**E-commerce App Frontend Documentation**

**Project Title**

**E-commerce App FrontEnd**

**Description**

The E-commerce App FrontEnd is a responsive and user-friendly interface for an online shopping platform. Built with modern web technologies, it allows users to browse products, add items to their cart, and make purchases.

**License**

This project is licensed under the MIT License. See the LICENSE file for details.

**Author(s)**

(Optional)

* Ahsan Malick - [Ahsan-Malick](https://github.com/Ahsan-Malick)

**FAQ**

**Q: What technologies are used in this project?** A: This project uses mainly React, Redux, and tailwind CSS for the frontend.

**Q: How can I report an issue or suggest a feature?** A: Use the [Issues](https://github.com/Ahsan-Malick/E-commerce-app-FrontEnd/issues) section of this repository.

**Redux Usage**

**State Management with Redux**

Redux manages the application's state in a centralized and predictable manner.

**Key Components**

1. **Store**: Holds the application state
   * const store = createStore(rootReducer, applyMiddleware(thunk));
2. **Reducers**: Specify how the state changes in response to actions.
   * const rootReducer = combineReducers({

products: productReducer,

* + cart: cartReducer,
  + });

1. **Actions**: Payloads of information that send data to the store
   * export const addToCart = (product) => ({ type: ADD\_TO\_CART,payload: product,});
2. **Thunks**: Functions that handle asynchronous operations.
   * export const fetchProducts = () => {return (dispatch) => {fetch('/api/products').then(response => response.json()).then(data => dispatch({ type: 'SET\_PRODUCTS', payload: data }));};};

**Redux in Components**

1. **useSelector**: Extract data from the Redux store state:
   * const products = useSelector((state) => state.products);
2. **useDispatch**: Dispatch actions to the Redux store.
   * const dispatch = useDispatch();

dispatch(addToCart(product));

**Other Hooks Used**

1. **useState**: Adds state to functional components/
   * const [query, setQuery] = useState('');
2. **useEffect**: Performs side effects in functional components.
   * useEffect(() => {

dispatch(fetchProducts());

}, [dispatch]);

1. **useMemo**: Memoizes expensive calculations.
   * const sortedProducts = useMemo(() => {

return products.slice().sort((a, b) => a.price - b.price);

}, [products]);

1. **useContext**: Accesses context for global state management.
   * const { theme } = useContext(ThemeContext);