user-friendly shopping experience**.
It also supports smooth integration with the **Cart Page** through React Context or state-based cart management, making it a crucial component of the ToyKart e-commerce system.

Part 4 – Categories Page

Aim:

To design and implement a **Categories Page** in the **ToyKart ReactJS e-commerce website** that displays different toy categories, allowing users to explore products grouped by type, such as Cars, Dolls, Educational Toys, Puzzles, and Soft Toys.

Theory :

In an e-commerce platform like **ToyKart**, the **Categories Page** plays a key role in enhancing the **browsing experience**. It organizes the product catalog into clear sections, helping users quickly find the type of toys they're interested in.
This improves **navigation efficiency**, reduces search time, and increases the likelihood of product discovery and purchase.

Using **ReactJS**, the Categories Page is implemented as a **functional component** that displays a collection of **category cards**. Each card represents a toy category, featuring an image, title, and a button or link that redirects to the **Product Listing Page** filtered by that category.

React's **component-based structure** ensures that each category card is reusable and dynamically generated from a predefined dataset, allowing easy updates or additions to the ToyKart catalog.

Working Concept:

When the user navigates to the **Categories Page** via the navigation bar or homepage, React Router loads the **Categories.js** component linked to the /categories route.

The page structure includes:

1. **Page Title / Header:** "Shop by Categories" — introducing the section.
2. **Category Grid:** A responsive grid layout displaying cards for each toy category.
3. **Category Card Component (CategoryCard.js):**
    - Category Image (representative of the toy type)
    - Category Name (e.g., Cars, Dolls)
    - "Explore" button that routes users to the **Products Page** filtered for that category.

React's <Link> component from **React Router** is used for navigation, ensuring the app remains a **Single Page Application (SPA)** — i.e., navigation occurs without reloading the page.

Core React Concepts Used:

Concept	Description
Components	`Categories.js` (main page) and `CategoryCard.js` (reusable category block).
Props	Pass category data (title, image, link) dynamically to cards.
Mapping	Generate all category cards from a single data array.
Routing	Navigate to `/products` with category filters using React Router.
SPA Architecture	Smooth transitions between categories and product listings.

Advantages of a Categories Page:

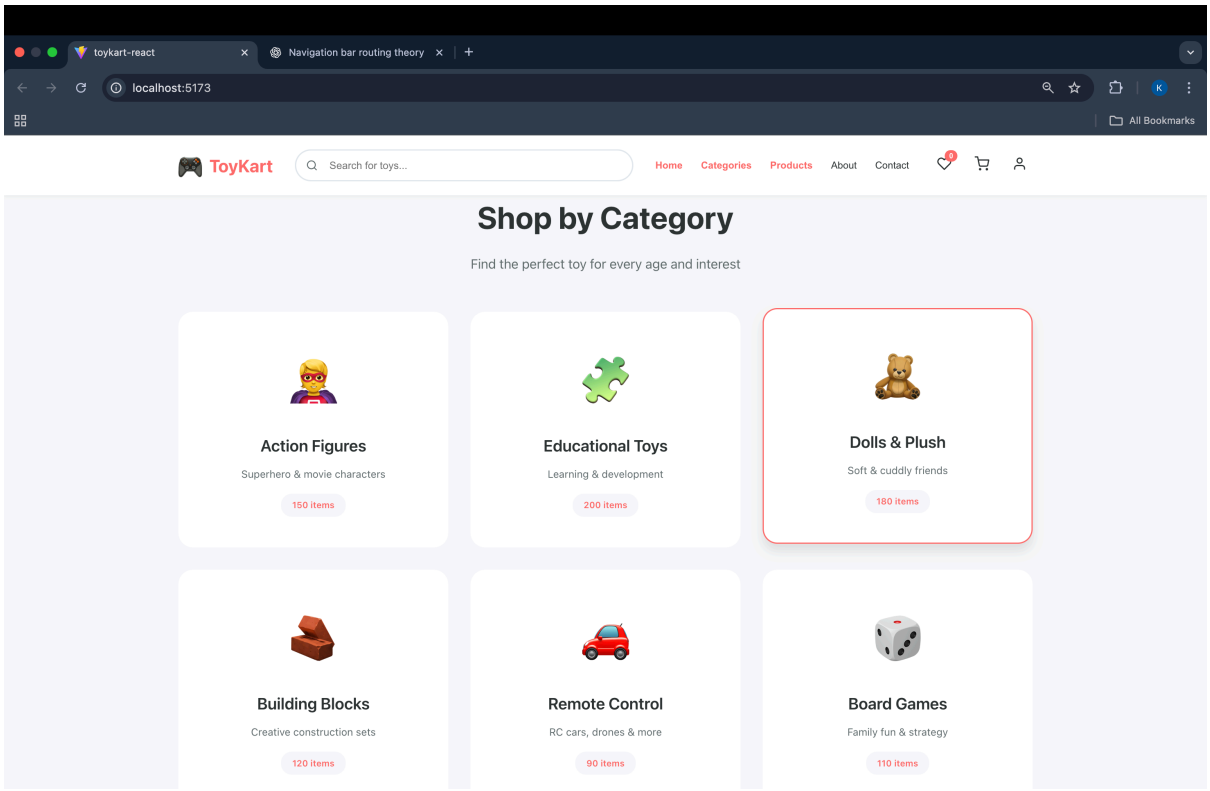
- **Improved Usability:** Users can directly browse by interest instead of scrolling through all products.
- **Visual Appeal:** Attractive category images enhance engagement and brand identity.
- **Reusability:** Category cards can be used in multiple sections like homepage banners or sidebars.
- **Scalability:** New categories can be added easily through data updates.
- **Seamless Navigation:** React Router ensures smooth transitions without reloading pages.

Code:

```
import CategoryCard from './CategoryCard';
import { categories } from '../../data/data';

const CategoriesSection = () => {
  return (
    <section className="categories" id="categories">
      <div className="container">
        <div className="section-header">
          <h2 className="section-title">Shop by Category</h2>
          <p className="section-subtitle">
            Find the perfect toy for every age and interest
          </p>
        </div>
        <div className="categories-grid">
          {categories.map(category => (
            <CategoryCard
              key={category.id}
              icon={category.icon}
              name={category.name}
              description={category.description}
              itemCount={category.itemCount}
```

```
            />
          ))}
        </div>
      </div>
    </section>
  );
};

export default CategoriesSection;

const CategoryCard = ({ icon, name, description, itemCount })
=> {
  return (
    <div className="category-card">
      <div className="category-icon">{icon}</div>
      <h3 className="category-name">{name}</h3>
      <p className="category-description">{description}</p>
      <span className="category-count">{itemCount} items</
span>
    </div>
  );
};

export default CategoryCard;
```

Output:



Conclusion:

The **Categories Page** of the **ToyKart** website provides a structured and visually engaging way for users to explore different toy collections. By utilizing **ReactJS components**, **React Router navigation**, and **dynamic rendering**, this page ensures a clean, efficient, and responsive browsing experience.
It strengthens the user journey by connecting the **Home Page** and **Product Listing Page**, acting as a central hub for category-based product discovery in the ToyKart application.

Part 5 – Cart Page

Aim:

To develop a **Cart Page** in the **ToyKart ReactJS e-commerce website** that displays all the products selected by the user, along with their quantities, prices, and total amount, allowing updates and navigation toward checkout.

Theory:

In any e-commerce application, the Cart Page serves as a vital intermediary between product selection and purchase. It provides users with an overview of all the items they have chosen, along with the ability to modify quantities, remove items, or proceed to checkout.

The ToyKart Cart Page is implemented using ReactJS components that dynamically update based on user interactions.

It leverages React's state management or Context API to maintain the cart's data — ensuring that when users add or remove toys, the changes reflect immediately without reloading the page.

Each cart item is represented through a **CartItem component**, displaying details such as:

- Toy Image
- Toy Name
- Unit Price
- Quantity Selector (+ / –)
- Subtotal
- "Remove" Button

The main Cart Page component (Cart.js) calculates the **total cost**, manages updates to quantities, and displays navigation options such as **"Continue Shopping"** or **"Proceed to Checkout."**

Working Concept:

When the user clicks the **Cart** option in the **ToyKart Navigation Bar**, React Router loads the /cart route and renders the **Cart.js** component.
The Cart Page fetches data from a **shared cart context** or **state**, which stores all items added from the Product Listing Page.

The typical structure includes:instance:

1. **Page Title:** “Your Shopping Cart”
2. **Cart Item List:** Generated dynamically using the map() function to render each item.
3. **Total Summary Section:** Displays subtotal, total price, and checkout button.
4. **Empty Cart Message:** Shown if no items are present.

This page uses **React Hooks** like useContext or useState for managing cart data and ensuring real-time updates.

Concept	Description
State Management	Maintains cart items and quantities.
Context API / Props	Shares cart data across multiple components.
Mapping	Dynamically displays all added cart items.
Conditional Rendering	Displays “Empty Cart” message when applicable.
Routing	Allows smooth navigation to `/checkout` or `/products`.

Advantages of a Dynamic Cart Page:

- **Real-Time Updates:** Instantly reflects changes in quantity or removal of items.
- **Persistent Cart State:** Maintains data across pages using Context API.
- **Interactive Design:** Provides a user-friendly interface for reviewing purchases.
- **Accurate Calculations:** Automatically updates total and subtotal values.
- **Seamless Navigation:** Integrated with Checkout for smooth order placement.

