# Developer Guide - Ramesh Aqua Project

## 📚 Table of Contents



## 🎯 Introduction

Welcome to the Ramesh Aqua project! This guide will help you write clean, maintainable code that follows our project standards. Whether you're new to React or an experienced developer, this guide will ensure your code fits seamlessly into our codebase.

### Tech Stack

- **Frontend**: React 18 with Hooks
- **State Management**: Redux Toolkit
- **Backend**: Firebase (Firestore, Authentication)
- **Styling**: React-Bootstrap + Custom CSS
- **Build Tool**: Vite
- **Routing**: React Router v6

## 📁 Project Structure

```
react-app/
├── src/
│   ├── components/           # React components
│   │   ├── admin/           # Admin-only components
│   │   ├── auth/            # Login, signup components
│   │   ├── common/          # Reusable UI components
│   │   ├── layout/          # Header, Footer, etc.
│   │   ├── mobile/          # Mobile-specific components
│   │   ├── pages/           # Page components
│   │   └── products/        # Product-related components
│   │
│   ├── firebase/            # Firebase configuration & services
│   │   ├── config.js        # Firebase initialization
│   │   ├── firestoreService.js  # Database operations
│   │   └── migration.js     # Data migration scripts
│   │
│   ├── store/               # Redux store
│   │   ├── store.js         # Store configuration
│   │   └── slices/          # Redux slices (state + actions)
│   │
│   ├── hooks/               # Custom React hooks
│   ├── contexts/            # React Context providers
│   ├── utils/               # Utility functions
│   ├── assets/              # Images, icons, etc.
│   │
│   ├── App.jsx              # Main app component
│   └── main.jsx             # Entry point
│
├── public/                  # Static assets
├── firestore.rules          # Firebase security rules
└── package.json             # Dependencies
```

### 🔑 Key Principles

1. **One component per file** - Each component gets its own file
2. **Co-locate styles** - CSS file next to component file
3. **Group by feature** - Related components in same folder
4. **Clear naming** - File names match component names

## ✍️ Code Formatting Standards

### File Naming Conventions

```
// ✅ CORRECT
components/
  ├── ProductCard.jsx          // PascalCase for components
  ├── ProductCard.css          // Matches component name
  ├── index.js                 // Barrel exports
  └── useDeviceDetect.js       // camelCase for hooks

// ❌ WRONG
components/
  ├── product-card.jsx         // Don't use kebab-case
  ├── productcard.jsx          // Don't use lowercase
  └── Product_Card.jsx         // Don't use snake_case
```

## Component File Structure

Every component file should follow this structure:

```
// 1. IMPORTS - Group and order imports
import { useState, useEffect } from 'react';                    // React imports
import { useNavigate } from 'react-router-dom';                 // Third-party imports
import { Container, Row, Col, Button } from 'react-bootstrap';  // UI library
import { useSelector, useDispatch } from 'react-redux';         // State management
import { FaShoppingCart, FaHeart } from 'react-icons/fa';       // Icons
import { fetchProducts } from '../../firebase/firestoreService'; // Local imports
import './ProductCard.css';                                      // Styles (last)

// 2. COMPONENT DEFINITION
const ProductCard = ({ product }) => {
  // 3. HOOKS (in this order)
  const navigate = useNavigate();              // Router hooks
  const dispatch = useDispatch();              // Redux hooks
  const user = useSelector(state => state.auth.user);  // Redux selectors

  // 4. STATE
  const [loading, setLoading] = useState(false);
  const [error, setError] = useState(null);

  // 5. EFFECTS
  useEffect(() => {
    // Effect logic
  }, [dependencies]);

  // 6. EVENT HANDLERS
  const handleClick = () => {
    // Handler logic
  };
```

```
  const handleAddToCart = async () => {
    // Async handler
  };

  // 7. HELPER FUNCTIONS
  const formatPrice = (price) => {
    return `₹${price}`;
  };

  // 8. CONDITIONAL RENDERING
  if (loading) return <Spinner />;
  if (error) return <Alert>{error}</Alert>;

  // 9. MAIN RENDER
  return (
    <div className="product-card">
      {/* JSX content */}
    </div>
  );
};

// 10. EXPORT
export default ProductCard;
```

## Indentation & Spacing

```
// ✅ CORRECT - 2 spaces indentation
const MyComponent = () => {
  return (
    <div>
      <h1>Title</h1>
      <p>Content</p>
    </div>
  );
};

// ❌ WRONG - Inconsistent spacing
const MyComponent = () => {
return (
<div>
    <h1>Title</h1>
        <p>Content</p>
</div>
);
};
```

