Day 3: API Integration and Data Setup for Food Tunk Website

Overview

Day 3 focused on integrating Sanity CMS with the Food Tunk website, creating structured schemas, fetching data using GROQ queries, and rendering it dynamically in a responsive Next.js application.

Key Highlights

1. Custom Schema Setup

- **Schema Creation**: Defined a schema for "Product Item" to structure data (e.g., menu items, specials, and location details).
- **Validation Rules**: Ensured data integrity with custom validations, like requiring positive prices and non-empty titles/descriptions.

2. Sanity API Integration

- **Configuration**: Connected to Sanity using project ID, dataset, and API token.
- **Environment Variables**: Used .env file for secure storage of sensitive information.

3. Data Fetching with GROQ

- **Querying**: Used GROQ queries to fetch structured data (e.g., menu categories, prices, descriptions, images).
- **Dynamic Updates**: Data is pulled live from Sanity, avoiding database insertion.

4. Frontend Rendering

- **Server-Side Rendering (SSR)**: Fetched data in getServerSideProps for faster load times and improved SEO.
- **Responsive Design**: Rendered mobile-friendly menu layouts dynamically using React's .map() method.
- **Dynamic Routing**: Added links for individual product detail pages.

5. Product Card Component

- **Reusable Design**: A dynamic, styled component to display product details like title, price, description, and image.

- **Interactivity**: Includes buttons for actions like adding items to the cart.
- **Optimized Features**: Uses libraries like next/image for efficient image rendering.

6. Environment and Security

- **Environment Variables**:
  - SANITY_PROJECT_ID, SANITY_DATASET, SANITY_API_TOKEN securely stored in .env file.
- **Security Measures**: Ensured API tokens are not exposed on the frontend.

Achievements

- Created a dynamic schema for product items in Sanity.
- Successfully integrated data fetching using GROQ queries.
- Built responsive and reusable frontend components.
- Secured sensitive configurations using environment variables.

Conclusion

The setup on Day 3 established a robust backend integration and dynamic frontend rendering for the Food Tunk website. This foundation supports real-time updates, scalability, and future feature enhancements.