Hackathon 3 Day 2:

PLANNING THE TECHNICAL FOUNDATION

Project Overview:

You are building an e-commerce platform focused on **clothing**, where customers can browse a variety of categories (like men's, women's, accessories), place orders, and track their shipments. You want a modern, responsive site with features like a smooth shopping experience, size options, and payment integrations.

Frontend:

- **Tech Stack:**
  - **HTML** for the basic structure.
  - **Next.js** for server-side rendering, optimized performance, and dynamic content fetching.
  - **Tailwind CSS** for styling (modern, mobile-first, and clean).
- **Pages & Components:**
  1. **Landing Page**: Featured collections (like summer sale, new arrivals, or trending clothing items). Use high-quality images to showcase fashion.
  2. **Product Listings**: Categories such as "Men", "Women", "Accessories", "Footwear", etc., with filters like size, color, and price range.
  3. **Product Details Page**: Detailed product page with size guides, fabric details, product descriptions, color options, and a size selector.
  4. **Shopping Cart**: Option to view added items, adjust quantities, and proceed to checkout.
  5. **Checkout Page**: Where users enter shipping and payment info, apply promo codes, and place orders.
  6. **Order Confirmation Page**: Post-purchase confirmation, including order summary, estimated delivery time, and contact support options.
  7. **User Account Page**: Customer accounts to view order history, save favorite products, manage addresses.
  8. **Order Tracking Page**: Customers can track real-time order status and delivery tracking.
- **Frontend Features:**
  - **Responsive Design**: Tailwind CSS will ensure the website is mobile-friendly and works well on all devices.
  - **Smooth Animations**: Animations for adding items to the cart, hover effects on product images, etc.
  - **SEO Optimization**: Optimize your product pages using Next.js' built-in SEO capabilities.
  - **State Management**: Use React Context or Redux for managing the shopping cart, user info, and session management.

Backend:

- **Tech Stack:**
  - **Sanity CMS**: Headless CMS for managing product data (including images, sizes, color variants, etc.), user data, and orders.
  - **Node.js**: Used for backend logic to handle order placement, payment integration, and customer management.
  - **Database**: Sanity stores content but you might need a separate DB like MongoDB or PostgreSQL for transactional data.
- **Backend Features:**
  - **Product Management**: Manage categories, products, and sizes via Sanity CMS. Ensure you have a schema for clothing items that include size charts, prices, colors, and images.
  - **Order Management**: Store and track customer orders—manage order status (pending, shipped, delivered).
  - **User Data**: Collect customer information, such as email, shipping address, and order history.

APIs:

- **Third-party API Integrations:**
  1. **Shipping API**: Integrate with services like FedEx, DHL, or local courier services to provide shipping options and real-time tracking.
  2. **Payment API**: Integrate payment gateways like **Any Account**, **JazzCAsh**, or **easyPaisa** for payment processing.
  3. **Order Tracking API**: Let customers track their orders through an external order tracking API.
- **API Endpoints:**
  - **/Api/products**: Fetch clothing products and categories from Sanity CMS.
  - **/Api/cart**: Handle adding/removing items to/from the shopping cart.
  - **/Api/order**: Handle order placement, updates, and confirmation.
  - **/api/user**: User authentication (sign up, login, profile management).
  - **/api/payment**: Process payments using a third-party payment gateway.
  - **/api/shipping**: Handle shipping details and real-time tracking.

Steps to Build:

1 Set up Next.js Project:

- Run npx create-next-app to create the project.