## Naming Conventions

```
// ✅ CORRECT
const ProductCard = () => {};           // PascalCase for components
const handleClick = () => {};           // camelCase for functions
const isLoading = true;                 // camelCase for variables
const MAX_ITEMS = 10;                   // UPPER_CASE for constants
const useDeviceDetect = () => {};       // camelCase with 'use' prefix for hooks

// ❌ WRONG
const product_card = () => {};          // Don't use snake_case
const HandleClick = () => {};           // Don't use PascalCase for functions
const IsLoading = true;                 // Don't use PascalCase for variables
const maxItems = 10;                    // Don't use camelCase for constants
```

## Comments

```
// ✅ CORRECT - Clear, meaningful comments

// Fetch products from Firebase when category changes
useEffect(() => {
  loadProducts();
}, [categoryId]);

/**
 * Handles adding product to cart
 * @param {Object} product - Product object with id, title, price
 */
const handleAddToCart = (product) => {
  dispatch(addToCart(product));
};

// ❌ WRONG - Obvious or unclear comments

// Set loading to true
setLoading(true);  // This is obvious from the code

// Do something
handleClick();     // Too vague
```

# 🧩 React Component Guidelines

## Functional Components (ALWAYS USE)

```
// ✅ CORRECT - Use functional components with hooks
const ProductCard = ({ product, onAddToCart }) => {
  const [quantity, setQuantity] = useState(1);

  return (
    <div className="product-card">
      <h3>{product.title}</h3>
      <button onClick={() => onAddToCart(product, quantity)}>
        Add to Cart
      </button>
    </div>
  );
};

// ❌ WRONG - Don't use class components
class ProductCard extends React.Component {
  constructor(props) {
    super(props);
    this.state = { quantity: 1 };
  }

  render() {
    return <div>...</div>;
  }
}
```

## Props Destructuring

```
// ✅ CORRECT - Destructure props for clarity
const ProductCard = ({ product, onAddToCart, showWishlist = true }) => {
  return (
    <div>
      <h3>{product.title}</h3>
      <p>{product.price}</p>
      {showWishlist && <button>Add to Wishlist</button>}
    </div>
  );
};

// ❌ WRONG - Using props object makes code verbose
const ProductCard = (props) => {
  return (
    <div>
      <h3>{props.product.title}</h3>
      <p>{props.product.price}</p>
      {props.showWishlist && <button>Add to Wishlist</button>}
    </div>
```

```
  );
};
```

## Conditional Rendering

```
// ✅ CORRECT - Clear conditional rendering patterns

// Pattern 1: Early returns for loading/error states
if (loading) return <Spinner animation="border" />;
if (error) return <Alert variant="danger">{error}</Alert>;
if (!product) return <div>Product not found</div>;

// Pattern 2: Inline conditions with &&
{product.discount && (
  <Badge bg="danger">{product.discount}% OFF</Badge>
)}

// Pattern 3: Ternary for either/or
{isInCart ? (
  <Button variant="success">In Cart</Button>
) : (
  <Button variant="primary">Add to Cart</Button>
)}

// ❌ WRONG - Mixing return types or complex nested conditions
const renderContent = () => {
  if (loading) {
    return <Spinner />;
  } else {
    if (error) {
      return <Alert>{error}</Alert>;
    } else {
      if (product) {
        return <div>{product.title}</div>;
      }
    }
  }
};
```

## Event Handlers

```
// ✅ CORRECT - Named event handlers with clear purpose

const ProductCard = ({ product }) => {
  const dispatch = useDispatch();
```

```
  // Handler for simple actions
  const handleAddToCart = () => {
    dispatch(addToCart(product));
  };

  // Handler with parameters
  const handleQuantityChange = (newQuantity) => {
    setQuantity(newQuantity);
  };

  // Async handler
  const handleAddToWishlist = async () => {
    try {
      setLoading(true);
      await addToWishlist(user.uid, product);
      showSuccessMessage('Added to wishlist!');
    } catch (error) {
      showErrorMessage('Failed to add to wishlist');
    } finally {
      setLoading(false);
    }
  };

  return (
    <div>
      <Button onClick={handleAddToCart}>Add to Cart</Button>
      <Button onClick={handleAddToWishlist}>Add to Wishlist</Button>
    </div>
  );
};

// ❌ WRONG - Inline functions or unclear names
<Button onClick={() => dispatch(addToCart(product))}>Add</Button>
<Button onClick={handle}>Add</Button>  // Too vague
<Button onClick={doStuff}>Add</Button> // Unclear purpose
```

## Component Size

```
// ✅ CORRECT - Break large components into smaller ones

// Main component
const ProductDetailPage = () => {
  return (
    <Container>
      <ProductImages images={product.images} />
      <ProductInfo product={product} />
```

```
        <ProductActions product={product} />
        <ProductSpecs specs={product.specifications} />
      </Container>
    );
  };

  // Smaller, focused components
  const ProductImages = ({ images }) => {
    const [selected, setSelected] = useState(0);
    return (
      <div>
        <img src={images[selected]} />
        {/* Thumbnails */}
      </div>
    );
  };

  // ❌ WRONG - One giant component with 500+ lines
  const ProductDetailPage = () => {
    // 500 lines of JSX...
  };
```

