***HACKATHON DAY TWO***

***TECHNICAL ANALYSIS***

Frontend Of Website

**1. Frontend Architecture**

* **Purpose**: Using Nextjs, That are compatible with Sanity CMS, using sanity that can serve content through its GraphQL or REST API.
  + using next js as strong candidate for server-side rendering (SSR), static site generation (SSG), and incremental static regeneration (ISR), all of which are beneficial
* **Routing**: Utilizing dynamic routing to handle different product categories, product pages, and other parts of the website (like checkout or order history).

Sanity CMS

**2. Content Management (Sanity CMS)**

* **Schema Design**: Sanity uses a schema-driven approach. Ensuring that the content models for products, categories, and variants (like sizes, colors) are well-structured.
  + **Product Schema**: By includding fields for product name, description, price, images, category, size, color, availability, etc.
  + **Category Schema**: This is essential for filtering products.

API integeration

* + 1. **API Integration**: Sanity offers an API that to be used to fetch product and content data efficiently. Alternatively, the REST API can be used depending on requirements.
  + Cache API responses where appropriate to reduce the number of requests and improve load times.
  + Using context API for managing products prices

WEBSITE

* **Product Listings and Filtering**
* **Product Layout**: Making a responsive grid to display products in various categories.Making a flexible layout using tailwind or custom css.
* **Filtering Products**: Implement filtering (by size, color, price, etc.) and sorting (by price, newest, best sellers) using React or a similar frontend library. Fetching the filtered data from Sanity through API queries.
* **Lazy Loading**: Products should be lazily loaded to improve performance, especially for long product lists
* **Product Pages**
* **Image** : Using images, with support for responsive images to ensure the images load fast on all devices.
  + Consider using sanity image through API quary to serve images based on the user's screen and device.
* **Product Details**: Display all relevant product details such as size options, colors, customer reviews, and products.
* **Add to Cart**: A carticon with React’s Context API or snipcart for user to see selected items.
* **Product Variants**: variant selection for different colors and sizes. Use client-side rendering to update product.

**3. Shopping Cart**

* **State Management**: Using libraries like context API to handle the shopping cart and user logins.
* **Cart Page**: Display items in the cart, allowing users to modify quantities, and remove items.
* **Checkout**: Ensure secure and smooth checkout experience with integration to payment providers (like Stripe, PayPal). This should be implemented in a way that protects sensitive user data.

**4. Functionality**

* **Search Engine**: Implement a robust search using Sanity’s own search capabilities. It should allow for fuzzy search and handle typo tolerance.
* **Search Filters**: Enable filtering options such as categories, size, and price range to refine search results.

**5. User Authentication**

* **Login & Registration**: Handle user accounts, order history, and saved carts. Authentication can be integrated using APIs, or custom authentication system.
* **Social Logins**: Allowing users to sign in via Google, Facebook, or other social login providers.

**6. Performance**

* **Lazy Loading**: For images, product details, and large components, using lazy loading.
* **Cart & Checkout**: Provide an easy-to-use and visually clear cart with smooth transitions and feedback when adding/removing items.

APIs

**API QUERY**

**CATEGORIES**

**SANITY**

**SUBCATEGORY**

LOGIN

**DATA**

**RETURN**

ORDER

VARIANTS

PRODUCTS

MENS

CART

**STORED**

DELIEVERY

SHIPMENT

TRACKING

CHILD

WOMENS