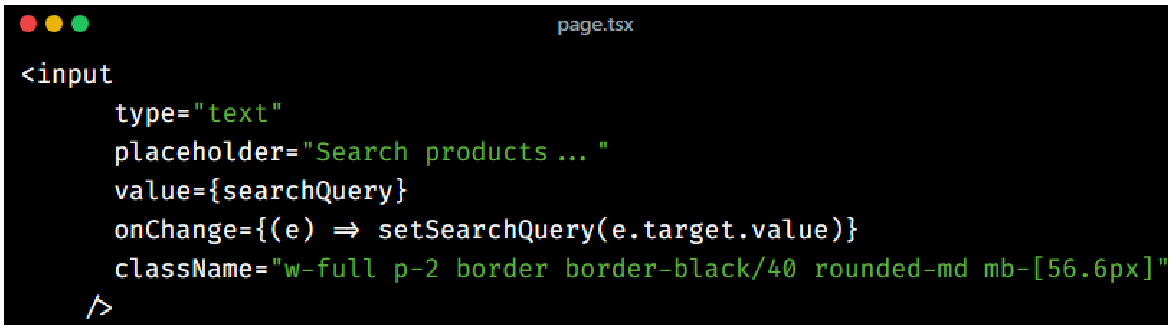
Day 4 - Dynamic Frontend Components - FURNIAURA

Steps Taken to Build and Integrate Components:

1. **Product Mapping Component (Item.tsx)**:
   - Created an Item.tsx component to map through products dynamically.
   - This component is responsible for displaying product details such as name, image, and price.
   - The component is designed to be reusable and is used throughout the website to display various products.

2. **Dynamic Routing for Product Pages**:
   - Used dynamic routing with slugs to create product pages.
   - Implemented the routing via the slug field in the shop/[slug] page.
   - This allows each product to have its own URL based on its slug, ensuring better SEO and a cleaner URL structure.

3. **Search Functionality**:
   - Implemented a search bar that filters products based on their name and tags.
   - Created a function for searching and passed a prop named searchQuery to the Item component.
   - Added an input field for search:
   - 

     page.tsx

     ```
     <input
           type="text"
           placeholder="Search products ... "
           value={searchQuery}
           onChange={(e) => setSearchQuery(e.target.value)}
           className="w-full p-2 border border-black/40 rounded-md mb-[56.6px]"
         />
     ```
   - This input field allows users to search for products by typing the name or tags.

4. **State Management**:
   - Managed the search query state in the parent component (pages) where the Item.tsx components are rendered.
   - Passed the searchQuery as a prop to filter and display only the relevant products.

Challenges Faced and Solutions Implemented:

- **Tag Filtering Issue**:
  - Initially, I thought that the tags were stored as a string, so I used the toLowerCase() method directly on them for comparison.
  - Upon realizing that the tags were actually stored as an array, I adjusted my approach by first converting the array of tags into a string and then applying the toLowerCase() method for case-insensitive comparison.

  Solution:

  - I converted the array of tags into a string format and applied the toLowerCase() method for accurate filtering.

Best Practices Followed During Development:

1. **Component Reusability**:
   - Created the Item.tsx component to handle the rendering of individual product details, ensuring that it can be reused wherever needed.

2. **Dynamic Routing**:
   - Utilized dynamic routing with slugs for individual product pages, which allows for a clean URL structure and easy management of SEO.

3. **Error Handling and UI**:
   - Implemented basic error handling for UI components and logging error messages in the console for debugging.
   - This helps ensure that the user experience remains smooth, even when issues arise.

4. **Efficient Image Handling**:
   - Used Sanity's urlFor() function to handle product images efficiently, ensuring that images are loaded properly.

5. **State and Prop Management**:
   - Properly managed states and passed props between components, making the code more modular and maintainable.

Conclusion:

This documentation summarizes the key steps taken, challenges faced, and best practices implemented during the development of the e-commerce website. The solution integrates dynamic routing, product search, and a reusable component architecture, ensuring a seamless and user-friendly experience.

Site Link: Nofil Store

Prepared by: Muhammad Nofil Shoaib

Roll No. #: 0072576

Slot: Tuesday - Afternon