# 🔥 Firebase Operations

## Firebase Service Pattern

All Firebase operations should go in `firebase/firestoreService.js`. Never write Firebase code directly in components.

```
  // ✅ CORRECT - Firebase operations in service file

  // In firebase/firestoreService.js
  import { collection, doc, getDoc, getDocs, query, where } from 'firebase/firestore';
  import { db } from './config';

  /**
   * Fetch a single product by ID
   * @param {string} productId - The product ID
   * @returns {Promise<Object|null>} Product object or null if not found
   */
  export const fetchProductById = async (productId) => {
    try {
      const productRef = doc(db, 'products', productId);
      const productDoc = await getDoc(productRef);
```

```
      if (productDoc.exists()) {
        console.log(`✅ Fetched product: ${productId}`);
        return {
          id: productDoc.id,
          ...productDoc.data()
        };
      } else {
        console.warn(`⚠️ Product not found: ${productId}`);
        return null;
      }
    } catch (error) {
      console.error(`❌ Error fetching product ${productId}:`, error);
      throw error;
    }
  };

  // In your component
  import { fetchProductById } from '../../firebase/firestoreService';

  const ProductDetailPage = () => {
    const { productId } = useParams();
    const [product, setProduct] = useState(null);
    const [loading, setLoading] = useState(true);
    const [error, setError] = useState(null);

    useEffect(() => {
      const loadProduct = async () => {
        try {
          setLoading(true);
          setError(null);
          const data = await fetchProductById(productId);
          setProduct(data);
        } catch (err) {
          setError('Failed to load product');
        } finally {
          setLoading(false);
        }
      };

      loadProduct();
    }, [productId]);

    // Render component...
  };

  // ❌ WRONG - Firebase code directly in component
  const ProductDetailPage = () => {
    useEffect(() => {
      // DON'T DO THIS!
```

```
    const productRef = doc(db, 'products', productId);
    getDoc(productRef).then(doc => {
      setProduct(doc.data());
    });
  }, []);
};
```

## Firebase CRUD Operations

```
// CREATE - Add new document
export const createProduct = async (productData) => {
  try {
    const productsRef = collection(db, 'products');
    const docRef = await addDoc(productsRef, {
      ...productData,
      createdAt: serverTimestamp()
    });

    console.log(`✅ Created product: ${docRef.id}`);
    return docRef.id;
  } catch (error) {
    console.error('❌ Error creating product:', error);
    throw error;
  }
};

// READ - Fetch single document
export const fetchProductById = async (productId) => {
  try {
    const productRef = doc(db, 'products', productId);
    const productDoc = await getDoc(productRef);

    if (!productDoc.exists()) {
      return null;
    }

    return {
      id: productDoc.id,
      ...productDoc.data()
    };
  } catch (error) {
    console.error(`❌ Error fetching product:`, error);
    throw error;
  }
};

// READ - Fetch multiple documents with query
```

```
export const fetchProductsByCategory = async (categoryId) => {
  try {
    const productsRef = collection(db, 'products');
    const q = query(
      productsRef,
      where('categoryId', '==', categoryId)
    );
    const querySnapshot = await getDocs(q);

    const products = [];
    querySnapshot.forEach((doc) => {
      products.push({
        id: doc.id,
        ...doc.data()
      });
    });

    console.log(`✅ Fetched ${products.length} products`);
    return products;
  } catch (error) {
    console.error('❌ Error fetching products:', error);
    throw error;
  }
};

// UPDATE - Update existing document
export const updateProduct = async (productId, updates) => {
  try {
    const productRef = doc(db, 'products', productId);
    await updateDoc(productRef, {
      ...updates,
      updatedAt: serverTimestamp()
    });

    console.log(`✅ Updated product: ${productId}`);
  } catch (error) {
    console.error('❌ Error updating product:', error);
    throw error;
  }
};

// DELETE - Delete document
export const deleteProduct = async (productId) => {
  try {
    const productRef = doc(db, 'products', productId);
    await deleteDoc(productRef);

    console.log(`✅ Deleted product: ${productId}`);
  } catch (error) {
```

```
      console.error('❌ Error deleting product:', error);
      throw error;
    }
  };
```

## Error Handling in Firebase Operations

```
  // ✅ CORRECT - Comprehensive error handling

  const loadProducts = async () => {
    try {
      setLoading(true);
      setError(null);

      const products = await fetchAllProducts();

      if (!products || products.length === 0) {
        setError('No products found');
        return;
      }

      setProducts(products);
    } catch (error) {
      console.error('Error loading products:', error);

      // User-friendly error messages
      if (error.code === 'permission-denied') {
        setError('You don\'t have permission to view products');
      } else if (error.code === 'unavailable') {
        setError('Unable to connect to the server. Please check your internet connection.');
      } else {
        setError('Failed to load products. Please try again.');
      }
    } finally {
      setLoading(false);
    }
  };

  // ❌ WRONG - No error handling
  const loadProducts = async () => {
    const products = await fetchAllProducts();
    setProducts(products);
  };
```