Code:

```
import { FiMinus, FiPlus, FiTrash2 } from 'react-icons/fi';

const CartItem = ({ item, onUpdateQuantity, onRemove }) => {
  return (
    <div className="cart-item">
      <div className="cart-item-image">
        <span className="cart-item-emoji">{item.image}</span>
      </div>

      <div className="cart-item-details">
        <h3 className="cart-item-name">{item.name}</h3>
        <p className="cart-item-category">{item.category}</p>
        <div className="cart-item-price">
          <span className="price">₹{item.price}</span>
          {item.originalPrice && (
                          <span  className="original-price">₹
{item.originalPrice}</span>
          )}
        </div>
      </div>

      <div className="cart-item-quantity">
        <button
          className="quantity-btn"
                    onClick={() => onUpdateQuantity(item.id,
item.quantity - 1)}
          disabled={item.quantity <= 1}
        >
          <FiMinus />
        </button>
            <span  className="quantity-value">{item.quantity}</
span>
        <button
          className="quantity-btn"
                    onClick={() => onUpdateQuantity(item.id,
item.quantity + 1)}
        >
          <FiPlus />
        </button>
      </div>

      <div className="cart-item-total">
                <span  className="total-price">₹{(item.price *
item.quantity).toFixed(2)}</span>
      </div>

      <button
```

```
        className="cart-item-remove"
        onClick={() => onRemove(item.id)}
      >
        <FiTrash2 />
      </button>
    </div>
  );
};

export default CartItem;

import ProductCard from '../Products/ProductCard';

const CartRecommendations = ({ products }) => {
  return (
    <section className="cart-recommendations">
      <div className="container">
           <h2 className="recommendations-title">You May Also
Like</h2>
        <div className="recommendations-grid">
          {products.slice(0, 4).map(product => (
            <ProductCard key={product.id} product={product} />
          ))}
        </div>
      </div>
    </section>
  );
};

export default CartRecommendations;

import { FiShoppingBag, FiArrowRight } from 'react-icons/fi';

const CartSummary = ({ subtotal, shipping, tax, total,
itemCount, onContinueShopping }) => {
  return (
    <div className="cart-summary">
      <h2 className="cart-summary-title">Order Summary</h2>

      <div className="cart-summary-details">
        <div className="summary-row">
          <span>Subtotal ({itemCount} items)</span>
          <span>₹{subtotal.toFixed(2)}</span>
        </div>

        <div className="summary-row">
          <span>Shipping</span>
                    <span>{shipping === 0 ? 'FREE' : `₹$
{shipping.toFixed(2)}`}</span>
```

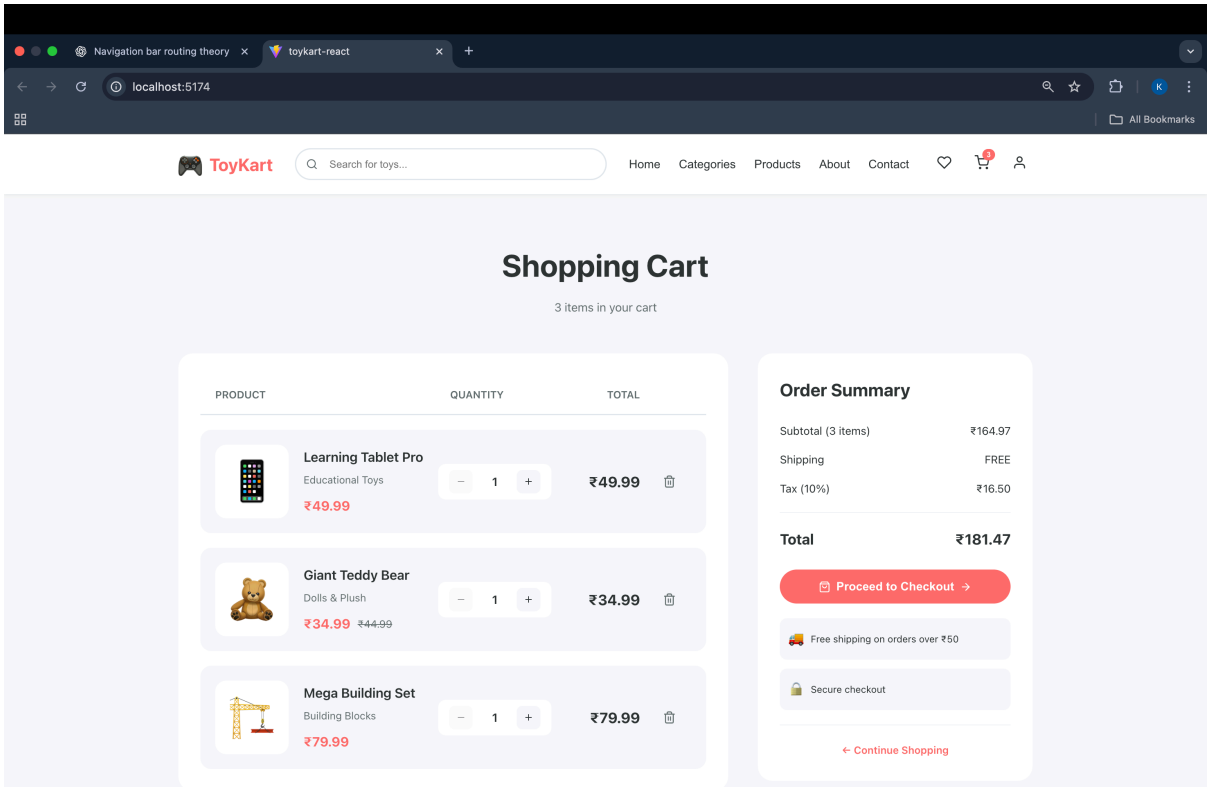
```
        </div>

        <div className="summary-row">
          <span>Tax (10%)</span>
          <span>₹{tax.toFixed(2)}</span>
        </div>

        <div className="summary-divider"></div>

        <div className="summary-row summary-total">
          <span>Total</span>
          <span>₹{total.toFixed(2)}</span>
        </div>
      </div>

      <button className="btn btn-primary checkout-btn">
        <FiShoppingBag />
        <span>Proceed to Checkout</span>
        <FiArrowRight />
      </button>

      <div className="cart-summary-badges">
        <div className="summary-badge">
          <span className="badge-icon">🚚</span>
           <span className="badge-text">Free shipping on orders
over ₹50</span>
        </div>
        <div className="summary-badge">
          <span className="badge-icon">🔒</span>
          <span className="badge-text">Secure checkout</span>
        </div>
      </div>

      <div className="continue-shopping">
             <a href="#" className="continue-link">← Continue
Shopping</a>
      </div>
    </div>
  );
};

export default CartSummary;

import { FiShoppingCart } from 'react-icons/fi';

const EmptyCart = ({ onContinueShopping }) => {
  return (
    <div className="empty-cart">
```

```
      <div className="empty-cart-icon">
        <FiShoppingCart />
      </div>
      <h2 className="empty-cart-title">Your Cart is Empty</h2>
      <p className="empty-cart-description">
          Looks like you haven't added any toys to your cart
yet.
        Start shopping and discover amazing toys!
      </p>
      <button
        className="btn btn-primary"
        onClick={onContinueShopping}
      >
        Start Shopping
      </button>
    </div>
  );
};

export default EmptyCart;

import { createContext, useContext, useState } from 'react';

const CartContext = createContext();

export const useCart = () => {
  const context = useContext(CartContext);
  if (!context) {
          throw new Error('useCart must be used within
CartProvider');
  }
  return context;
};

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

  const addToCart = (product) => {
    setCartItems(prevItems => {
       const existingItem = prevItems.find(item => item.id ===
product.id);

      if (existingItem) {
        // If item exists, increase quantity
        return prevItems.map(item =>
          item.id === product.id
            ? { ...item, quantity: item.quantity + 1 }
            : item
        );
```

```
      } else {
        // If item doesn't exist, add new item
        return [...prevItems, { ...product, quantity: 1 }];
      }
    });
  };

  // Remove item from cart
  const removeFromCart = (productId) => {
    setCartItems(prevItems => prevItems.filter(item => item.id
!== productId));
  };

  // Update item quantity
  const updateQuantity = (productId, newQuantity) => {
    if (newQuantity < 1) return;

    setCartItems(prevItems =>
      prevItems.map(item =>
        item.id === productId
          ? { ...item, quantity: newQuantity }
          : item
      )
    );
  };

  // Clear cart
  const clearCart = () => {
    setCartItems([]);
  };

  // Get cart total
  const getCartTotal = () => {
          return  cartItems.reduce((total,  item)  =>  total  +
(item.price * item.quantity), 0);
  };

  // Get cart item count
  const getCartItemCount = () => {
          return  cartItems.reduce((count,  item)  =>  count  +
item.quantity, 0);
  };

  const value = {
    cartItems,
    addToCart,
    removeFromCart,
    updateQuantity,
    clearCart,
```

```
    getCartTotal,
    getCartItemCount
  };

  return (
    <CartContext.Provider value={value}>
      {children}
    </CartContext.Provider>
  );
};
```

Output:



Conclusion:

The Cart Page of the ToyKart ReactJS e-commerce website functions as a crucial component connecting browsing and purchasing.
By using React components, Context API, and dynamic rendering, it provides a responsive, interactive, and efficient experience for users managing their selected toys.
Its integration with the Checkout Page ensures a complete and smooth shopping flow — from selecting toys to final payment — within a single-page React environment.

Part 6: Contact Page

Aim:

To design and implement a **Contact Page** for the **ToyKart ReactJS e-commerce website** that enables users to easily reach customer support, submit queries, and access store contact information through an interactive and user-friendly interface.

Theory:

In any e-commerce platform, the **Contact Page** plays an important role in establishing **trust, communication, and customer satisfaction**. It allows users to connect with the business for product inquiries, feedback, complaints, or order-related issues.

For the **ToyKart** website, the Contact Page provides a simple and efficient way for users to get in touch with the ToyKart support team.

It typically contains a **contact form**, **company details**, and **social media links**, ensuring customers can communicate through multiple channels.

Using **ReactJS**, the Contact Page is structured as a **functional component** that includes:

- A **Contact Form** for users to submit their name, email, subject, and message.
- **Static Information** like customer service email, phone number, and store address.
- **Social Media Icons or Links** for extended connectivity.
- Optional **Google Maps or location section** to display store location (if applicable).

React handles form state efficiently using hooks like useState, enabling real-time input tracking and form validation.

The logic is handled via React's **useState** and **useEffect** hooks:

- useEffect loads both cart and inventory data from local storage when the component mounts.
- The remaining quantity is calculated for each medicine and displayed alongside its order quantity and total cost.

Working Concept:

When the user navigates to the **Contact Page** via the Navigation Bar, React Router loads the /contact route, rendering the **Contact.js** component.
The component contains:

1. **Page Header:** “Contact ToyKart Support” or “We’d Love to Hear From You!”
2. **Contact Form:**
    - Input fields for name, email, and message.
    - “Submit” button for sending the message (handled using a form event in React).
3. **Contact Details Section:**
    - Customer Support Email (e.g., support@toykart.com)
    - Phone Number (e.g., +91 98765 43210)
    - Working Hours
4. **Social Links:** Icons linking to ToyKart’s Facebook, Instagram, and Twitter pages.