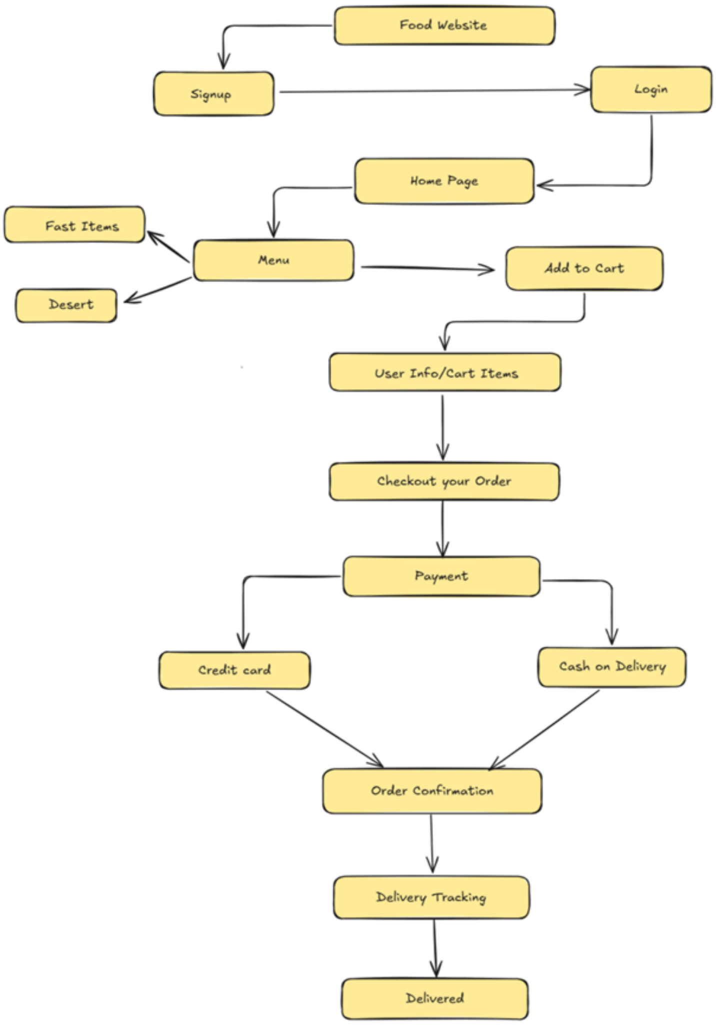
2 Sanity CMS Setup:

- Create a **Sanity CMS** project and define schemas for products, categories, and orders.
- Upload product data like clothing items (name, description, price, sizes, colors) and images into Sanity.
- Fetch product data from Sanity using the Sanity Client.

3 API's:

- Integration with third party APIs for
- Shipping APIs
- Payment APIs
- Order tracking APIs

