**Assignment 4 — ShopEase Cart, Forms & Redux**

**Objective:**

Add **Cart** and **Checkout** functionality to the ShopEase app, demonstrate controlled forms with validation, and manage application state using **Redux Toolkit** and **createAsyncThunk** (Thunk).

**Tasks (requirements)**

1. **Login Form (Controlled Components & Validation)**
   * Create a **Login** page with a controlled form (email, password).
   * Validate email format (must be a valid email).
   * Validate password: minimum **6 characters**.
   * Show inline validation messages and prevent submission on invalid input.
   * On successful validation, simulate login (no backend required) and route to Home or Products.
2. **Redux Toolkit Setup**
   * Install and configure **@reduxjs/toolkit** and **react-redux**.
   * Create a Redux store and provide it to the app.
3. **Cart Slice (cartSlice)**
   * Implement cartSlice with reducers/actions:
     + addToCart(product, qty = 1) — adds product or increases quantity.
     + removeFromCart(productId) — removes a product entirely.
     + clearCart() — empties the cart.
     + Optionally, updateQuantity(productId, qty).
   * Store cart items with shape: { id, name, price, qty, subtotal } (or similar).
4. **Show Cart Summary**
   * Display **total items** and **total price** (sum of subtotals) in a Cart summary area (header or cart page).
   * Ensure totals update reactively on add/remove/quantity change.
5. **Async Product Fetch with createAsyncThunk**
   * Use createAsyncThunk to fetch the product list (e.g., from https://fakestoreapi.com/products or a provided mock).
   * Store fetched products in Redux state (e.g., productsSlice).
   * Handle loading and error states in the UI.
6. **Cart Page / Checkout**
   * Create a **Cart** page that lists all cart items with quantity and subtotal.
   * Allow users to **remove** items directly from the cart page and update quantities (if implemented).
   * Provide a **Clear Cart** action/button.
   * (Optional) Show a basic **Checkout** summary with totals and a “Place Order” button that clears the cart.

**Implementation notes (recommended)**

• Use **functional components** and React hooks.

• Keep Redux logic modular: store/, slices/productsSlice.js, slices/cartSlice.js.

• Use createSlice, createAsyncThunk, and configureStore from Redux Toolkit.

• Use useSelector and useDispatch to connect components to Redux.

• For form validation you may use plain validation or a library (e.g., react-hook-form + yup) — plain validation is acceptable.

• Keep UI simple and usable; reuse ProductCard where appropriate.

• No real backend auth required — simulate login state in local state or Redux.

**Deliverables (what to submit)**

1. **README.md** containing:

o Setup and run instructions (npm install, npm run dev).

o Notes on Redux slices, createAsyncThunk usage, and any decisions made.

o How to test the login (if simulated) and cart flows.

1. **ZIP File of Project** uploaded before the deadline.

**Grading Rubric (suggested)**

**Functionality (60%)**

* Redux store configured and provided correctly *(10%)*
* cartSlice with required reducers/actions (addToCart, removeFromCart, clearCart) *(20%)*
* Async fetching of products with createAsyncThunk and proper loading/error handling *(15%)*
* Cart page displays items, allows removal, and shows accurate totals *(15%)*

**Forms & Validation (20%)**

* Controlled Login form implemented *(6%)*
* Email validation (correct format) *(7%)*
* Password validation (min 6 chars) and inline error messages *(7%)*

**Code Quality & Structure (15%)**

* Clear folder structure and slice separation
* Proper use of Redux Toolkit patterns and hooks
* Readable and maintainable code with comments where needed

**Styling & UX (5%)**

* Clean, readable UI for cart and forms
* Basic responsiveness and clear feedback on actions

**Submission Instructions**

• Upload the **ZIP file** of your project before the deadline.

• Ensure **README.md** includes setup steps and notes on Redux slices and the async thunk.