The **onSubmit()** function (in the form) can be used to validate input fields and display a confirmation message using state or toast notifications, creating an interactive experience.

Core React Concepts Used:

Concept	Description
Components	Separate UI blocks for form, contact info, and social links.
useState Hook	Manages user input fields (name, email, message).
Event Handling	Handles form submission and validation.
Routing	Connects `/contact` route to the Contact component via React Router.
Conditional Rendering	Displays success or error messages after form submission.

Advantages of a React Contact Page:

- Enhanced User Interaction: Users can send feedback or queries directly from the site.
- Instant Feedback: React state enables dynamic message confirmation.
- Trust Building: Displaying official contact details increases credibility.
- Reusability: Form components can be reused for newsletter or feedback sections.
- **Responsiveness:** CSS ensures the form adjusts smoothly across all devices

- **Code:**

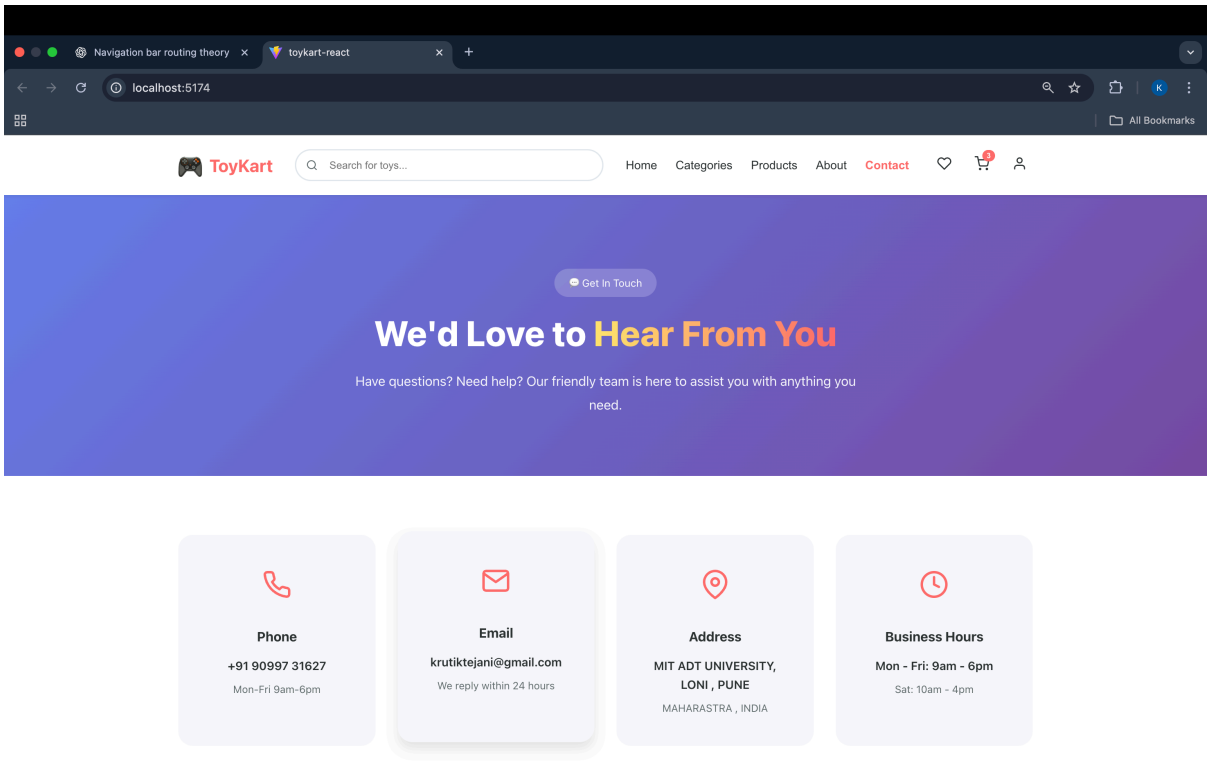
```
import { useState } from 'react';
import { FiSend } from 'react-icons/fi';

const ContactForm = () => {
  const [formData, setFormData] = useState({
    name: '',
    email: '',
    subject: '',
    message: ''
  });

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

  const handleSubmit = (e) => {
    e.preventDefault();
    // Handle form submission
    console.log('Form submitted:', formData);
    alert('Thank you for contacting us! We will get back to you soon.');
    setFormData({ name: '', email: '', subject: '', message: '' });
  };

  return (
    <section className="contact-form-section">
      <div className="container">
        <div className="contact-form-wrapper">
          <div className="contact-form-header">
            <h2 className="section-title">Send Us a Message</h2>
            <p className="section-subtitle">
              Fill out the form below and we'll get back to you as soon as possible
            </p>
          </div>

          <form className="contact-form" onSubmit={handleSubmit}>
            <div className="form-row">
              <div className="form-group">
                <label htmlFor="name">Full Name *</label>
                <input
                  type="text"
                  id="name"
                  name="name"
                  value={formData.name}
                  onChange={handleChange}
                  required
                  placeholder="Krutik Tejani"
                />
              </div>
```

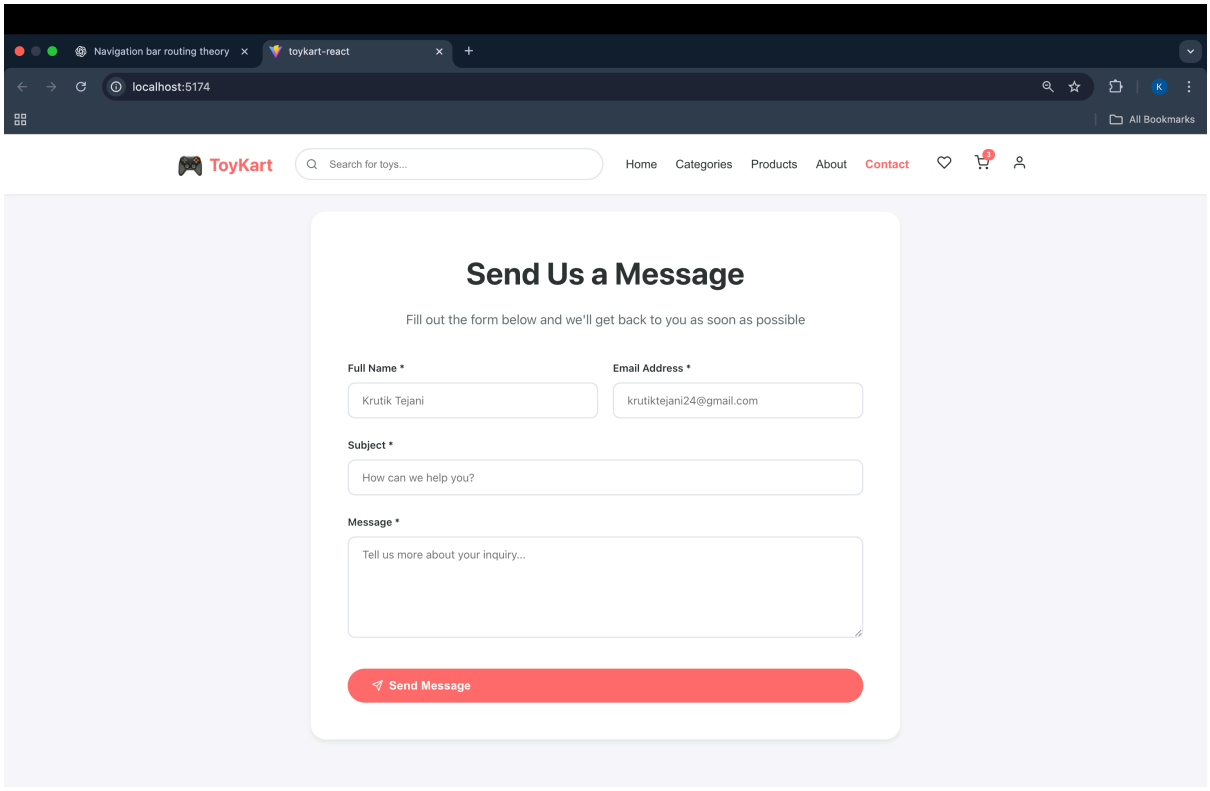
```
            <div className="form-group">
              <label htmlFor="email">Email Address *</label>
              <input
                type="email"
                id="email"
                name="email"
                value={formData.email}
                onChange={handleChange}
                required
                placeholder="krutiktejani24@gmail.com"
              />
            </div>
          </div>

          <div className="form-group">
            <label htmlFor="subject">Subject *</label>
            <input
              type="text"
              id="subject"
              name="subject"
              value={formData.subject}
              onChange={handleChange}
              required
              placeholder="How can we help you?"
            />
          </div>

          <div className="form-group">
            <label htmlFor="message">Message *</label>
            <textarea
              id="message"
              name="message"
              value={formData.message}
              onChange={handleChange}
              required
              rows="6"
              placeholder="Tell us more about your inquiry..."
            />
          </div>

          <button type="submit" className="btn btn-primary btn-submit">
            <FiSend />
            <span>Send Message</span>
          </button>
        </form>
      </div>
    </div>
  </section>
 );
};

export default ContactForm;
```