## Firebase Security Rules

Always test your operations against security rules:

```
// In firestore.rules
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {
    // Products - anyone can read
    match /products/{productId} {
      allow read: if true;
      allow write: if request.auth.token.admin == true;
    }

    // Wishlists - only owner can access
    match /wishlists/{userId}/items/{productId} {
      allow read, write: if request.auth.uid == userId;
    }
  }
}
```

# 📱 Mobile UI Development

## Responsive Design Strategy

Our app uses a **mobile-first** approach with three breakpoints:

```
/* Mobile: < 768px (default styles) */
.container {
  padding: 10px;
  font-size: 14px;
}

/* Tablet: 768px - 1024px */
@media (min-width: 768px) {
  .container {
    padding: 20px;
    font-size: 16px;
  }
}

/* Desktop: > 1024px */
@media (min-width: 1024px) {
  .container {
    padding: 30px;
    max-width: 1200px;
```

```
      margin: 0 auto;
    }
  }
```

## Device Detection Hook

Use our custom hook for device detection:

```
// ✅ CORRECT - Using useDeviceDetect hook
import { useDeviceDetect } from '../../hooks/useDeviceDetect';

const MyComponent = () => {
  const { isMobile, isTablet, isDesktop } = useDeviceDetect();

  return (
    <div>
      {isMobile && <MobileHeader />}
      {isDesktop && <DesktopHeader />}

      <div className={isMobile ? 'mobile-content' : 'desktop-content'}>
        {/* Content */}
      </div>
    </div>
  );
};

// The hook implementation (already created)
// hooks/useDeviceDetect.js
import { useState, useEffect } from 'react';

export const useDeviceDetect = () => {
  const [screenWidth, setScreenWidth] = useState(window.innerWidth);

  useEffect(() => {
    const handleResize = () => setScreenWidth(window.innerWidth);
    window.addEventListener('resize', handleResize);
    return () => window.removeEventListener('resize', handleResize);
  }, []);

  return {
    isMobile: screenWidth < 768,
    isTablet: screenWidth >= 768 && screenWidth < 1024,
    isDesktop: screenWidth >= 1024,
    screenWidth
  };
};
```

## Mobile Component Pattern

Mobile components follow these guidelines:

```
// ✅ CORRECT - Mobile component structure

import { useState } from 'react';
import { useNavigate, useLocation } from 'react-router-dom';
import './BottomNavigation.css';

const BottomNavigation = () => {
  const navigate = useNavigate();
  const location = useLocation();

  // Navigation items configuration
  const navItems = [
    { id: 'home', icon: 'FaHome', label: 'Home', path: '/' },
    { id: 'categories', icon: 'FaThLarge', label: 'Categories', path: '/categories' },
    { id: 'cart', icon: 'FaShoppingCart', label: 'Cart', path: '/cart', badge: cartCount },
    { id: 'profile', icon: 'FaUser', label: 'Profile', path: '/profile' }
  ];

  const isActive = (path) => location.pathname === path;

  return (
    <nav className="bottom-navigation">
      {navItems.map((item) => (
        <button
          key={item.id}
          className={`nav-item ${isActive(item.path) ? 'active' : ''}`}
          onClick={() => navigate(item.path)}
        >
          <span className="nav-icon">
            {/* Icon component */}
          </span>
          <span className="nav-label">{item.label}</span>
          {item.badge > 0 && (
            <span className="nav-badge">{item.badge}</span>
          )}
        </button>
      ))}
    </nav>
  );
};

export default BottomNavigation;
```

## Mobile CSS Patterns

```
/* ✅ CORRECT - Mobile-first CSS */

/* Base styles (mobile) */
.bottom-navigation {
  position: fixed;
  bottom: 0;
  left: 0;
  right: 0;
  height: 60px;
  background: #ffffff;
  box-shadow: 0 -2px 10px rgba(0, 0, 0, 0.1);
  display: flex;
  justify-content: space-around;
  align-items: center;
  z-index: 1000;
  /* Hide on desktop */
  display: flex;
}

.nav-item {
  flex: 1;
  display: flex;
  flex-direction: column;
  align-items: center;
  gap: 4px;
  padding: 8px;
  border: none;
  background: none;
  color: #666;
  cursor: pointer;
  transition: all 0.3s ease;
  position: relative;
}

.nav-item.active {
  color: #007bff;
}

/* Ripple effect for better mobile UX */
.nav-item::after {
  content: '';
  position: absolute;
  top: 50%;
  left: 50%;
  width: 0;
  height: 0;
```

```
  border-radius: 50%;
  background: rgba(0, 123, 255, 0.2);
  transform: translate(-50%, -50%);
  transition: width 0.3s, height 0.3s;
}

.nav-item:active::after {
  width: 100%;
  height: 100%;
}

/* Badge for notifications */
.nav-badge {
  position: absolute;
  top: 4px;
  right: 8px;
  background: #dc3545;
  color: white;
  font-size: 10px;
  font-weight: bold;
  padding: 2px 6px;
  border-radius: 10px;
  min-width: 18px;
  text-align: center;
}

/* Hide on tablet and desktop */
@media (min-width: 768px) {
  .bottom-navigation {
    display: none;
  }
}
```

