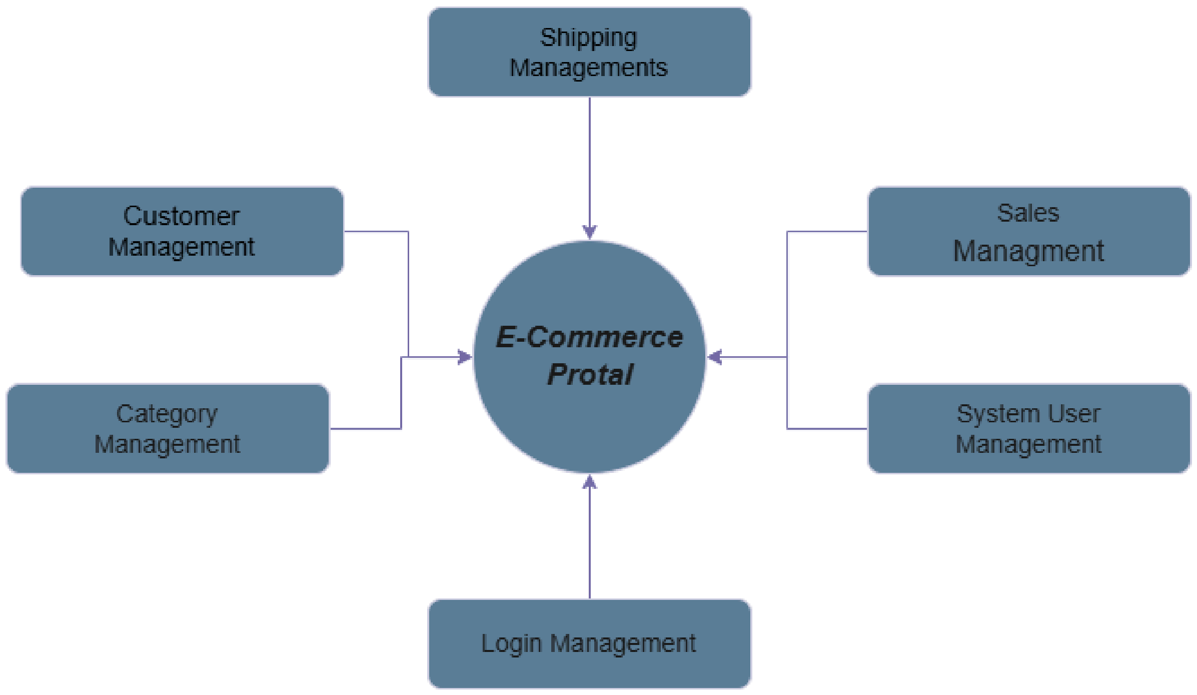
HACKATHON-03
DAY-02

Planning the Technical Foundation



1. Define Technical Requirements

01: Frontend Requirements:

Framework and Styling
We're using Next.js as the backbone of our application to ensure fast, SEO-friendly, and dynamic rendering. For styling, Tailwind CSS gives us a flexible, utility-first approach to create sleek and responsive designs quickly.

Routing and Navigation
With Next.js's file-based routing, creating static and dynamic pages is effortless. Features like nested routes and dynamic URLs let us design user-friendly navigation.

Optimized Rendering
Critical pages like the homepage and product pages will use server-side rendering (SSR) for speed and SEO. For static pages like FAQs, we'll use static site generation (SSG), while incremental static regeneration (ISR) ensures content stays fresh without frequent rebuilds.

02: Sanity CMS as Backend:

Sanity CMS will act as the central hub for managing all the content on the platform, including products, categories, blogs, and promotional banners.

Its **structured content model** ensures flexibility in defining and organizing various data types.

Sanity offers **real-time collaboration**, allowing multiple users to edit content simultaneously.

Changes are instantly reflected on the frontend, ensuring up-to-date content for users.

03: Third-Party APIs:

Payment Gateways: Handle secure transactions and multiple payment options. Stripe

Shipping and Logistics: Provide real-time shipping rates, tracking, and label generation. ShipEngine

Cart and Checkout: Snipcart is a powerful and easy-to-integrate solution for adding cart and checkout functionality to any website or e-commerce platform.