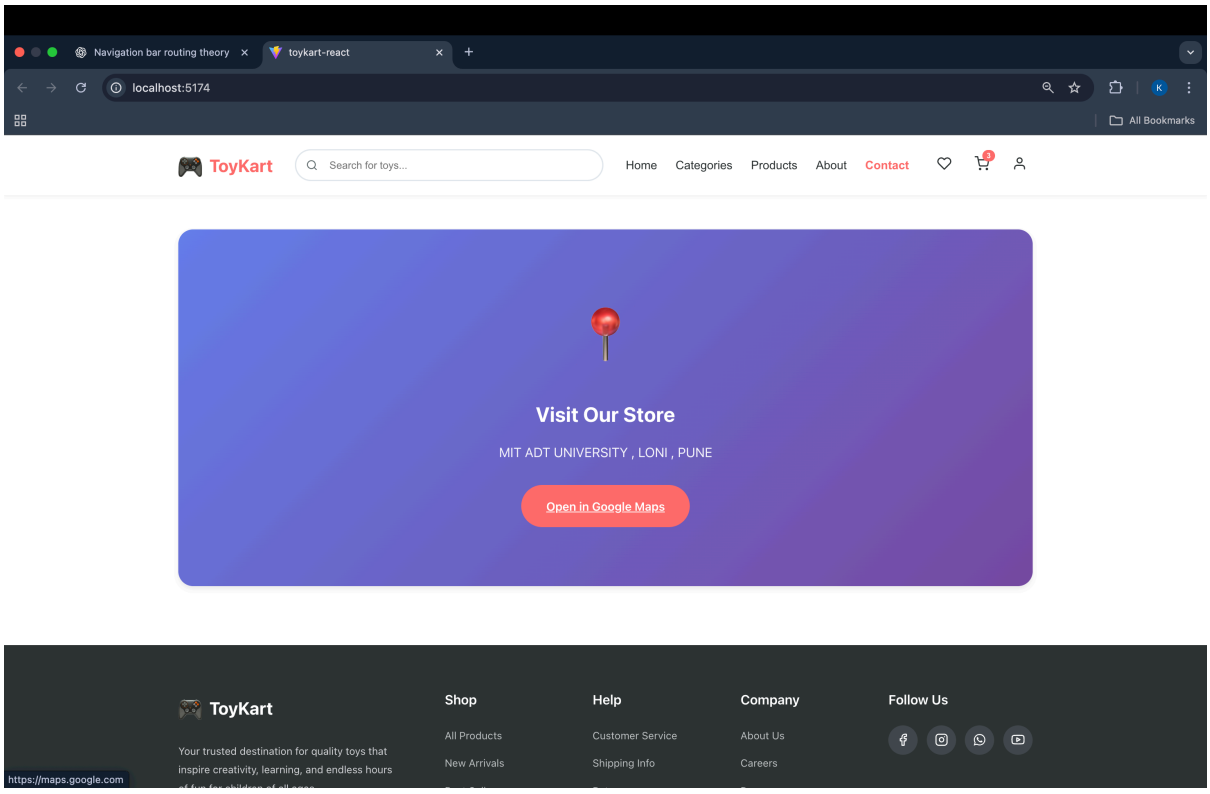
```
const ContactHero = () => {
  return (
    <section className="contact-hero">
      <div className="container">
        <div className="contact-hero-content">
          <span className="contact-badge">💬 Get In Touch</span>
          <h1 className="contact-hero-title">
            We'd Love to <span className="gradient-text">Hear From You</span>
          </h1>
          <p className="contact-hero-description">
            Have questions? Need help? Our friendly team is here to assist you
            with anything you need.
          </p>
        </div>
      </div>
    </section>
  );
};

export default ContactHero;



import { FiPhone, FiMail, FiMapPin, FiClock } from 'react-icons/fi';

const ContactInfo = () => {
  const contactDetails = [
    {
      id: 1,
      icon: <FiPhone />,
      title: "Phone",
      info: "+91 90997 31627",
      subInfo: "Mon-Fri 9am-6pm"
    },
    {
      id: 2,
      icon: <FiMail />,
      title: "Email",
      info: "krutiktejani@gmail.com",
      subInfo: "We reply within 24 hours"
    },
    {
      id: 3,
      icon: <FiMapPin />,
      title: "Address",
      info: "MIT ADT UNIVERSITY, LONI , PUNE",
      subInfo: "MAHARASTRA , INDIA"
    },
    {
      id: 4,
      icon: <FiClock />,
      title: "Business Hours",
      info: "Mon - Fri: 9am - 6pm",
```

```
    subInfo: "Sat: 10am - 4pm"
  }
 ];

 return (
  <section className="contact-info">
   <div className="container">
    <div className="contact-info-grid">
     {contactDetails.map(detail => (
      <div key={detail.id} className="contact-info-card">
       <div className="contact-info-icon">{detail.icon}</div>
       <h3 className="contact-info-title">{detail.title}</h3>
       <p className="contact-info-main">{detail.info}</p>
       <p className="contact-info-sub">{detail.subInfo}</p>
      </div>
     ))}
    </div>
   </div>
  </section>
 );
};

export default ContactInfo;
```

Output:




Navigation bar routing theory
toykart-react
localhost:5174
All Bookmarks
ToyKart
Search for toys...
Home
Categories
Products
About
Contact
Send Us a Message
Fill out the form below and we'll get back to you as soon as possible
Full Name *
Krutik Tejani
Email Address *
krutiktejani24@gmail.com
Subject *
How can we help you?
Message *
Tell us more about your inquiry...
Send Message


Navigation bar routing theory
toykart-react
localhost:5174
All Bookmarks
ToyKart
Search for toys...
Home
Categories
Products
About
Contact
Visit Our Store
MIT ADT UNIVERSITY , LONI , PUNE
Open in Google Maps
ToyKart
Your trusted destination for quality toys that inspire creativity, learning, and endless hours
Shop
All Products
New Arrivals
Help
Customer Service
Shipping Info
Company
About Us
Careers
Follow Us
https://maps.google.com

Conclusion:

The **Contact Page** of the **ToyKart ReactJS e-commerce website** establishes an effective communication bridge between customers and the business.

By using **React components**, **form state management**, and **routing**, it provides an intuitive and responsive interface for submitting queries and viewing company information.
This enhances user trust, improves customer support, and contributes to the overall professionalism and completeness of the ToyKart application.

Part 7: User Profile Page

Aim:

To design and implement a **Login Page** for the **ToyKart ReactJS e-commerce website** that allows registered users to securely sign in, access personalized features such as profile and order history, and maintain session state across the application.

Theory:

In any e-commerce platform, the **Login Page** is essential for **user authentication** and **personalized access**. It ensures that users can securely log into their accounts to view order history, manage personal details, and proceed through a faster checkout process.

For the **ToyKart** website, the Login Page is built using **ReactJS functional components** and leverages concepts such as **state management**, **form validation**, and **conditional rendering** to ensure a smooth and secure user experience.

The page typically includes:

- Input fields for **Email / Username** and **Password**
- A **Login Button** to trigger authentication
- A **"Forgot Password?"** link
- A **"Create Account"** link redirecting new users to the Registration page

By using React's useState hook, the page dynamically captures user input and validates it before submitting. On successful login, users are redirected to their **Profile Page** or **Home Page**, while invalid credentials trigger an error message.

Working Concept:

When a user selects the **Login** option from the ToyKart Navigation Bar, React Router loads the /login route, displaying the **Login.js** component.

The structure typically includes:

1. **Page Title:** "Welcome Back to ToyKart!"
2. **Login Form:**
    - Email / Username field
    - Password field (with show/hide functionality)
    - "Login" button for submitting credentials

3. **Link Section:**
    - "Forgot Password?"
    - "Create an Account" redirect link (/register)
4. **Conditional Messages:**
    - "Invalid Credentials" warning if login fails.
    - "Login Successful" confirmation if credentials match.

Authentication logic can be implemented using:

- **LocalStorage** (for demo or small projects)
- **Backend API Calls** (for real authentication systems)
- **React Context or Redux** (to maintain login state globally across pages)

Core React Concepts Used:

Concept	Description
useState Hook	Captures and manages email and password input fields.
Event Handling	Handles form submission and login logic.
Conditional Rendering	Displays success or error messages dynamically.
Routing	Navigates to `/login`, `/register`, or `/profile` using React Router.
Context / LocalStorage	Stores login session or user token for later access.

Advantages of a React-Based Login Page:

- **Secure Authentication:** Validates credentials before allowing access.
- **Dynamic Validation:** Provides instant feedback without reloading the page.
- **Persistent Sessions:** Maintains login state across pages using context or storage.
- **Seamless Navigation:** Integrates with other routes for profile and checkout.
- **Modern UI Experience:** Responsive, user-friendly, and consistent with ToyKart's design.

Code:

```
import { useState } from 'react';
import { FiMail, FiLock, FiEye, FiEyeOff } from 'react-icons/fi';
import { useAuth } from '../context/AuthContext';

const Login = ({ setCurrentPage }) => {
  const { login } = useAuth();
  const [formData, setFormData] = useState({
    email: '',
    password: ''
  });
  const [showPassword, setShowPassword] = useState(false);
  const [errors, setErrors] = useState({});

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

  const validateForm = () => {
    const newErrors = {};

    if (!formData.email) {
      newErrors.email = 'Email is required';
    } else if (!/\S+@\S+\.\S+/.test(formData.email)) {
      newErrors.email = 'Email is invalid';
    }

    if (!formData.password) {
      newErrors.password = 'Password is required';
    } else if (formData.password.length < 6) {
      newErrors.password = 'Password must be at least 6 characters';
    }

    return newErrors;
  };

  const handleSubmit = (e) => {
    e.preventDefault();

    const newErrors = validateForm();

    if (Object.keys(newErrors).length > 0) {
      setErrors(newErrors);
      return;
    }
```