## Mobile Container Wrapper

```
// ✅ CORRECT - Use MobileContainer for all pages

import { MobileContainer } from '../components/mobile';

const MyPage = () => {
  return (
    <MobileContainer>
      {/* Your page content */}
      <h1>Page Title</h1>
      <p>Content...</p>
    </MobileContainer>
  );
```

```
};

// MobileContainer handles:
// - Device detection
// - Bottom navigation spacing
// - Safe area insets (for notched phones)
// - Conditional rendering of mobile/desktop layouts
```

## Touch-Friendly Interactive Elements

```
/* ✅ CORRECT - Touch-friendly sizes */

.mobile-button {
  /* Minimum 44x44px for touch targets */
  min-height: 44px;
  min-width: 44px;
  padding: 12px 20px;
  font-size: 16px;

  /* Prevent text selection on touch */
  -webkit-user-select: none;
  user-select: none;

  /* Prevent tap highlight */
  -webkit-tap-highlight-color: transparent;
}

/* Active state for touch feedback */
.mobile-button:active {
  transform: scale(0.95);
  opacity: 0.8;
}

/* ❌ WRONG - Too small for touch */
.tiny-button {
  height: 20px;
  width: 20px;
  padding: 2px;
}
```

## Mobile Header Pattern

```
// ✅ CORRECT - Mobile header with actions

const MobileHeader = ({
```

```
    title,
    showBack = false,
    onBack,
    actions = []
  }) => {
    const navigate = useNavigate();

    const handleBack = () => {
      if (onBack) {
        onBack();
      } else {
        navigate(-1);
      }
    };

    return (
      <header className="mobile-header">
        <div className="header-left">
          {showBack && (
            <button
              className="back-button"
              onClick={handleBack}
              aria-label="Go back"
            >
              <FaArrowLeft />
            </button>
          )}
        </div>

        <div className="header-center">
          <h1 className="header-title">{title}</h1>
        </div>

        <div className="header-right">
          {actions.map((action, index) => (
            <button
              key={index}
              className="header-action"
              onClick={action.onClick}
              aria-label={action.label}
            >
              {action.icon}
            </button>
          ))}
        </div>
      </header>
    );
  };
```

# State Management (Redux)

## Redux Slice Pattern

```
// ✅ CORRECT - Redux slice structure

// store/slices/cartSlice.js
import { createSlice } from '@reduxjs/toolkit';

const initialState = {
  items: [],
  totalQuantity: 0,
  totalPrice: 0
};

const cartSlice = createSlice({
  name: 'cart',
  initialState,
  reducers: {
    // Add item to cart
    addToCart: (state, action) => {
      const { product, quantity = 1 } = action.payload;
      const existingItem = state.items.find(item => item.id === product.id);

      if (existingItem) {
        existingItem.quantity += quantity;
      } else {
        state.items.push({
          ...product,
          quantity
        });
      }

      // Update totals
      state.totalQuantity = state.items.reduce((sum, item) => sum + item.quantity, 0);
      state.totalPrice = state.items.reduce((sum, item) =>
        sum + (parseFloat(item.price) * item.quantity), 0
      );
    },

    // Remove item from cart
    removeFromCart: (state, action) => {
      const productId = action.payload;
      state.items = state.items.filter(item => item.id !== productId);

      // Update totals
```

```
        state.totalQuantity = state.items.reduce((sum, item) => sum + item.quantity, 0);
        state.totalPrice = state.items.reduce((sum, item) =>
          sum + (parseFloat(item.price) * item.quantity), 0
        );
      },

      // Increment quantity
      incrementQuantity: (state, action) => {
        const productId = action.payload;
        const item = state.items.find(item => item.id === productId);

        if (item) {
          item.quantity += 1;
          state.totalQuantity += 1;
          state.totalPrice += parseFloat(item.price);
        }
      },

      // Decrement quantity
      decrementQuantity: (state, action) => {
        const productId = action.payload;
        const item = state.items.find(item => item.id === productId);

        if (item && item.quantity > 1) {
          item.quantity -= 1;
          state.totalQuantity -= 1;
          state.totalPrice -= parseFloat(item.price);
        }
      },

      // Clear cart
      clearCart: (state) => {
        state.items = [];
        state.totalQuantity = 0;
        state.totalPrice = 0;
      }
    }
  });

  export const {
    addToCart,
    removeFromCart,
    incrementQuantity,
    decrementQuantity,
    clearCart
  } = cartSlice.actions;

  export default cartSlice.reducer;
```

