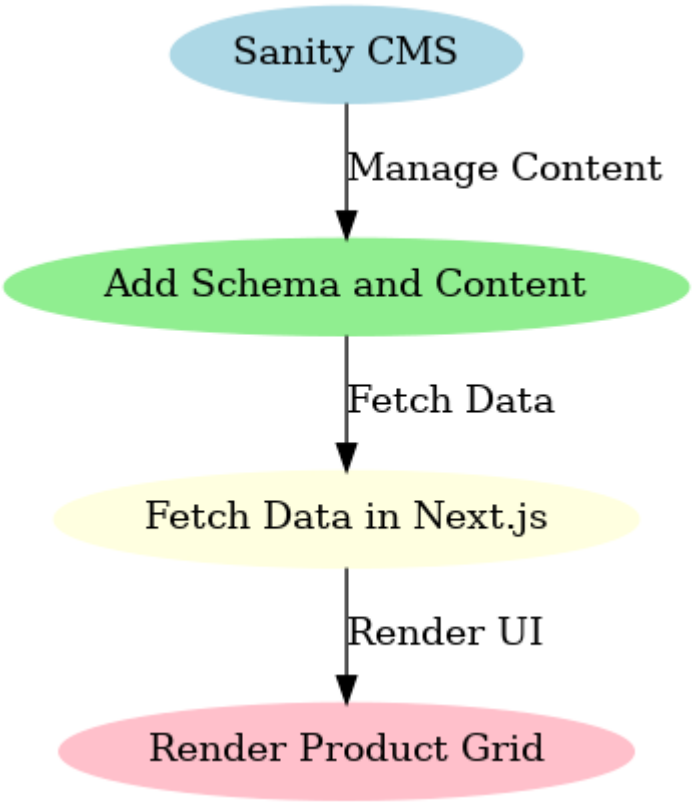
DAY 3

Fetching and Displaying Content with Sanity and Next.js



1. Schema in Sanity:
I started by creating a custom schema for the product section in Sanity CMS. This schema, called cardsSection, defines the structure of the content I want to work with. It includes fields for a

banner image, product images, headings, and prices. The schema ensures that the content is well-structured and can be easily managed through the Sanity Studio interface.

2. Content Creation:
After setting up the schema, I added product data directly into Sanity Studio. Using the defined schema, I entered product details such as the product images, titles, and prices. This allows content managers to easily update or add new product information without touching the code.

3. GROQ Query:
To fetch the product data from Sanity, I used GROQ (Graph-Relational Object Queries), which is a query language designed specifically for querying Sanity's content. In the shopproduct.tsx file, I wrote a GROQ query that pulls the relevant data—like images, headings, and prices—associated with the products stored in Sanity.

4. Rendering in Next.js:
Once the data is fetched using the GROQ query, I mapped the returned data in the shopproduct.tsx component. This allowed me to dynamically render product cards on the page. Each card contains an image, title, and price, based on the fetched data.
5. Responsive Display:
To ensure a good user experience across all devices, I used Tailwind CSS to create a responsive grid layout for the product cards. This ensures that the cards automatically adjust their size and layout depending on the screen size, providing an optimal display on both mobile and desktop devices.