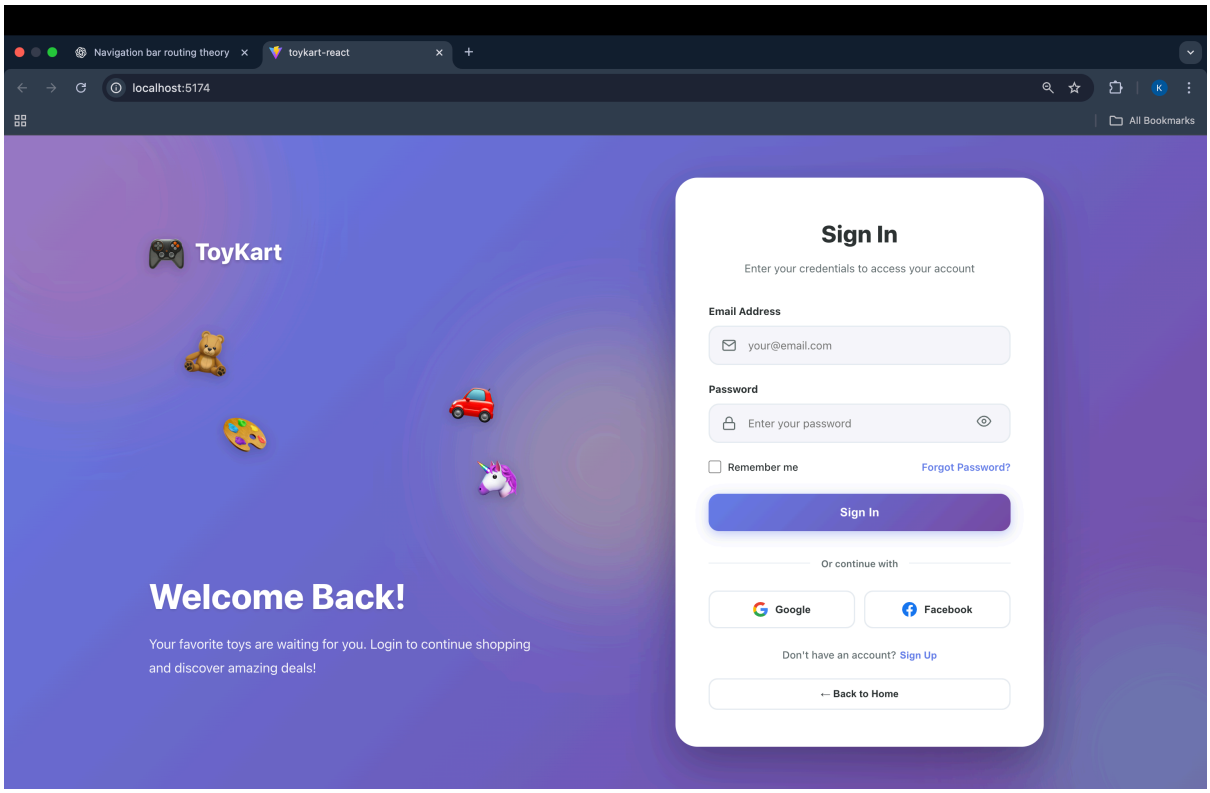
```
  const success = login(formData.email, formData.password);

  if (success) {
    alert('Login successful! Welcome back!');
    setCurrentPage('home');
    window.scrollTo({ top: 0, behavior: 'smooth' });
  }
 };

 return (
  <section className="modern-auth-page">
    <div className="auth-background">
      <div className="auth-shapes">
        <div className="shape shape-1"></div>
        <div className="shape shape-2"></div>
        <div className="shape shape-3"></div>
      </div>
    </div>

    <div className="auth-content-wrapper">
      <div className="auth-left-section">
        <div className="auth-brand">
          <div className="brand-logo">🎮</div>
          <h2 className="brand-name">ToyKart</h2>
        </div>

        <div className="auth-illustration">
          <div className="illustration-content">
            <div className="floating-toy toy-1">🧸</div>
            <div className="floating-toy toy-2">🚗</div>
            <div className="floating-toy toy-3">🎨</div>
            <div className="floating-toy toy-4">🦄</div>
          </div>
          <h1 className="auth-welcome-title">Welcome Back!</h1>
          <p className="auth-welcome-text">
              Your favorite toys are waiting for you. Login to continue shopping and discover amazing
deals!
          </p>
        </div>
      </div>

      <div className="auth-right-section">
        <div className="auth-form-container">
          <div className="form-header">
            <h2 className="form-title">Sign In</h2>
            <p className="form-subtitle">Enter your credentials to access your account</p>
          </div>

          <form className="modern-auth-form" onSubmit={handleSubmit}>
            <div className="modern-form-group">
              <label htmlFor="email">Email Address</label>
```

```
  <div className="modern-input-wrapper">
    <FiMail className="input-icon" />
    <input
      type="email"
      id="email"
      name="email"
      value={formData.email}
      onChange={handleChange}
      placeholder="your@email.com"
      className={errors.email ? 'error' : ''}
    />
  </div>
  {errors.email && <span className="error-text">{errors.email}</span>}
 </div>

 <div className="modern-form-group">
  <label htmlFor="password">Password</label>
  <div className="modern-input-wrapper">
    <FiLock className="input-icon" />
    <input
      type={showPassword ? 'text' : 'password'}
      id="password"
      name="password"
      value={formData.password}
      onChange={handleChange}
      placeholder="Enter your password"
      className={errors.password ? 'error' : ''}
    />
    <button
      type="button"
      className="password-toggle-btn"
      onClick={() => setShowPassword(!showPassword)}
    >
      {showPassword ? <FiEyeOff /> : <FiEye />}
    </button>
  </div>
  {errors.password && <span className="error-text">{errors.password}</span>}
 </div>

 <div className="form-footer-options">
  <label className="remember-checkbox">
    <input type="checkbox" />
    <span>Remember me</span>
  </label>
  <a href="#" className="forgot-password">Forgot Password?</a>
 </div>

 <button type="submit" className="modern-submit-btn">
  Sign In
 </button>
</form>

<div className="auth-divider">
```

```
          <span>Or continue with</span>
        </div>

        <div className="social-auth-buttons">
          <button className="social-auth-btn google-btn">
            <svg width="20" height="20" viewBox="0 0 20 20" fill="none">
                                  <path d="M19.8 10.2273C19.8 9.51821 19.7364 8.83639 19.6182
8.18185H10V12.0491H15.4818C15.2273 13.3 14.4636 14.3591 13.3182
15.0682V17.5773H16.7091C18.6091 15.8364 19.8 13.2727 19.8 10.2273Z" fill="#4285F4"/>
                <path d="M10 20C12.7 20 14.9636 19.1045 16.7091 17.5773L13.3182 15.0682C12.3545
15.6682 11.1273 16.0227 10 16.0227C7.39545 16.0227 5.19091 14.2636 4.35909
11.9H0.859091V14.4909C2.59545 17.9591 6.00909 20 10 20Z" fill="#34A853"/>
                  <path d="M4.35909 11.9C4.14545 11.3 4.02273 10.6591 4.02273 10C4.02273 9.34091
4.14545 8.7 4.35909 8.1V5.50909H0.859091C0.313636 6.59091 0 7.75909 0 10C0 12.2409 0.313636
13.4091 0.859091 14.4909L4.35909 11.9Z" fill="#FBBC05"/>
                  <path d="M10 3.97727C11.2318 3.97727 12.3409 4.38636 13.2091 5.21364L16.2091
2.21364C14.9591 1.04545 12.6955 0 10 0C6.00909 0 2.59545 2.04091 0.859091 5.50909L4.35909
8.1C5.19091 5.73636 7.39545 3.97727 10 3.97727Z" fill="#EA4335"/>
            </svg>
            Google
          </button>
          <button className="social-auth-btn facebook-btn">
            <svg width="20" height="20" viewBox="0 0 20 20" fill="none">
                  <path d="M20 10C20 4.47715 15.5229 0 10 0C4.47715 0 0 4.47715 0 10C0 14.9912
3.65684 19.1283 8.4375 19.8785V12.8906H5.89844V10H8.4375V7.79688C8.4375 5.29063 9.93047
3.90625 12.2146 3.90625C13.3084 3.90625 14.4531 4.10156 14.4531
4.10156V6.5625H13.1922C11.95 6.5625 11.5625 7.3334 11.5625 8.125V10H14.3359L13.8926
12.8906H11.5625V19.8785C16.3432 19.1283 20 14.9912 20 10Z" fill="#1877F2"/>
            </svg>
            Facebook
          </button>
        </div>

        <div className="auth-switch">
          <p>Don't have an account?
            <button
              onClick={() => setCurrentPage('register')}
              className="switch-link"
            >
              Sign Up
            </button>
          </p>
        </div>

        <button
          className="back-home-btn"
          onClick={() => setCurrentPage('home')}
        >
          ← Back to Home
        </button>
      </div>
    </div>
  </div>
```

```
    </section>
  );
};

export default Login;
```

OUTPUT



Conclusion:

The **Login Page** of the **ToyKart ReactJS e-commerce website** provides a secure and efficient way for users to access personalized features and manage their shopping experience.

By combining **React Router**, **form state management**, and **conditional rendering**, it ensures smooth authentication flow and responsive interface design.
This component not only enhances user engagement and trust but also integrates seamlessly with other pages such as **Profile**, **Cart**, and **Checkout**, forming the foundation of ToyKart's user management system.

Part 8: Registration Page

Aim:

To design and implement a **Registration Page** for the **ToyKart ReactJS e-commerce website** that allows new users to create an account by submitting their personal details such as name, email, and password, enabling them to log in and access personalized shopping features.

Theory:

The **Registration Page** is a crucial component of any e-commerce platform as it enables **new users** to join the platform and access features like order tracking, wishlists, and personalized recommendations.
In the **ToyKart** ReactJS website, the Registration Page provides a structured form interface where users can securely register their information, which can later be used for login and account management.

This page is built using **React functional components** and **form handling techniques** with the help of React's **useState hook** to capture user inputs dynamically. It also uses **basic validation logic** to ensure the correctness of details like email format and password strength before account creation.

The main elements of the Registration Page include:

- Input fields for **Full Name**, **Email**, **Password**, and **Confirm Password**
- A **Register Button** to submit details
- A link to redirect existing users to the **Login Page**
- Validation messages for incorrect or incomplete inputs

The component can connect to a **backend API** for user data storage or use **LocalStorage** for demonstration purposes in front-end-only projects.

Working Concept:

When a user clicks the **"Register"** or **"Create Account"** option in the **ToyKart website**, React Router loads the /register route and displays the **Registration.js** component.

The Registration Page layout typically includes:

1. **Page Header:** "Create Your ToyKart Account"
2. **Registration Form Fields:**
    - Full Name
    - Email Address

- Password
- Confirm Password

3. **Buttons and Links:**
   - "Register" button to create an account
   - "Already have an account? Login" link to navigate to /login
4. **Validation Logic:**
   - Ensures all fields are filled
   - Confirms that password and confirm password match
   - Validates email pattern
5. **Confirmation or Error Messages:**
   - Displays success message upon registration
   - Shows relevant error messages for invalid inputs

The form submission is handled by a custom **handleRegister()** function that processes the data and either stores it (locally or via API) or shows an appropriate success alert.

Core React Concepts Used:

Concept	Description
useState Hook	Tracks input values like name, email, and password.
Event Handling	Manages form submission and validation.
Conditional Rendering	Displays validation and success messages dynamically.
Routing	Links `/register` route to the Registration component via React Router.
LocalStorage / API Integration	Stores user data for login functionality.