## Using Redux in Components

```
// ✅ CORRECT - Using Redux in components

import { useSelector, useDispatch } from 'react-redux';
import { addToCart, incrementQuantity, decrementQuantity } from '../../store/slices/cartSlice';

const ProductCard = ({ product }) => {
  const dispatch = useDispatch();

  // Select data from store
  const cartItems = useSelector(state => state.cart.items);
  const cartItem = cartItems.find(item => item.id === product.id);
  const quantity = cartItem ? cartItem.quantity : 0;

  // Dispatch actions
  const handleAddToCart = () => {
    dispatch(addToCart({ product, quantity: 1 }));
  };

  const handleIncrement = () => {
    dispatch(incrementQuantity(product.id));
  };

  const handleDecrement = () => {
    dispatch(decrementQuantity(product.id));
  };

  return (
    <div>
      {quantity === 0 ? (
        <button onClick={handleAddToCart}>Add to Cart</button>
      ) : (
        <div className="quantity-controls">
          <button onClick={handleDecrement}>-</button>
          <span>{quantity}</span>
          <button onClick={handleIncrement}>+</button>
        </div>
      )}
    </div>
  );
};
```

## 🎨 Styling Guidelines

## CSS Organization

```
/* ✅ CORRECT - Organized CSS */

/* ================================
   COMPONENT NAME
   ================================ */

/* Layout & Structure */
.product-card {
  display: flex;
  flex-direction: column;
  border-radius: 8px;
  overflow: hidden;
}

/* Typography */
.product-title {
  font-size: 18px;
  font-weight: 600;
  color: #333;
}

/* Colors & Backgrounds */
.product-card {
  background: #ffffff;
  border: 1px solid #e0e0e0;
}

/* Spacing */
.product-card {
  padding: 16px;
  margin-bottom: 16px;
}

/* States */
.product-card:hover {
  box-shadow: 0 4px 12px rgba(0, 0, 0, 0.1);
  transform: translateY(-2px);
}

.product-card.loading {
  opacity: 0.6;
  pointer-events: none;
}

/* Responsive */
@media (min-width: 768px) {
```

```
    .product-card {
      flex-direction: row;
    }
  }
```

## CSS Class Naming (BEM Convention)

```
/* ✅ CORRECT - BEM naming */

/* Block */
.product-card { }

/* Element (part of block) */
.product-card__image { }
.product-card__title { }
.product-card__price { }

/* Modifier (variation of block/element) */
.product-card--featured { }
.product-card__price--discounted { }

/* Example usage */
<div className="product-card product-card--featured">
  <img className="product-card__image" />
  <h3 className="product-card__title">Product Name</h3>
  <span className="product-card__price product-card__price--discounted">
    ₹299
  </span>
</div>
```

## Colors & Variables

```
/* ✅ CORRECT - Use CSS variables */

:root {
  /* Brand Colors */
  --primary-color: #007bff;
  --secondary-color: #6c757d;
  --success-color: #28a745;
  --danger-color: #dc3545;
  --warning-color: #ffc107;
  --info-color: #17a2b8;

  /* Neutral Colors */
  --text-primary: #212529;
```

```
    --text-secondary: #6c757d;
    --text-muted: #999;
    --border-color: #dee2e6;
    --background-color: #f8f9fa;

    /* Spacing */
    --spacing-xs: 4px;
    --spacing-sm: 8px;
    --spacing-md: 16px;
    --spacing-lg: 24px;
    --spacing-xl: 32px;

    /* Border Radius */
    --radius-sm: 4px;
    --radius-md: 8px;
    --radius-lg: 12px;
    --radius-full: 9999px;

    /* Shadows */
    --shadow-sm: 0 1px 2px rgba(0, 0, 0, 0.05);
    --shadow-md: 0 4px 6px rgba(0, 0, 0, 0.1);
    --shadow-lg: 0 10px 15px rgba(0, 0, 0, 0.1);
  }

  /* Usage */
  .button-primary {
    background: var(--primary-color);
    padding: var(--spacing-md);
    border-radius: var(--radius-md);
    box-shadow: var(--shadow-md);
  }
```

