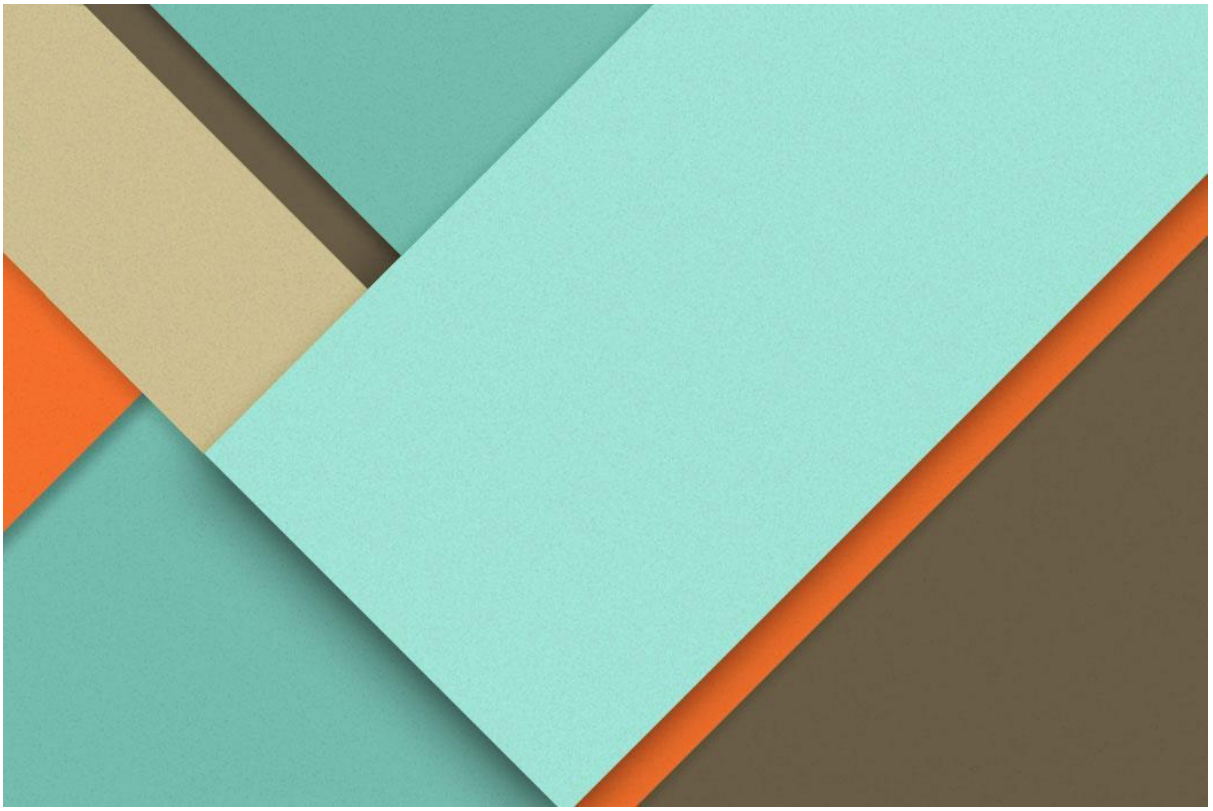

EComm My Store

Development Technical Test

—

ShyftLabs.io

Objective

- You will create a Single Page App that will contain 2 pages of an ECommerce application i.e Product Listing Page & Cart Details Page
- Product Listing Page: A product listing page is a web page or component within a website or e-commerce platform that is specifically designed to display a list of products or items available for sale. It serves as an essential element of online retail and e-commerce websites. The primary purpose of a product listing page is to provide users with an organised and visually appealing way to browse and explore the various products or items offered by the online store.
- Cart Details Page: A cart details page, often referred to as a "shopping cart" or "cart page," is a web page or component within an e-commerce website or application that displays a summary of the items a customer has added to their shopping cart during an online shopping session.
- You can use any backend & frontend technology.
- You can use any type of DB.
- You can use any routing scheme and URI structure.

Deliverables

Upon the receipt of this document, you have 72 hours to complete and deliver the following

- A github repository (to be shared via email) including instructions on how to build your code and see the running app. Please note if we cannot build your code at our end you won't be shortlisted for the next round of interviews.

Should your coding meet the requirements of the role, we will then set up the following

- A presentation of the working app + a small change to the app (to be delivered while sharing screen on a Google Meet)

Mandatory requirements (MUST FINISH 😎)

Project Structure:

- Create a project structure that separates components, actions, reducers, and the Redux store.
- Install the necessary packages: react, redux, react-redux, and any other dependencies you may need.

Create Components:

- Create a ProductList component to display a list of products. Each product should have a title, description, price, and an "Add to Cart" button.
- Create a Cart component to display the items added to the cart and their details.

Redux Setup:

- Create a Redux store with appropriate reducers and actions.

Actions and Reducers:

- Define actions like addToCart, removeFromCart, and clearCart.
- Create a cart reducer to manage the state of the shopping cart.
- Define initial states for products and the cart in your reducers.

Product Listing Page:

- Build a product listing page that will open by default on / route, as the application starts up.
- Use FakeStore APIs to invoke API calls in order to fetch the mock products list.
- The page should be responsive and compatible on both web and mobile. Hint: Use GRID view to display the products.
- The page should display 20 products in the initial view port.
- Each row should display 4 on the Web view and 2 products on mobile view.
- Pagination should be handled for up to 100 products and new products list should be appended at the end of the page on scroll.
- Connect the ProductList component / page to the Redux store.
- Dispatch actions when a user adds a product to the cart.
- Ensure the product list updates correctly when items are added or removed from the cart.

Cart Details Page:

- Create a separate route (e.g., /cart) for the cart details.
- Connect the Cart component to the Redux store.
- Display the cart summary in a section that will show the total amount of the cart.
- Display the items in the cart, including their names, quantities, and total cost.
- Implement functionality to remove items and update the quantity from the cart.
- Based on the actions performed on the cart details page the respective cart summary calculations should be updated.

Styling:

- Apply SCSS or use a CSS-in-JS solution to style your components and make the app visually appealing.

Documentation:

- Document your code, especially for actions, reducers, and any complex components.

Optional requirements (INFINITY AND BEYOND 🚀)

Testing (Optional):

- Write unit tests for your components, actions, and reducers using testing libraries like Jest and React Testing Library.

Deployment (Optional):

- Once your application is complete, deploy it to a hosting service like Netlify, Vercel, Heroku, Firebase or AWS.

- It would be ideal if a docker container is built and deployed.
- Share a link to your application in the readme.