E-Commerce Product Listing Page

Overview:

The E-Commerce Product Listing Page project is an interactive web application that fetches and displays a list of products from an API. The page provides users with the ability to search and filter products, and navigate between different parts of the application seamlessly. The project was implemented using React.js, styled with Styled-Components, and optimized for fast loading times and a smooth user experience.

- Repository Name: e-commerce-product-listing
- GitHub Link: E-Commerce Product Listing

Fetching and Displaying Products from an API

In the first part of the project, the focus was on fetching a list of products from a backend API and displaying them on the product listing page. Upon component mounting, the application makes an API request to retrieve product data. The data is then stored locally within the React component's state. Once the data is retrieved, it is dynamically rendered as a list of product items.

The approach to this task ensures the page loads the product data efficiently while handling potential errors during the API call. The products are displayed with their basic details, such as the name, description, and price, in a clean and organized manner. This lays the foundation for further functionalities like search and filtering.

Search and Filter Functionalities

To enhance the user experience, search and filter functionalities were implemented. The search functionality allows users to enter keywords and dynamically filter the displayed products based on matching names or descriptions. This functionality improves usability by narrowing down results according to user input.

The filter functionality was added to enable users to refine their product search based on criteria like price range, product category, or other attributes. Filters are applied dynamically without reloading the page, improving the user’s experience by providing real-time results. The search and filter functionalities rely on state management and re-rendering components based on the user’s interactions.

React Router for Navigation

React Router was used to manage client-side navigation within the application. This enabled the creation of multiple pages such as the product listing page, product detail page, and a cart page, all of which could be accessed without reloading the browser. Navigation between these pages was handled using links and route definitions.

For instance, clicking on a product would redirect the user to the product detail page, where more in-depth information about the product is displayed. Similarly, users can navigate to the cart to view items they have selected. This routing system makes the application feel seamless and responsive, offering an enhanced user experience typical of modern single-page applications (SPAs).

Styled-Components for Styling

The design and visual structure of the page were implemented using Styled-Components, a library that allows for writing CSS directly in JavaScript files. Styled-Components was chosen for its benefits, including the ability to scope styles locally to components, preventing clashes with global CSS.

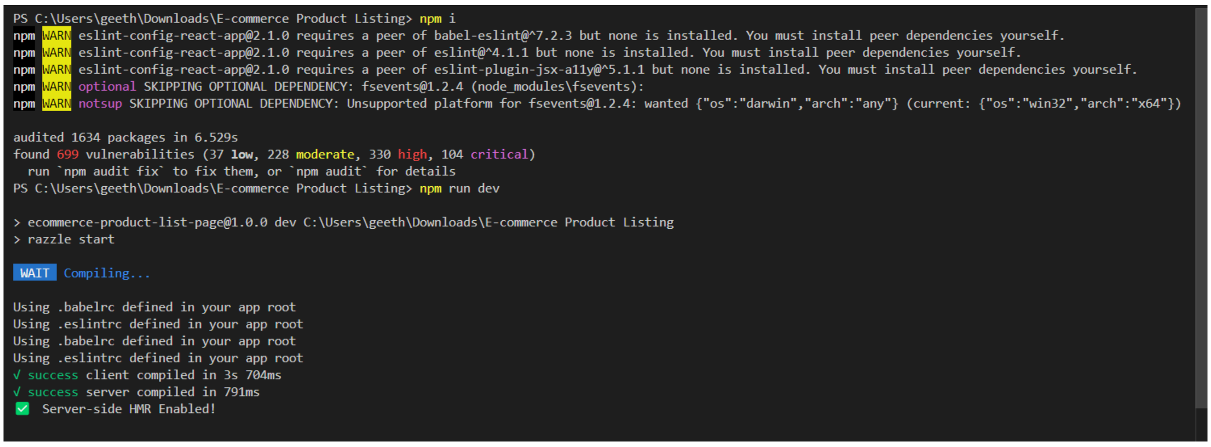
This approach allowed for more maintainable and reusable styles. Each component (e.g., product cards, images, buttons) was given its own styles, ensuring a consistent and visually appealing interface. Furthermore, Styled-Components supports dynamic styling based on props, making it easier to apply conditional styles as the application evolves.