# 🔧 Common Patterns

## Loading States

```
  // ✅ CORRECT - Loading state pattern

  const ProductsPage = () => {
    const [products, setProducts] = useState([]);
    const [loading, setLoading] = useState(true);
    const [error, setError] = useState(null);

    useEffect(() => {
      const loadProducts = async () => {
```

```
      try {
        setLoading(true);
        setError(null);

        const data = await fetchAllProducts();
        setProducts(data);
      } catch (err) {
        setError('Failed to load products');
      } finally {
        setLoading(false);
      }
    };

    loadProducts();
  }, []);

  // Loading state
  if (loading) {
    return (
      <Container className="text-center py-5">
        <Spinner animation="border" variant="primary" />
        <p className="mt-3">Loading products...</p>
      </Container>
    );
  }

  // Error state
  if (error) {
    return (
      <Container className="py-5">
        <Alert variant="danger">
          <Alert.Heading>Error</Alert.Heading>
          <p>{error}</p>
          <Button onClick={() => window.location.reload()}>
            Try Again
          </Button>
        </Alert>
      </Container>
    );
  }

  // Empty state
  if (products.length === 0) {
    return (
      <Container className="text-center py-5">
        <h3>No products found</h3>
        <Button onClick={() => navigate('/categories')}>
          Browse Categories
        </Button>
```

```
        </Container>
      );
    }

    // Success state
    return (
      <Container>
        <Row>
          {products.map(product => (
            <Col key={product.id} md={4}>
              <ProductCard product={product} />
            </Col>
          ))}
        </Row>
      </Container>
    );
  };
```

## Form Handling

```
// ✅ CORRECT - Form handling pattern

const LoginForm = () => {
  const [formData, setFormData] = useState({
    email: '',
    password: ''
  });
  const [errors, setErrors] = useState({});
  const [submitting, setSubmitting] = useState(false);

  // Handle input change
  const handleChange = (e) => {
    const { name, value } = e.target;
    setFormData(prev => ({
      ...prev,
      [name]: value
    }));

    // Clear error for this field
    if (errors[name]) {
      setErrors(prev => ({
        ...prev,
        [name]: ''
      }));
    }
  };
```

```
  // Validate form
  const validate = () => {
    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;
  };

  // Handle form submission
  const handleSubmit = async (e) => {
    e.preventDefault();

    // Validate
    const newErrors = validate();
    if (Object.keys(newErrors).length > 0) {
      setErrors(newErrors);
      return;
    }

    // Submit
    try {
      setSubmitting(true);
      await loginUser(formData.email, formData.password);
      // Success - redirect or show message
    } catch (error) {
      setErrors({ submit: error.message });
    } finally {
      setSubmitting(false);
    }
  };

  return (
    <Form onSubmit={handleSubmit}>
      <Form.Group className="mb-3">
        <Form.Label>Email</Form.Label>
        <Form.Control
          type="email"
          name="email"
```

```
            value={formData.email}
            onChange={handleChange}
            isInvalid={!!errors.email}
          />
          <Form.Control.Feedback type="invalid">
            {errors.email}
          </Form.Control.Feedback>
        </Form.Group>

        <Form.Group className="mb-3">
          <Form.Label>Password</Form.Label>
          <Form.Control
            type="password"
            name="password"
            value={formData.password}
            onChange={handleChange}
            isInvalid={!!errors.password}
          />
          <Form.Control.Feedback type="invalid">
            {errors.password}
          </Form.Control.Feedback>
        </Form.Group>

        {errors.submit && (
          <Alert variant="danger">{errors.submit}</Alert>
        )}

        <Button
          type="submit"
          variant="primary"
          disabled={submitting}
        >
          {submitting ? 'Logging in...' : 'Login'}
        </Button>
      </Form>
    );
};
```

## Modal/Dialog Pattern

```
// ✅ CORRECT - Modal pattern

const ConfirmDialog = ({
  show,
  title,
  message,
  onConfirm,
```

```
    onCancel,
    confirmText = 'Confirm',
    cancelText = 'Cancel'
  }) => {
    const [processing, setProcessing] = useState(false);

    const handleConfirm = async () => {
      try {
        setProcessing(true);
        await onConfirm();
      } catch (error) {
        console.error('Error:', error);
      } finally {
        setProcessing(false);
      }
    };

    return (
      <Modal show={show} onHide={onCancel} centered>
        <Modal.Header closeButton>
          <Modal.Title>{title}</Modal.Title>
        </Modal.Header>

        <Modal.Body>
          <p>{message}</p>
        </Modal.Body>

        <Modal.Footer>
          <Button
            variant="secondary"
            onClick={onCancel}
            disabled={processing}
          >
            {cancelText}
          </Button>
          <Button
            variant="primary"
            onClick={handleConfirm}
            disabled={processing}
          >
            {processing ? 'Processing...' : confirmText}
          </Button>
        </Modal.Footer>
      </Modal>
    );
  };

  // Usage
  const MyComponent = () => {
```

```
  const [showDialog, setShowDialog] = useState(false);

  const handleDelete = async () => {
    await deleteProduct(productId);
    setShowDialog(false);
    // Show success message
  };

  return (
    <>
      <Button onClick={() => setShowDialog(true)}>
        Delete Product
      </Button>

      <ConfirmDialog
        show={showDialog}
        title="Delete Product"
        message="Are you sure you want to delete this product? This action cannot be undone."
        onConfirm={handleDelete}
        onCancel={() => setShowDialog(false)}
        confirmText="Delete"
        cancelText="Cancel"
      />
    </>
  );
};
```