Advantages of a React-Based Registration Page:

- Real-Time Validation: Users receive instant feedback for incorrect entries.
- Seamless Navigation: React Router allows smooth transition to Login Page.
- Secure Data Handling: Passwords and credentials can be encrypted or validated.
- Reusability: Form components and validation logic can be reused across other forms.
- Enhanced UX: Clean, responsive, and user-friendly design improves sign-up experience

- **Code:**

```
import { useState } from 'react';
import { FiUser, FiMail, FiLock, FiEye, FiEyeOff } from 'react-icons/fi';
import { useAuth } from '../context/AuthContext';

const Register = ({ setCurrentPage }) => {
  const { register } = useAuth();
  const [formData, setFormData] = useState({
    name: '',
    email: '',
    password: '',
    confirmPassword: ''
  });
  const [showPassword, setShowPassword] = useState(false);
  const [showConfirmPassword, setShowConfirmPassword] = useState(false);
  const [errors, setErrors] = useState({});

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

  const validateForm = () => {
    const newErrors = {};

    if (!formData.name) {
      newErrors.name = 'Name is required';
    } else if (formData.name.length < 3) {
      newErrors.name = 'Name must be at least 3 characters';
    }

    if (!formData.email) {
      newErrors.email = 'Email is required';
    } else if (!/\S+@\S+\.\S+/.test(formData.email)) {
      newErrors.email = 'Email is invalid';
    }

    if (!formData.password) {
      newErrors.password = 'Password is required';
    } else if (formData.password.length < 6) {
      newErrors.password = 'Password must be at least 6 characters';
    }

    if (!formData.confirmPassword) {
      newErrors.confirmPassword = 'Please confirm your password';
    } else if (formData.password !== formData.confirmPassword) {
      newErrors.confirmPassword = 'Passwords do not match';
```

```
  }

  return newErrors;
};

const handleSubmit = (e) => {
  e.preventDefault();

  const newErrors = validateForm();

  if (Object.keys(newErrors).length > 0) {
    setErrors(newErrors);
    return;
  }

  const success = register(formData.name, formData.email, formData.password);

  if (success) {
    alert('Registration successful! Welcome to ToyKart!');
    setCurrentPage('home');
    window.scrollTo({ top: 0, behavior: 'smooth' });
  }
};

return (
  <section className="modern-auth-page">
    <div className="auth-background">
      <div className="auth-shapes">
        <div className="shape shape-1"></div>
        <div className="shape shape-2"></div>
        <div className="shape shape-3"></div>
      </div>
    </div>

    <div className="auth-content-wrapper">
      <div className="auth-left-section">
        <div className="auth-brand">
          <div className="brand-logo">🎮</div>
          <h2 className="brand-name">ToyKart</h2>
        </div>

        <div className="auth-illustration">
          <div className="illustration-content">
            <div className="floating-toy toy-1">🎁</div>
            <div className="floating-toy toy-2">🎉</div>
            <div className="floating-toy toy-3">🎈 </div>
            <div className="floating-toy toy-4">⭐</div>
          </div>
          <h1 className="auth-welcome-title">Join ToyKart!</h1>
          <p className="auth-welcome-text">
            Create your account and unlock exclusive deals, special offers, and endless fun!
```

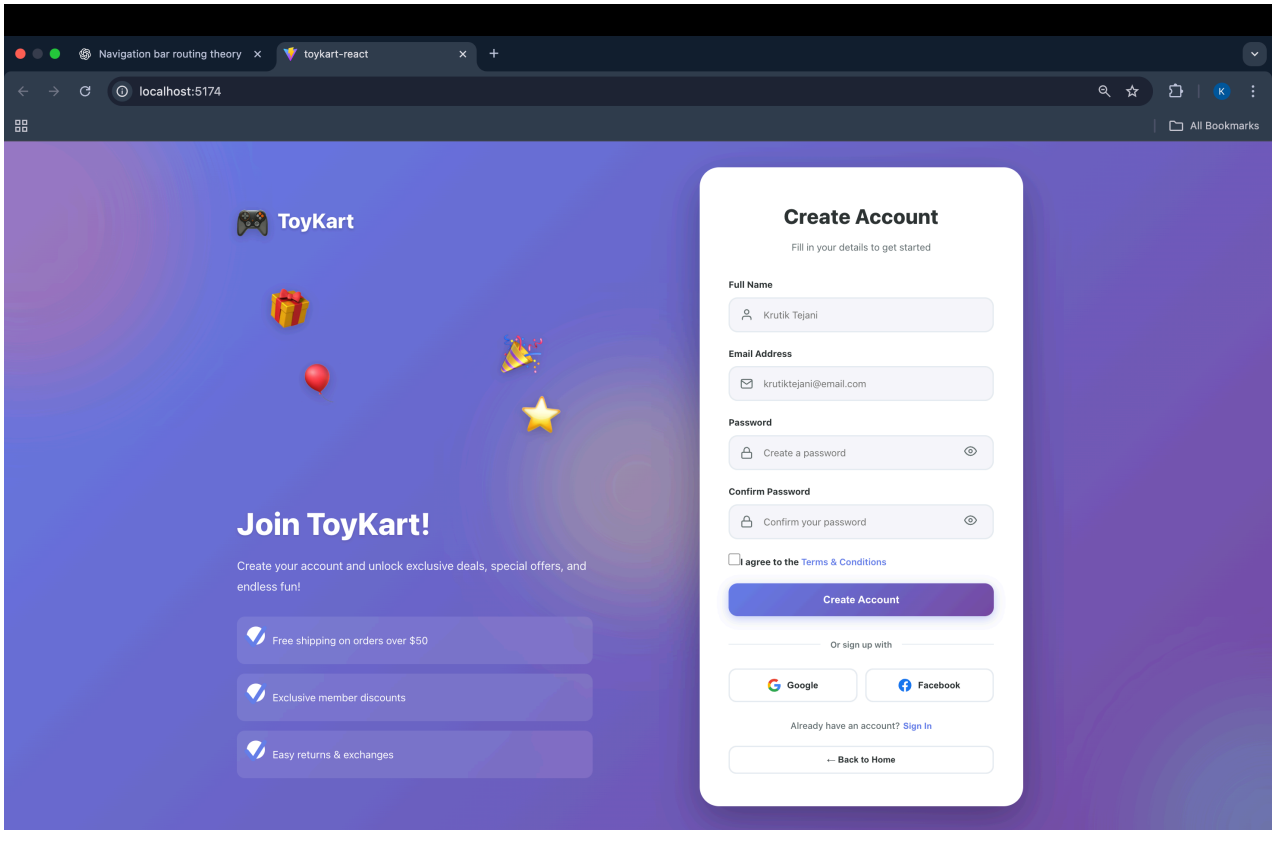
```
    </p>
    <div className="features-list">
     <div className="feature-item">
      <span className="feature-icon">✓</span>
      <span>Free shipping on orders over $50</span>
     </div>
     <div className="feature-item">
      <span className="feature-icon">✓</span>
      <span>Exclusive member discounts</span>
     </div>
     <div className="feature-item">
      <span className="feature-icon">✓</span>
      <span>Easy returns & exchanges</span>
     </div>
    </div>
   </div>
  </div>

  <div className="auth-right-section">
   <div className="auth-form-container">
    <div className="form-header">
     <h2 className="form-title">Create Account</h2>
     <p className="form-subtitle">Fill in your details to get started</p>
    </div>

    <form className="modern-auth-form" onSubmit={handleSubmit}>
     <div className="modern-form-group">
      <label htmlFor="name">Full Name</label>
      <div className="modern-input-wrapper">
       <FiUser className="input-icon" />
       <input
        type="text"
        id="name"
        name="name"
        value={formData.name}
        onChange={handleChange}
        placeholder="Krutik Tejani"
        className={errors.name ? 'error' : ''}
       />
      </div>
      {errors.name && <span className="error-text">{errors.name}</span>}
     </div>

     <div className="modern-form-group">
      <label htmlFor="email">Email Address</label>
      <div className="modern-input-wrapper">
       <FiMail className="input-icon" />
       <input
        type="email"
        id="email"
        name="email"
        value={formData.email}
        onChange={handleChange}
```

```
      placeholder="krutiktejani@email.com"
      className={errors.email ? 'error' : ''}
     />
    </div>
    {errors.email && <span className="error-text">{errors.email}</span>}
  </div>

  <div className="modern-form-group">
    <label htmlFor="password">Password</label>
    <div className="modern-input-wrapper">
      <FiLock className="input-icon" />
      <input
        type={showPassword ? 'text' : 'password'}
        id="password"
        name="password"
        value={formData.password}
        onChange={handleChange}
        placeholder="Create a password"
        className={errors.password ? 'error' : ''}
      />
      <button
        type="button"
        className="password-toggle-btn"
        onClick={() => setShowPassword(!showPassword)}
      >
        {showPassword ? <FiEyeOff /> : <FiEye />}
      </button>
    </div>
    {errors.password && <span className="error-text">{errors.password}</span>}
  </div>

  <div className="modern-form-group">
    <label htmlFor="confirmPassword">Confirm Password</label>
    <div className="modern-input-wrapper">
      <FiLock className="input-icon" />
      <input
        type={showConfirmPassword ? 'text' : 'password'}
        id="confirmPassword"
        name="confirmPassword"
        value={formData.confirmPassword}
        onChange={handleChange}
        placeholder="Confirm your password"
        className={errors.confirmPassword ? 'error' : ''}
      />
      <button
        type="button"
        className="password-toggle-btn"
        onClick={() => setShowConfirmPassword(!showConfirmPassword)}
      >
        {showConfirmPassword ? <FiEyeOff /> : <FiEye />}
      </button>
    </div>
```

```
            {errors.confirmPassword && <span className="error-text">{errors.confirmPassword}</
span>}
          </div>

          <div className="modern-form-group">
            <label className="remember-checkbox">
              <input type="checkbox" required />
              <span>I agree to the <a href="#" className="terms-link">Terms & Conditions</a></span>
            </label>
          </div>

          <button type="submit" className="modern-submit-btn">
            Create Account
          </button>
        </form>

        <div className="auth-divider">
          <span>Or sign up with</span>
        </div>

        <div className="social-auth-buttons">
          <button className="social-auth-btn google-btn">
            <svg width="20" height="20" viewBox="0 0 20 20" fill="none">
                                <path d="M19.8 10.2273C19.8 9.51821 19.7364 8.83639 19.6182
8.18185H10V12.0491H15.4818C15.2273 13.3 14.4636 14.3591 13.3182
15.0682V17.5773H16.7091C18.6091 15.8364 19.8 13.2727 19.8 10.2273Z" fill="#4285F4"/>
              <path d="M10 20C12.7 20 14.9636 19.1045 16.7091 17.5773L13.3182 15.0682C12.3545
15.6682 11.1273 16.0227 10 16.0227C7.39545 16.0227 5.19091 14.2636 4.35909
11.9H0.859091V14.4909C2.59545 17.9591 6.00909 20 10 20Z" fill="#34A853"/>
              <path d="M4.35909 11.9C4.14545 11.3 4.02273 10.6591 4.02273 10C4.02273 9.34091
4.14545 8.7 4.35909 8.1V5.50909H0.859091C0.313636 6.59091 0 7.75909 0 10C0 12.2409 0.313636
13.4091 0.859091 14.4909L4.35909 11.9Z" fill="#FBBC05"/>
              <path d="M10 3.97727C11.2318 3.97727 12.3409 4.38636 13.2091 5.21364L16.2091
2.21364C14.9591 1.04545 12.6955 0 10 0C6.00909 0 2.59545 2.04091 0.859091 5.50909L4.35909
8.1C5.19091 5.73636 7.39545 3.97727 10 3.97727Z" fill="#EA4335"/>
            </svg>
            Google
          </button>
          <button className="social-auth-btn facebook-btn">
            <svg width="20" height="20" viewBox="0 0 20 20" fill="none">
                <path d="M20 10C20 4.47715 15.5229 0 10 0C4.47715 0 0 4.47715 0 10C0 14.9912
3.65684 19.1283 8.4375 19.8785V12.8906H5.89844V10H8.4375V7.79688C8.4375 5.29063 9.93047
3.90625 12.2146 3.90625C13.3084 3.90625 14.4531 4.10156 14.4531
4.10156V6.5625H13.1922C11.95 6.5625 11.5625 7.3334 11.5625 8.125V10H14.3359L13.8926
12.8906H11.5625V19.8785C16.3432 19.1283 20 14.9912 20 10Z" fill="#1877F2"/>
            </svg>
            Facebook
          </button>
        </div>

        <div className="auth-switch">
          <p>Already have an account?
            <button
```

```
            onClick={() => setCurrentPage('login')}
            className="switch-link"
          >
            Sign In
          </button>
        </p>
      </div>

      <button
        className="back-home-btn"
        onClick={() => setCurrentPage('home')}
      >
        ← Back to Home
      </button>
    </div>
  </div>
 </div>
</section>
);
};

export default Register;
```