Optimizing Product Image Loading

To improve the performance and loading speed of the product listing page, several techniques were employed to optimize image loading. Lazy loading was utilized to load product images only when they are visible within the user's viewport. This technique reduces the initial load time of the page, as images that are not immediately needed are not downloaded until the user scrolls to them.

Additionally, image file sizes were optimized by serving compressed formats such as WebP, which is more efficient than traditional formats like JPEG or PNG. Responsive images were also used to ensure that different image sizes are served depending on the user's device resolution, thus optimizing bandwidth usage for mobile users without compromising image quality.

Working screenshots:



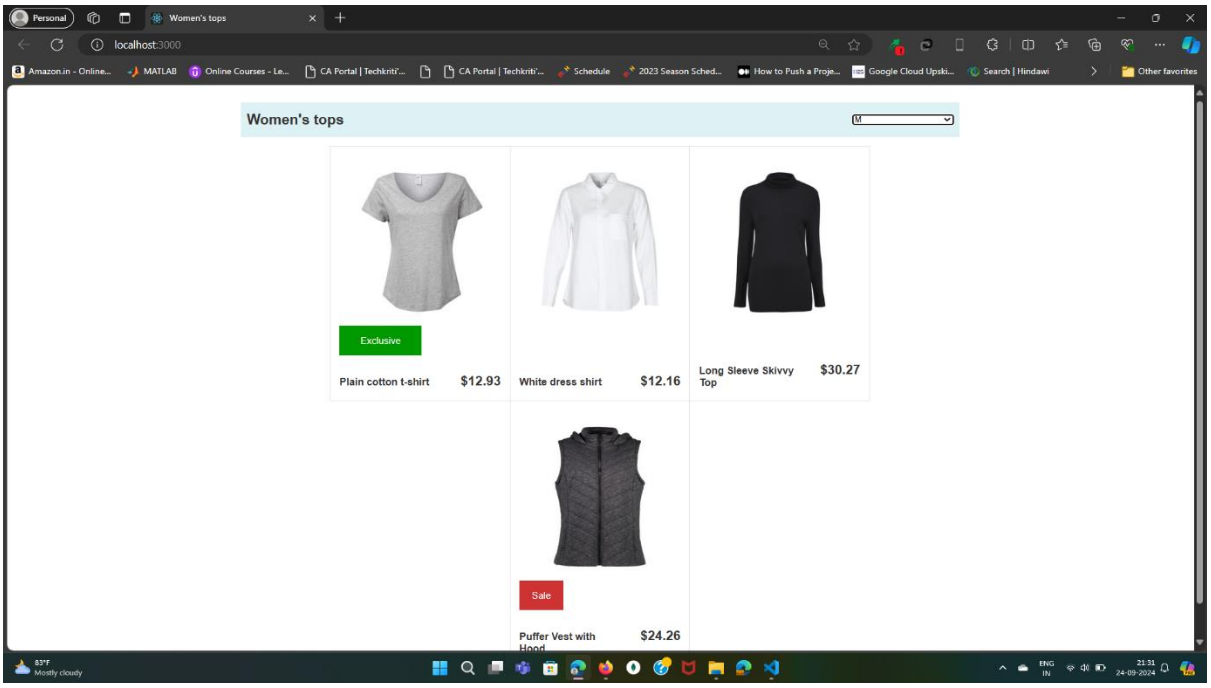
```
PS C:\Users\geeth\Downloads\E-commerce Product Listing> npm i
npm WARN eslint-config-react-app@2.1.0 requires a peer of babel-eslint@^7.2.3 but none is installed. You must install peer dependencies yourself.
npm WARN eslint-config-react-app@2.1.0 requires a peer of eslint@^4.1.1 but none is installed. You must install peer dependencies yourself.
npm WARN eslint-config-react-app@2.1.0 requires a peer of eslint-plugin-jsx-a11y@^5.1.1 but none is installed. You must install peer dependencies yourself.
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@1.2.4 (node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.4: wanted {"os":"darwin","arch":"any"} (current: {"os":"win32","arch":"x64"})

audited 1634 packages in 6.529s
found 699 vulnerabilities (37 low, 228 moderate, 330 high, 104 critical)
  run `npm audit fix` to fix them, or `npm audit` for details
PS C:\Users\geeth\Downloads\E-commerce Product Listing> npm run dev

> ecommerce-product-list-page@1.0.0 dev C:\Users\geeth\Downloads\E-commerce Product Listing
> razzle start

 WAIT  Compiling...

Using .babelrc defined in your app root
Using .eslintrc defined in your app root
Using .babelrc defined in your app root
Using .eslintrc defined in your app root
√ success client compiled in 3s 704ms
√ success server compiled in 791ms
✅  Server-side HMR Enabled!
```