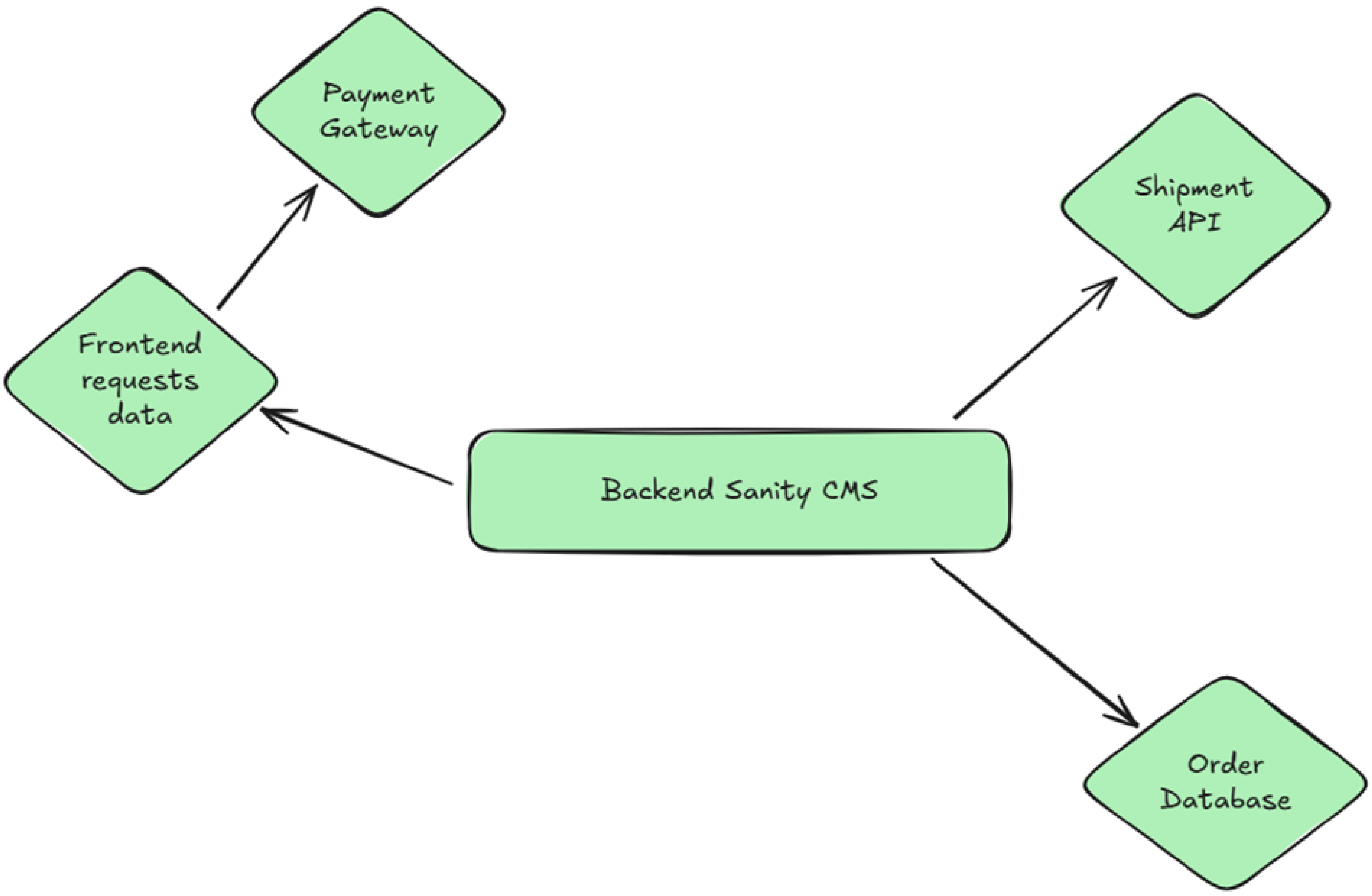
Frontened Requirement Flow chart



Frontend Requirements:

- **Mobile-friendly ,responsive design.**
- **Pages: Home, Product listing, product Details, checkout, Order Confirmation**

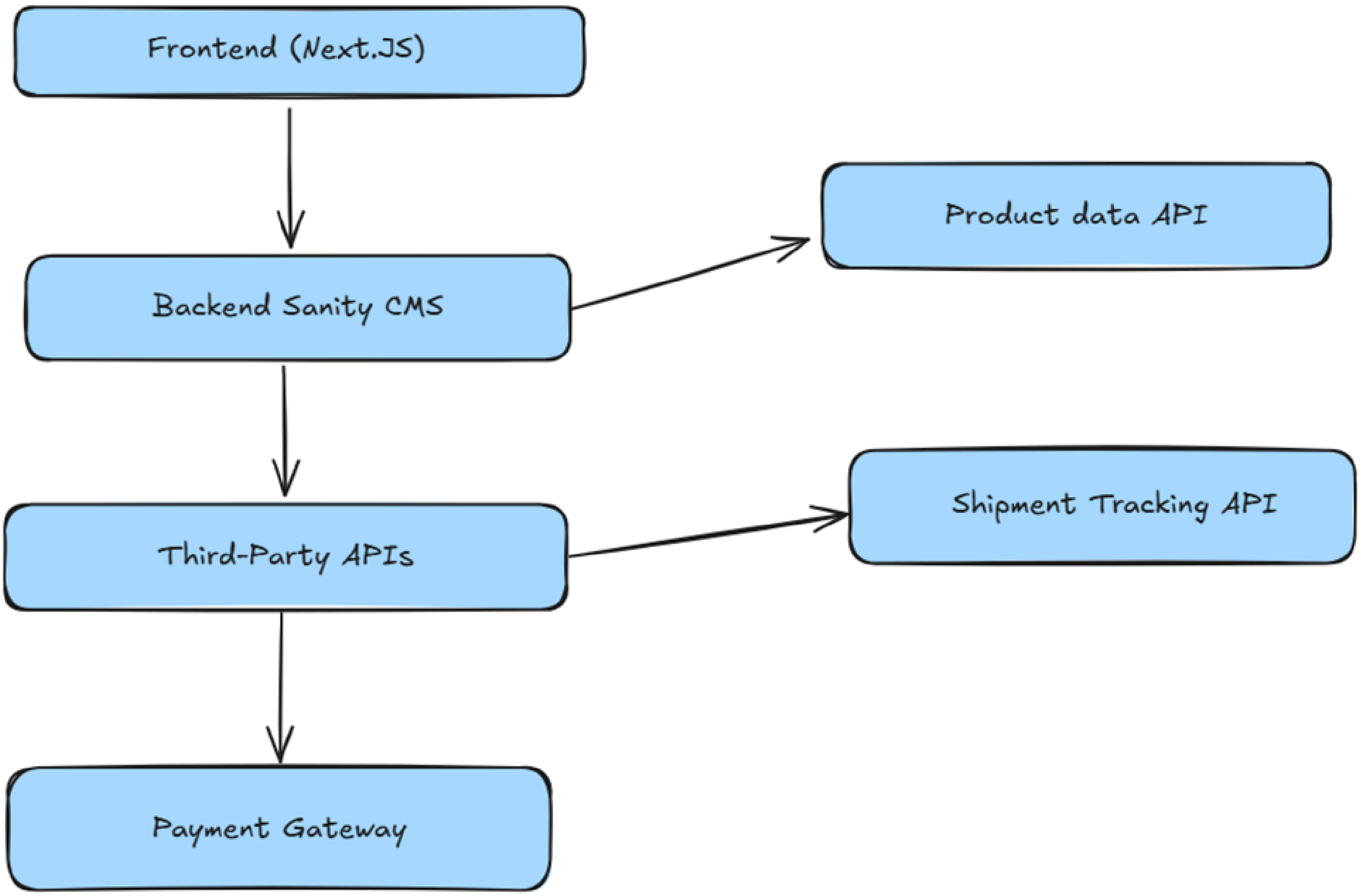
Backend Requirement Flow Chart:



