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

3. 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

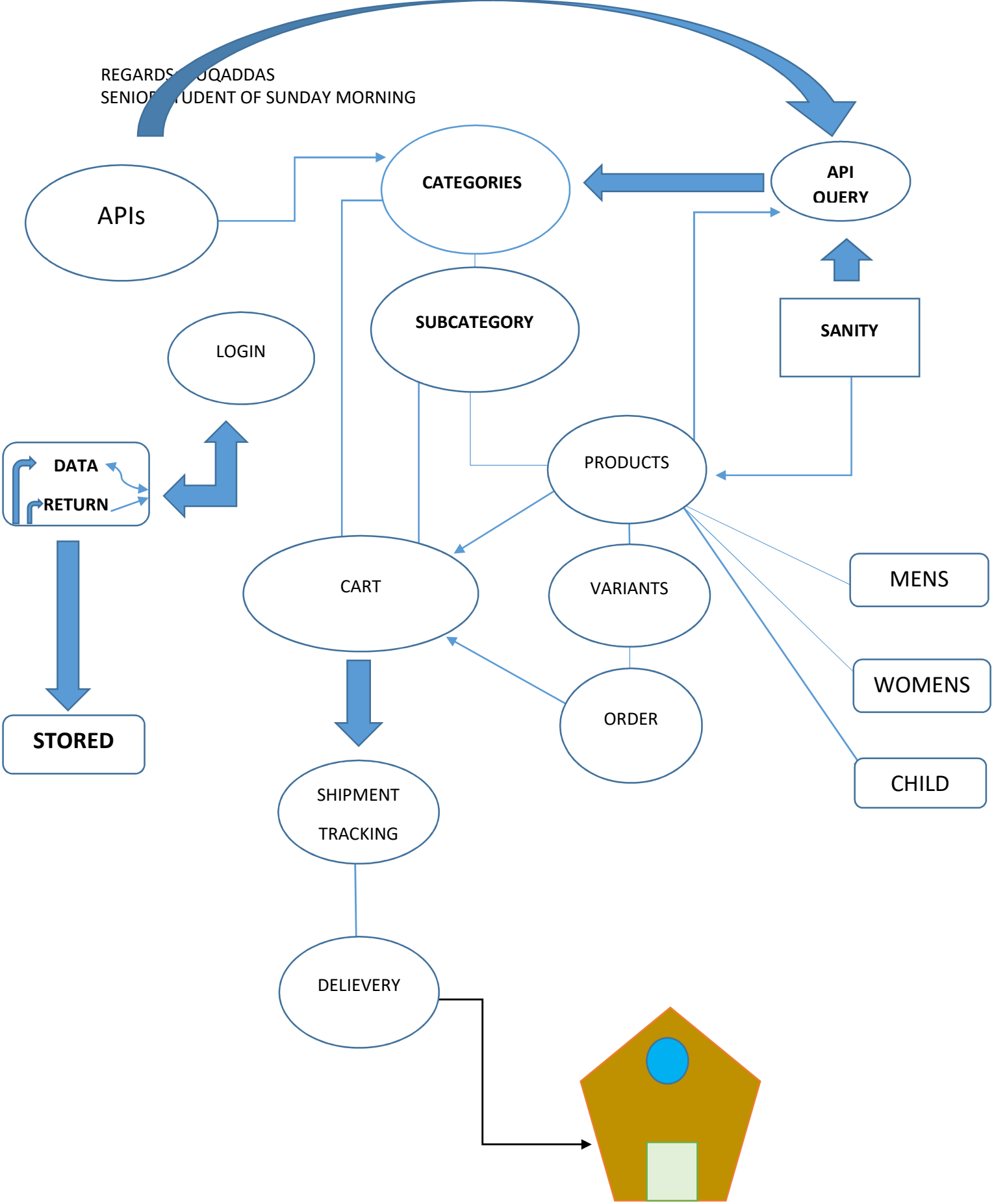
- **LOGIN & REGISTRATION**: Handle user accounts, order history, and saved carts. Authentication can be integrated using APIs, or custom authentication system.
- **SOCIAL LOGINS**: Allowing users to sign in via Google, Facebook, or other social login providers.

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.

REGARDS: UQADDAS
SENIOR STUDENT OF SUNDAY MORNING



REGARDS:MUQADDAS
SENIOR STUDENT OF SUNDAY MORNING