Inventory Management: Tracks product stock levels directly within Snipcart. Updates inventory in real-time as customers complete purchases.

Design System Architecture

[frontend (next.js)]

| | |

|

[Sanity CMS]   [Third-Party APIs]   [Payment Gateway]

[Product Data (API)] [Shipment Tracking(API)] [stripe(payment)]

Detailed Key Workflows

User Registration:

A new user signs up on the platform.

The user’s data (e.g., name, email, password) is securely stored in Sanity CMS.

A confirmation email is sent to the user to verify their account.

Product Browsing:

Users explore product categories and use filters to refine their search.

The frontend fetches product data dynamically via the Sanity API.

Products, along with images, descriptions, and prices, are displayed seamlessly to the user.

Order Placement:

The user selects items to add to their shopping cart.

At checkout, payment details are entered, and the order is processed securely.

Order details, including products, customer info, and payment status, are saved in Sanity CMS for tracking.

Shipment Tracking:

The system fetches order status updates (e.g., "-ShipEngine," "In Transit") using a third-party shipping API.

Real-time shipment tracking details are displayed to the user on their order history or tracking page.

Payment Processing:

The selected payment method is securely processed through a **third-party payment gateway** (e.g., Stripe).

Fraud prevention and validation checks are performed during the transaction.

Plan API Requirements:

1: All products fetch:

Endpoint Name:/Products

Method: Get

Description: This endpoint provides real-time updates on the delivery status of perishable items, ensuring that customers can track the delivery progress for time-sensitive orders.

Response Example

```
[{
  "orderId": 123,
  "name": "Iphone-15",
  "image": "image(url)",
  "Price": 20000,
}]
```

02: Create a new order

Endpoint Name:/order

Method: post

Description: This endpoint is used to create a new order in the system and store the relevant details in Sanity

Payload Example:

```
{
  "Customer Info": {
    "name": "John Doe",
    "email": "john@example.com",
    "address": "123 Main St, City, Country",
    "phone": "123-456-7890"
  },
  "products": [
    {
      "productid": 1,
      "ProductName": " Headphones",
      "quantity": 2,
      "price": 200
    },
    {
      "productid": 2,
      "ProductName": "Bluetooth Speaker",
      "quantity": 1,
      "price": 5000
    }
  ],
  "Payment Status": "Paid"
}
```

Response Example:

```
{“order Id”:” ORD111”,” status”: “Order Created”}
```

03: Tracking and shipment

Endpoint Name:/Shipment

Method: GET

Description:
This endpoint allows users to track the status of their orders by fetching real-time shipment updates via a third-party API. It provides details such as the shipment ID, order ID, status, and the expected delivery date.

Response Example:

```
{
 "shipmentId": "ABC123456",
 "orderId": 123,
 "status": "Out for Delivery",
 "expectedDeliveryDate": "2025-01-18"
 }
```

Sanity Schema Example:

Product Schema:

```
export default {
  name: 'product',
  type: 'document',
  title: 'Product',
fields: [
    {
    name: 'name',
    type: 'string',
    title: 'Product Name',
    description: 'The name of the product.',
    validation: Rule => Rule.required().min(3).max(100)
    },
    {
    name: 'price',
    type: 'number',
    title: 'Price',
    description: 'The price of the product in USD.',
    validation: Rule => Rule.required().min(0)
    },
    {
    name: 'stock',
    type: 'number',
    title: 'Stock Level',
    description: 'The available stock of this product.',
    validation: Rule => Rule.required().min(0)
    },
    {
    name: 'description',
    type: 'text',
    title: 'Product Description',
    description: 'A detailed description of the product.',
    validation: Rule => Rule.optional().max(500)
    },
    {
    name: 'image',
    type: 'image',
    title: 'Product Image',
    description: 'An image representing the product.',
    options: {
    hotspot: true
    }
    },
            {
            name: 'category',
            type: 'string',
            title: 'Product Category',
            description: 'The category under which the product falls (e.g.,
    Electronics, Fashion, etc.).',
            validation: Rule => Rule.required().min(3)
            }
      ]
    };
```