Backend Requirements:

- **Manage products, Customers and Orders using sanity CMS**
- **Real-time inventory updates and product management**
- **Customizable schemas for data organization**

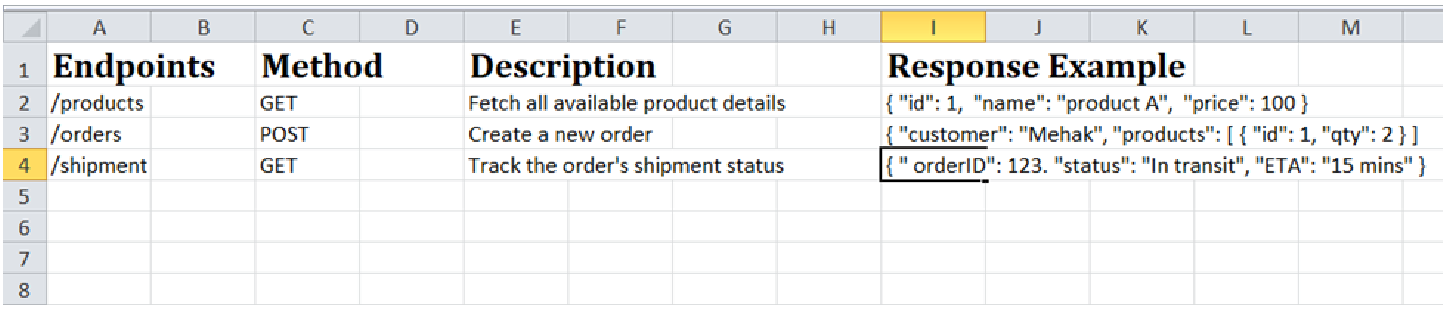
Design System Architecture:



Component

- The Frontend connects to Sanity CMS for Fetching and Managing data
- APIs interact with both the CMS and external services for tasks such as payment and shipping

API's Endpoints:



Endpoints	Method	Description	Response Example
/products	GET	Fetch all available product details	{ "id": 1, "name": "product A", "price": 100 }
/orders	POST	Create a new order	{ "customer": "Mehak", "products": [{ "id": 1, "qty": 2 }]
/shipment	GET	Track the order's shipment status	{ " orderID": 123. "status": "In transit", "ETA": "15 mins" }