Output:



Conclusion:

The **Registration Page** of the **ToyKart ReactJS e-commerce website** is a key component that facilitates user onboarding by allowing new users to create their accounts easily and securely.

Built using **React components**, **form handling**, and **routing**, it ensures a dynamic and responsive interface with real-time validation.
By connecting this page with the **Login** and **Profile** sections, ToyKart achieves a complete and user-focused authentication system that enhances accessibility and user trust across the platform.

Part 9 – About page

Aim:

To design and implement an **About Page** for the **ToyKart ReactJS e-commerce website** that provides users with essential information about the brand, its mission, vision, and the purpose behind the platform, fostering trust and transparency among customers.

Theory:

The **About Page** is an integral part of any professional website as it introduces the brand to its audience. It helps users understand **who runs the business**, **what values it upholds**, and **how it aims to serve customers**. In an e-commerce platform like **ToyKart**, the About Page adds authenticity and builds a personal connection with users.

In ReactJS, the About Page is created as a **functional component** that contains **static and dynamic content** such as brand history, mission statement, achievements, and team information. It is rendered when users navigate to the /about route via the navigation bar using **React Router**.

This page typically includes well-structured sections with appealing UI elements such as text blocks, icons, and images. The layout can be designed using **CSS Flexbox** or **Grid** to ensure responsiveness across devices.

Working Concept:

When the user clicks on the **"About"** link in the ToyKart website navigation bar, React Router loads the /about route and displays the **About.js** component.

The page layout generally includes the following sections:

1. **Header Section:**
   Displays the title — *"About ToyKart"* or *"Our Story"*.

2. **Introduction:**
   Briefly explains ToyKart's origin, idea, and motivation behind building the platform.

3. **Mission and Vision Statements:**

   - **Mission:** To deliver joy and creativity through a wide variety of toys accessible to everyone.

   - **Vision:** To become a leading and trusted online toy destination promoting learning and fun.

4. **Core Values Section:**
   Highlights brand values such as quality, safety, affordability, and customer satisfaction.

5. **Team and Brand Story (Optional):**
   Introduces founders or the development journey of ToyKart.

6. **Customer Commitment Message:**
   Reassures users about secure shopping, fast delivery, and reliable customer service.

Core React Concepts Used:

Concept	Description
Functional Components	The About Page is created as a reusable React component.
React Router (`<Route>`)	Manages page navigation without reloading.
Static Rendering	Displays static text and brand content.
Responsive Design	Uses CSS for a clean layout across devices.
Reusability	Navbar and Footer components are reused on this page.

Advantages of Having an About Page:

- **Builds Trust:** Helps customers know the brand better.
- **Brand Identity:** Communicates ToyKart's mission and values.
- **Professional Appearance:** Enhances credibility and completeness of the site.
- **SEO Benefits:** Improves website discoverability through descriptive content.
- **Customer Connection:** Establishes emotional engagement with users.

Code:

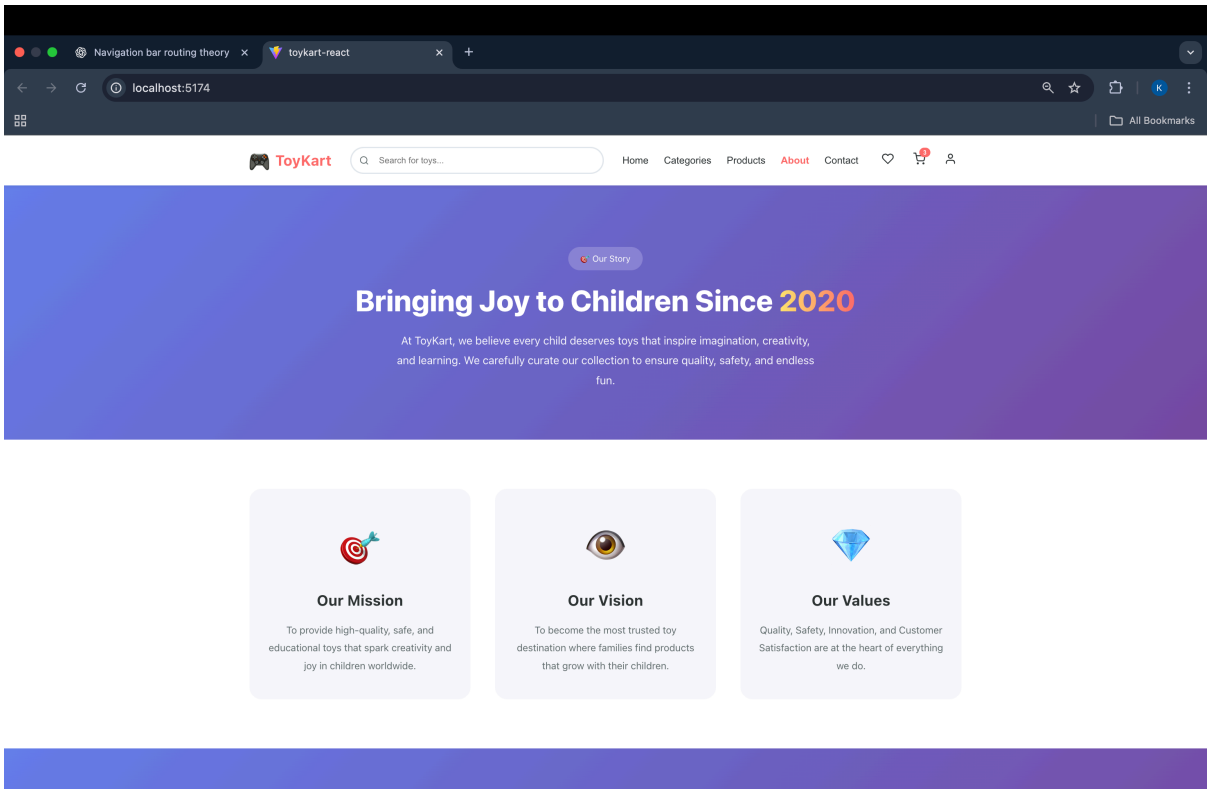
```
const AboutHero = () => {
 return (
  <section className="about-hero">
   <div className="container">
    <div className="about-hero-content">
     <span className="about-badge">🎯 Our Story</span>
     <h1 className="about-hero-title">
      Bringing Joy to Children Since <span className="gradient-text">2020</span>
     </h1>
     <p className="about-hero-description">
      At ToyKart, we believe every child deserves toys that inspire imagination,
      creativity, and learning. We carefully curate our collection to ensure
      quality, safety, and endless fun.
     </p>
    </div>
   </div>
  </section>
 );
};

export default AboutHero ;
```