Women's tops

M

Exclusive

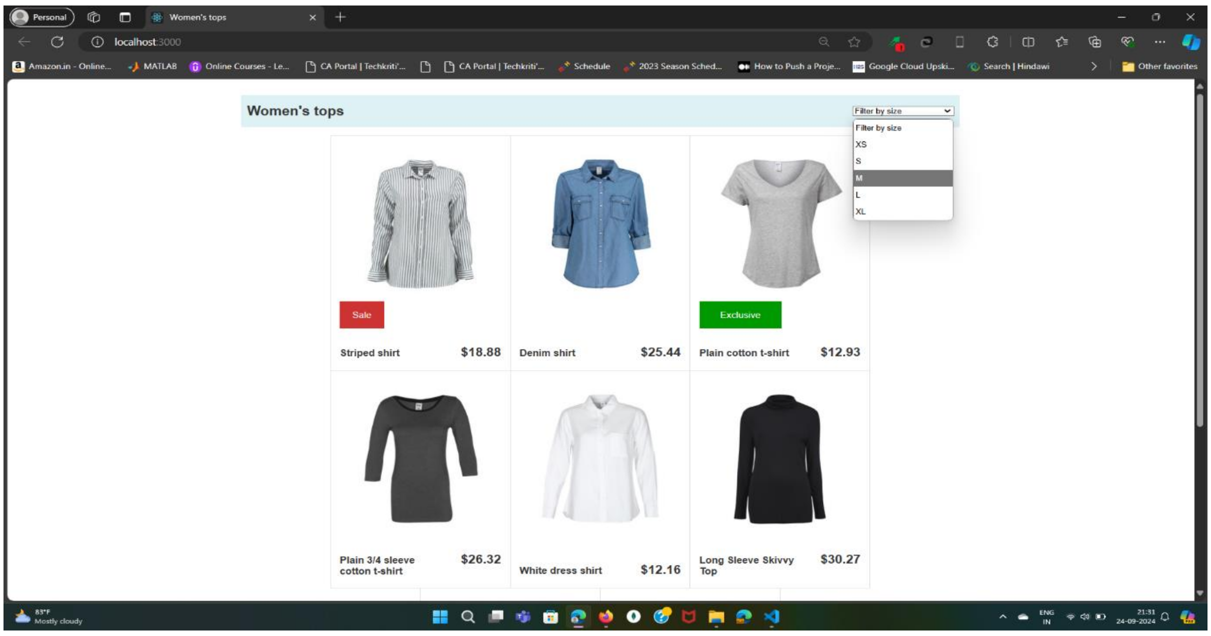
Plain cotton t-shirt $12.93

White dress shirt $12.16

Long Sleeve Skivvy Top $30.27

Sale

Puffer Vest with Hood $24.26



Women's tops

Filter by size

Filter by size
XS
S
M
L
XL

Sale

Striped shirt $18.88

Denim shirt $25.44

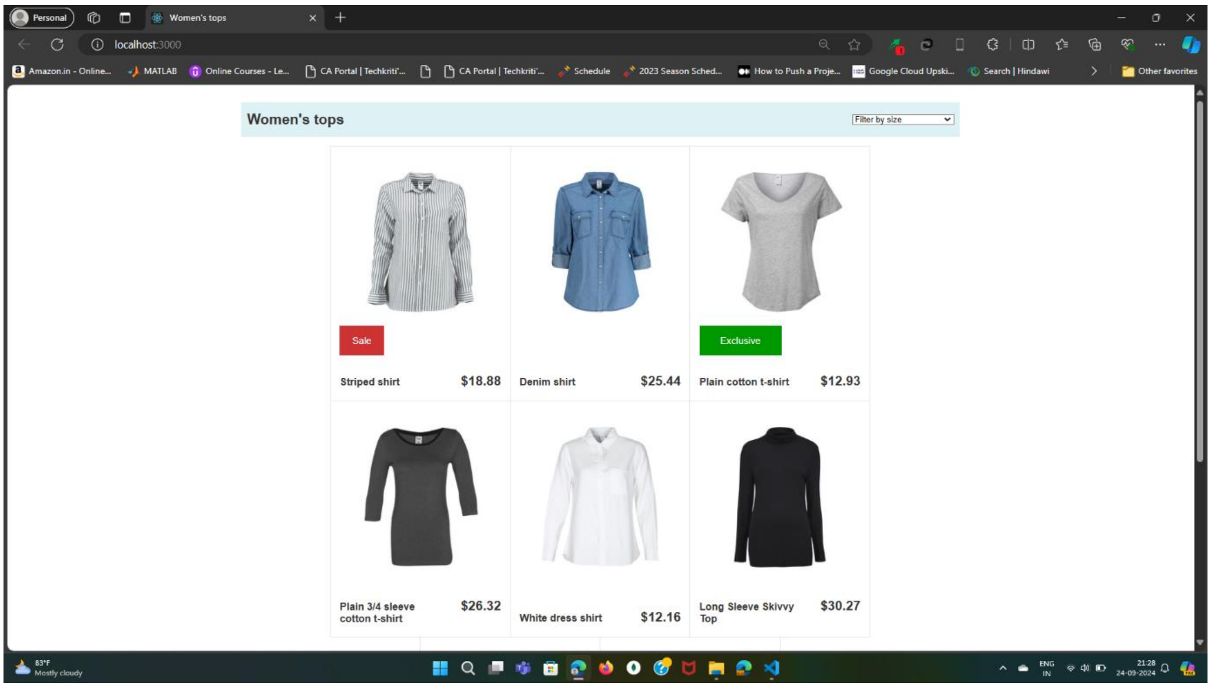
Exclusive

Plain cotton t-shirt $12.93

Plain 3/4 sleeve cotton t-shirt $26.32

White dress shirt $12.16

Long Sleeve Skivvy Top $30.27



Inference:

In developing the E-Commerce Product Listing Page, I implemented a seamless process for fetching and displaying products from an API by using React's state management. The integration of search and filter functionalities allowed users to easily navigate through the product catalog, providing a dynamic and responsive user experience.

I used React Router to enable smooth navigation between pages, ensuring the application functions like a modern single-page app. For styling, Styled-Components allowed me to organize and encapsulate CSS within components, leading to a clean and maintainable codebase.

To optimize performance, I applied techniques like lazy loading, and image compression, improving the overall loading time of product images, especially on devices with lower bandwidth. This not only enhanced user experience but also made the application more efficient.

By combining these techniques, I successfully created a functional and responsive e-commerce interface.