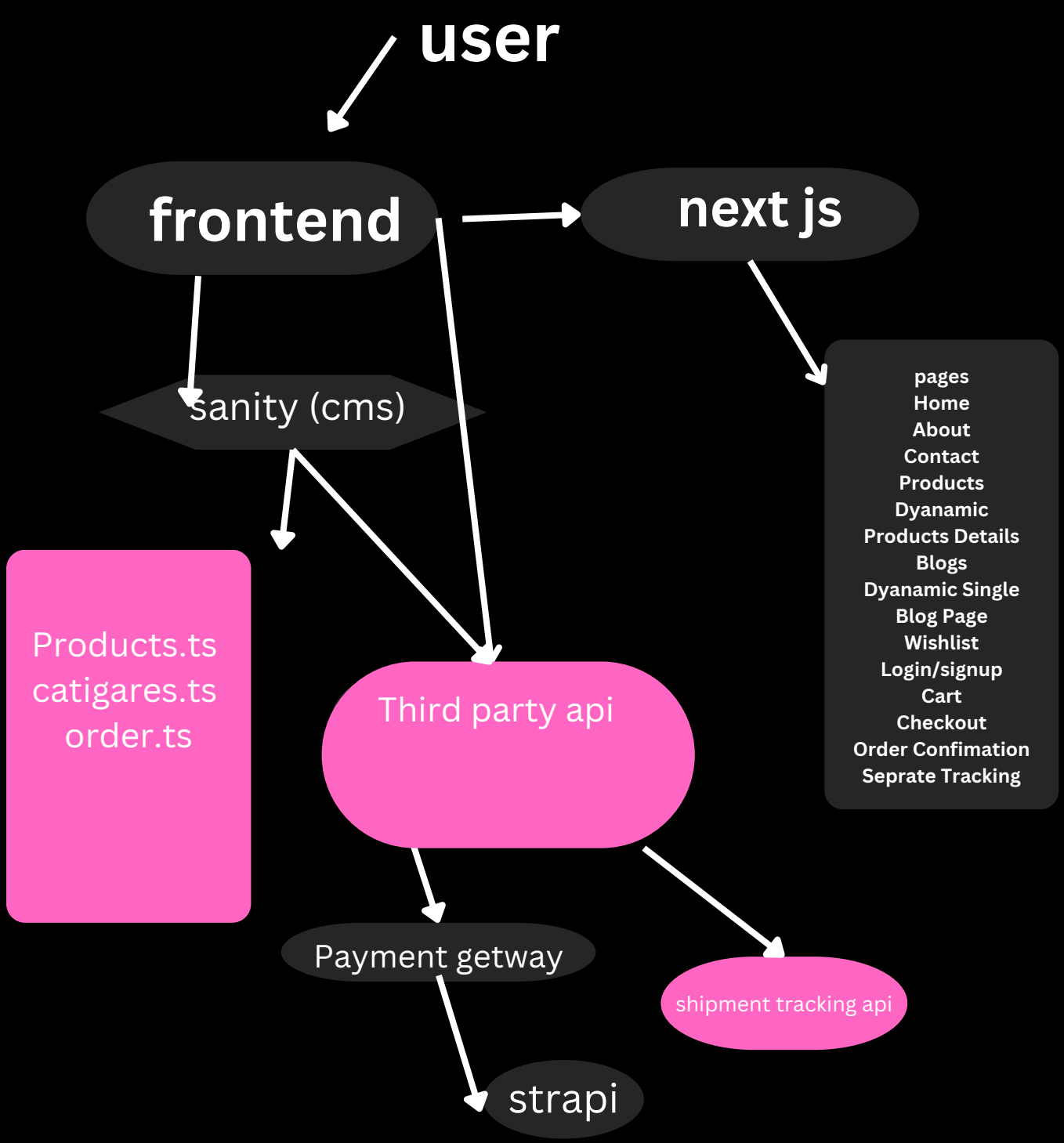
PLANNING THE TECHNICAL FOUNDATION

Hackthoone Day 2

General-E-Commerce Website

Project Overview : Hekto Website

Design System Architecture



1. Frontend (User Interaction Layer)

Framework: Next.js

Purpose: Handles the user interface, displaying dynamic and responsive pages.

Home: Main landing page.

About: Provides an overview of the business/website.

Contact: Allows users to send queries or feedback.

Product: Displays product categories and details.

Dynamic Product Details: A page showing details for each product (using slug-based routing).

Shop Grid: Displays multiple products in a grid format.

Wishlist: Lets users save their favorite products.

Cart: Summarizes selected products and checkout options.

Order Confirmation: Final page after payment is completed.

Shipment Tracking: Page to track the delivery status of orders.

Technologies:

Tailwind CSS: For responsive design and grid-based layouts.

React Components: Dynamic rendering is done using reusable components like Products.ts, categories.ts, and order.ts.

Data Fetching: Data is fetched from Sanity or backend via APIs (e.g., getServerSideProps, useEffect hooks).

2. Backend (Data Handling Layer)

Sanity CMS

Purpose: Manages the database for products, categories, and orders.

Data Types:

Products

Fields: _id, name, price, description, slug, image

Categories

Fields: _id, name, products.

Orders

Fields: _id, user, product, status, totalAmount

Integration: Data is fetched through frontend APIs and connected to the Sanity schema.

Strapi

Purpose: It handles additional backend logic that goes beyond Sanity, such as user authentication and custom APIs.

Functionality

It saves payment data (e.g., fetching data from the Stripe webhook) and manages shipment tracking status

Third-Party APIs (External Systems Layer)

Payment Gateway:

Purpose: Enables secure and smooth transactions.

APIs Used: Stripe, Razorpay, or PayPal.

Workflow:

The user enters their payment details on the frontend.

Backend processes these details via the payment gateway API.

Successful transaction status is stored in Sanity or Strapi.

Shipment Tracking API:

Purpose: Provides real-time delivery tracking for orders.

Workflow:

After an order is placed, the backend generates a tracking ID.

Real-time delivery status is fetched via the shipment API and displayed on the frontend.

4. Integration & Workflow

User Interaction:

Users interact with the website via the Next.js frontend.

They browse products, add items to the cart, and proceed to checkout.

Backend Communication:

The frontend fetches data from Sanity CMS and Strapi.

It integrates with external services (payment gateway and shipment tracking) via APIs.

Data Display:

The frontend dynamically displays products, categories, and order details.

Payment and shipment details are updated in real time.

Order Completion:

After successful payment, users are redirected to the order confirmation page.

Delivery status is tracked via the shipment API.

Diagram Analysis (Mapping)

Component	Purpose
Next.js (Frontend)	Handles dynamic routing, responsive UI, and user interaction.
Sanity CMS	Stores and manages data for products, categories, and orders.
Third-Party APIs	Handles payments and shipment tracking.
Strapi	Manages backend logic and integrates APIs beyond Sanity's capabilities.