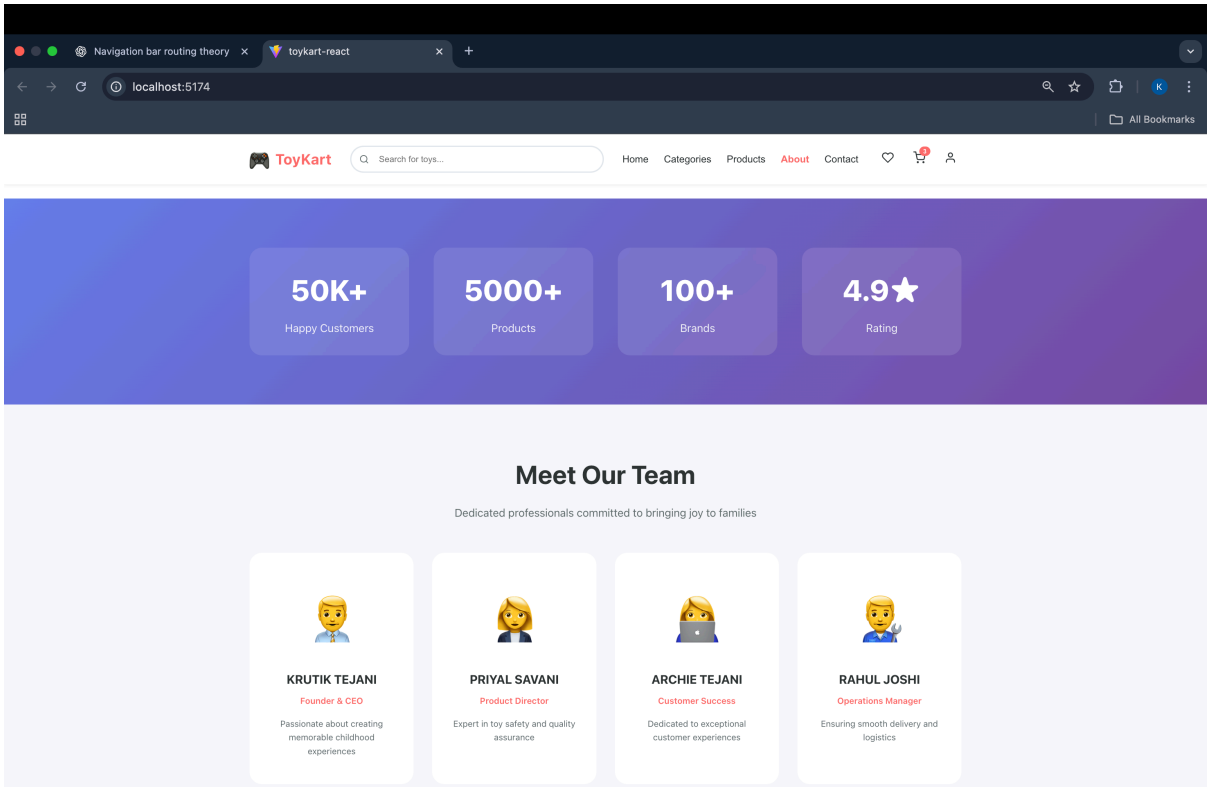
```
const AboutMission = () => {
  const missions = [
    {
      id: 1,
      icon: "🎯",
      title: "Our Mission",
      description: "To provide high-quality, safe, and educational toys that spark creativity and joy in children worldwide."
    },
    {
      id: 2,
      icon: "👁️",
      title: "Our Vision",
      description: "To become the most trusted toy destination where families find products that grow with their children."
    },
    {
      id: 3,
      icon: "💎",
      title: "Our Values",
      description: "Quality, Safety, Innovation, and Customer Satisfaction are at the heart of everything we do."
    }
  ];

  return (
    <section className="about-mission">
      <div className="container">
        <div className="mission-grid">
          {missions.map(mission => (
            <div key={mission.id} className="mission-card">
              <div className="mission-icon">{mission.icon}</div>
              <h3 className="mission-title">{mission.title}</h3>
              <p className="mission-description">{mission.description}</p>
            </div>
          ))}
        </div>
      </div>
    </section>
  );
};

export default AboutMission;



const AboutStats = () => {
  const stats = [
    { id: 1, number: "50K+", label: "Happy Customers" },
    { id: 2, number: "5000+", label: "Products" },
    { id: 3, number: "100+", label: "Brands" },
    { id: 4, number: "4.9★", label: "Rating" }
```

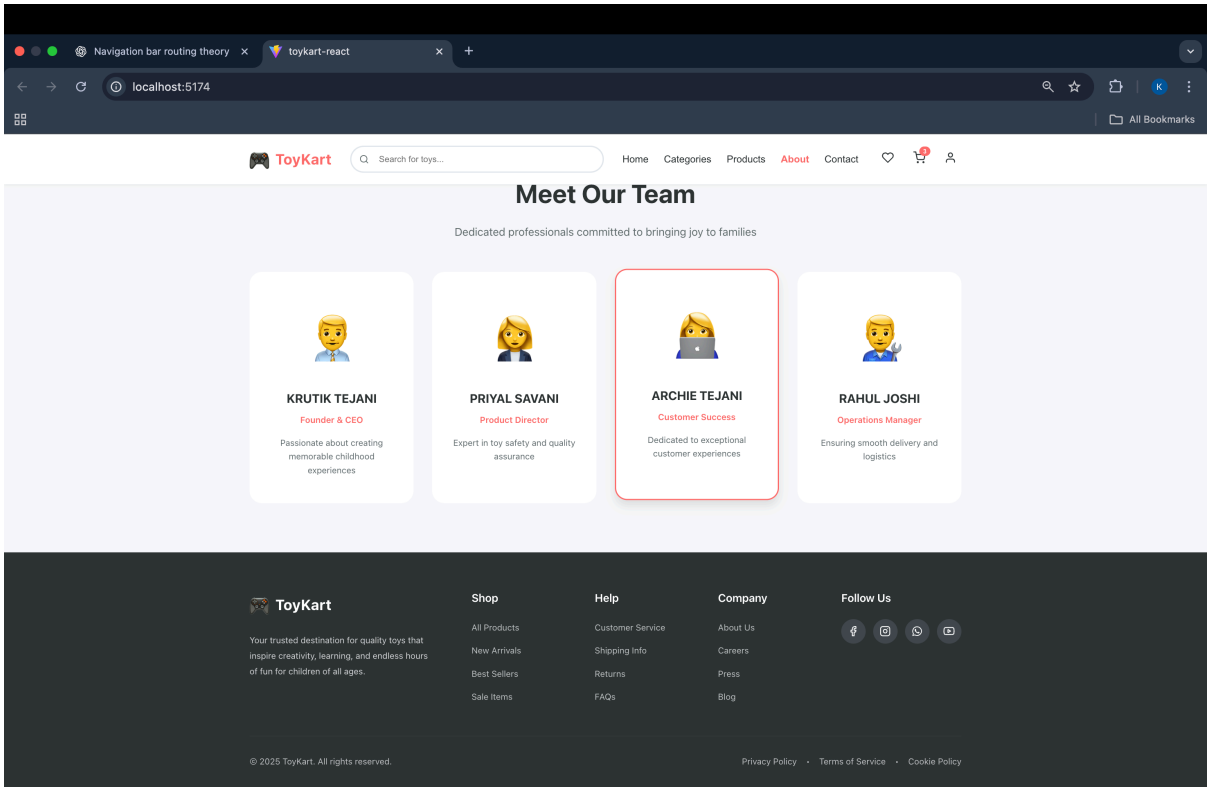
```
  ];

  return (
    <section className="about-stats">
      <div className="container">
        <div className="stats-grid">
          {stats.map(stat => (
            <div key={stat.id} className="stat-box">
              <h3 className="stat-number">{stat.number}</h3>
              <p className="stat-label">{stat.label}</p>
            </div>
          ))}
        </div>
      </div>
    </section>
  );
};

export default AboutStats;

const AboutTeam = () => {
  const team = [
    {
      id: 1,
      name: "KRUTIK TEJANI",
      role: "Founder & CEO",
      avatar: " 👨‍💼 ",
      description: "Passionate about creating memorable childhood experiences"
    },
    {
      id: 2,
      name: "PRIYAL SAVANI",
      role: "Product Director",
      avatar: " 👩 ",
      description: "Expert in toy safety and quality assurance"
    },
    {
      id: 3,
      name: "ARCHIE TEJANI",
      role: "Customer Success",
      avatar: " 👩‍💻 ",
      description: "Dedicated to exceptional customer experiences"
    },
    {
      id: 4,
      name: "RAHUL JOSHI",
      role: "Operations Manager",
      avatar: " 👨‍🔧 ",
      description: "Ensuring smooth delivery and logistics"
    }
  ];
```

```
 return (
  <section className="about-team">
    <div className="container">
      <div className="section-header">
        <h2 className="section-title">Meet Our Team</h2>
        <p className="section-subtitle">
          Dedicated professionals committed to bringing joy to families
        </p>
      </div>
      <div className="team-grid">
        {team.map(member => (
          <div key={member.id} className="team-card">
            <div className="team-avatar">{member.avatar}</div>
            <h3 className="team-name">{member.name}</h3>
            <p className="team-role">{member.role}</p>
            <p className="team-description">{member.description}</p>
          </div>
        ))}
      </div>
    </div>
  </section>
 );
};

export default AboutTeam;
```

OutPut:




Navigation bar routing theory
toykart-react
localhost:5174
All Bookmarks
ToyKart
Search for toys...
Home
Categories
Products
About
Contact
50K+
Happy Customers
5000+
Products
100+
Brands
4.9★
Rating
Meet Our Team
Dedicated professionals committed to bringing joy to families
KRUTIK TEJANI
Founder & CEO
Passionate about creating memorable childhood experiences
PRIYAL SAVANI
Product Director
Expert in toy safety and quality assurance
ARCHIE TEJANI
Customer Success
Dedicated to exceptional customer experiences
RAHUL JOSHI
Operations Manager
Ensuring smooth delivery and logistics


Navigation bar routing theory
toykart-react
localhost:5174
All Bookmarks
ToyKart
Search for toys...
Home
Categories
Products
About
Contact
Meet Our Team
Dedicated professionals committed to bringing joy to families
KRUTIK TEJANI
Founder & CEO
Passionate about creating memorable childhood experiences
PRIYAL SAVANI
Product Director
Expert in toy safety and quality assurance
ARCHIE TEJANI
Customer Success
Dedicated to exceptional customer experiences
RAHUL JOSHI
Operations Manager
Ensuring smooth delivery and logistics
ToyKart
Your trusted destination for quality toys that inspire creativity, learning, and endless hours of fun for children of all ages.
Shop
All Products
New Arrivals
Best Sellers
Sale Items
Help
Customer Service
Shipping Info
Returns
FAQs
Company
About Us
Careers
Press
Blog
Follow Us
© 2025 ToyKart. All rights reserved.
Privacy Policy
Terms of Service
Cookie Policy

Conclusion:

The **About Page** of the **ToyKart ReactJS e-commerce website** serves as an informative and engaging section that reflects the brand's purpose, ethics, and values. It allows visitors to understand ToyKart's dedication to offering safe, quality, and fun toys for all age groups.

Developed using **React functional components** and **React Router**, the About Page maintains consistency with the website's design and structure while promoting trust and professionalism.
This page not only enhances user experience but also contributes to ToyKart's brand credibility and customer loyalty.