# 🧪 Testing Your Code

## Manual Testing Checklist

Before committing code, test these scenarios:

```
// ✅ Testing Checklist

// 1. Loading States
// - Does spinner show while loading?
// - Is content hidden during loading?
// - Does loading state clear after data loads?

// 2. Error States
// - What happens if Firebase is down?
// - What if user has no internet?
// - Are error messages user-friendly?
```

```
// 3. Empty States
// - What if there are no products?
// - What if user has empty cart?
// - Are empty states helpful?

// 4. Success States
// - Does data display correctly?
// - Are images loading?
// - Is formatting correct?

// 5. User Interactions
// - Do buttons work on click?
// - Do forms validate correctly?
// - Are success messages shown?

// 6. Mobile Testing
// - Test on mobile viewport (375px)
// - Test on tablet (768px)
// - Test on desktop (1024px+)
// - Touch targets at least 44x44px?

// 7. Performance
// - Are images optimized?
// - Are there unnecessary re-renders?
// - Is data cached when appropriate?
```

### Browser Console Testing

```
// Check for errors
console.log('✅ Component mounted');
console.error('❌ Error occurred:', error);
console.warn('⚠️ Warning:', warning);

// Debug data
console.log('Products:', products);
console.log('User:', user);
console.log('Cart Items:', cartItems);
```

## 🐛 Troubleshooting

### Common Issues & Solutions

#### Issue 1: Component not updating

```
// ❌ PROBLEM
const MyComponent = ({ product }) => {
  const [data, setData] = useState(product);
  // data never updates when product prop changes!
};

// ✅ SOLUTION
const MyComponent = ({ product }) => {
  const [data, setData] = useState(product);

  useEffect(() => {
    setData(product);
  }, [product]);
};
```

### Issue 2: Infinite loop in useEffect

```
// ❌ PROBLEM
useEffect(() => {
  setProducts([...products, newProduct]);
}, [products]); // Dependencies include products!

// ✅ SOLUTION - Option 1: Remove dependency
useEffect(() => {
  loadProducts();
}, []); // Empty array = run once

// ✅ SOLUTION - Option 2: Use functional update
useEffect(() => {
  setProducts(prev => [...prev, newProduct]);
}, [newProduct]); // Only depends on newProduct
```

### Issue 3: Firebase permission denied

```
// ❌ PROBLEM: No error handling
const loadData = async () => {
  const data = await fetchProducts();
  setProducts(data);
};

// ✅ SOLUTION: Handle permission errors
const loadData = async () => {
  try {
    const data = await fetchProducts();
    setProducts(data);
```

```
    } catch (error) {
      if (error.code === 'permission-denied') {
        setError('You don\'t have permission to access this data');
      } else {
        setError('Failed to load data');
      }
    }
  };
```

### Issue 4: State not updating immediately

```
// ❌ PROBLEM
const handleClick = () => {
  setCount(count + 1);
  console.log(count); // Shows old value!
};

// ✅ SOLUTION: Use useEffect to log after update
useEffect(() => {
  console.log('Count updated:', count);
}, [count]);

// Or use functional update
const handleClick = () => {
  setCount(prev => {
    const newCount = prev + 1;
    console.log('New count:', newCount);
    return newCount;
  });
};
```

## 📝 Code Review Checklist

Before submitting code for review:

- [ ] No console.log statements (unless intentional)
- [ ] No commented-out code
- [ ] All imports used
- [ ] No unused variables
- [ ] Consistent indentation (2 spaces)
- [ ] Component names match file names
- [ ] All functions have clear names

- [ ] Error handling in place
- [ ] Loading states implemented
- [ ] Mobile responsive
- [ ] Works on Chrome, Firefox, Safari
- [ ] No errors in browser console
- [ ] Firebase operations in service file
- [ ] Redux actions properly dispatched
- [ ] CSS follows BEM convention
- [ ] Touch-friendly on mobile (44px targets)

## 🎓 Learning Resources

### Official Documentation

- React Docs
- Redux Toolkit
- Firebase Docs
- React Router
- React Bootstrap

### Internal Resources

- `MOBILE_UI_IMPLEMENTATION.md` - Mobile UI patterns
- `FIREBASE_SETUP.md` - Firebase configuration
- `PRODUCT_DETAIL_FIREBASE.md` - Firebase integration examples

## 🤝 Getting Help

When you're stuck:

1. **Check the documentation** - Look at existing code examples
2. **Search the codebase** - Similar components might already exist
3. **Check browser console** - Error messages are helpful
4. **Ask for help** - Include error messages and code snippets

## Happy Coding! 🚀

Remember: Write code that you'd want to read in 6 months. Your future self will thank you!