HACATHON DAY 2

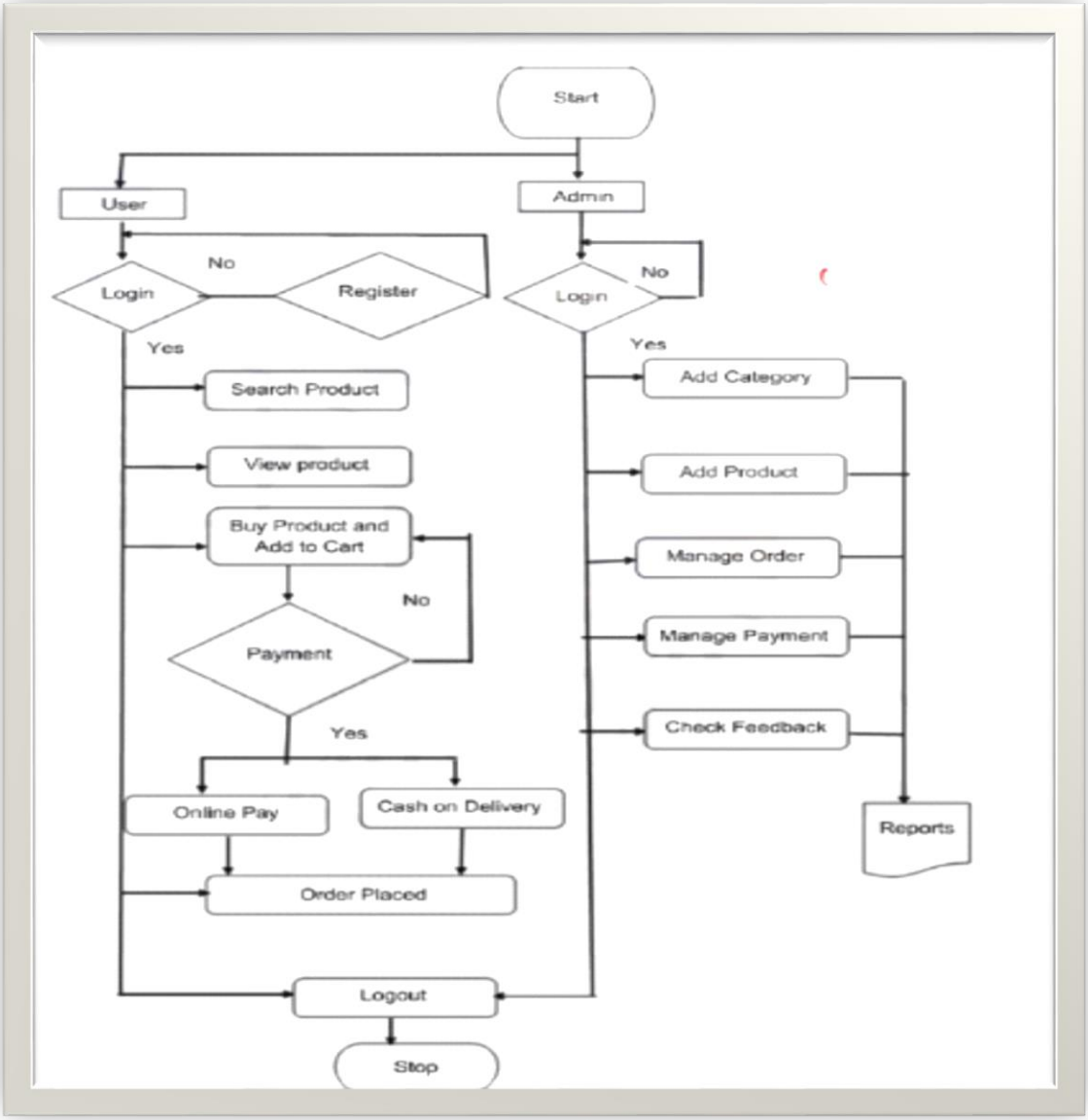
Planning The Technical Foundation

Work Flow Diagram

- **Work Flow Diagram Visual Representaion**

[Visit Homepage] ⇒[Browses Product] ⇒ [View Product Detail Page] ⇒ [Add Product To Cart] ⇒ [Proceeds To Checkout] ⇒>>> >> [Complete Profile and Payment Details] ⇒ [Order Configuration] ⇒>> >> [Tracks Shipment] ⇒>> [Receives Delivery at Home]

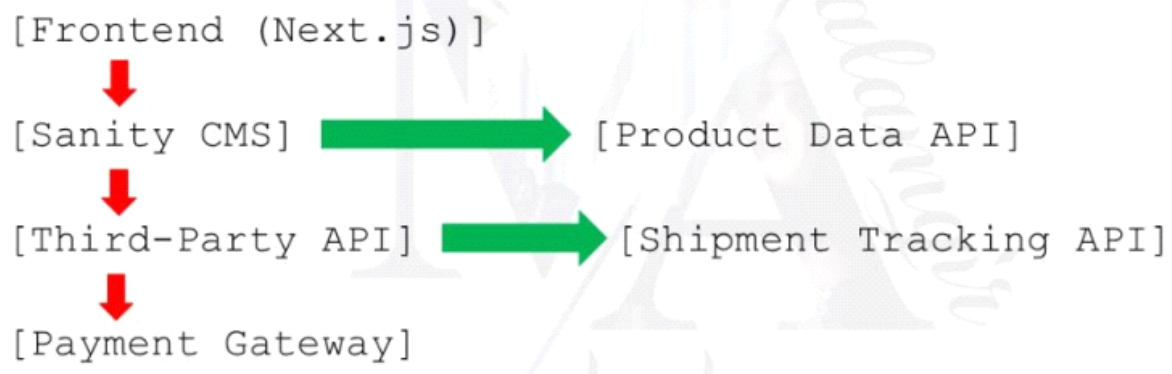
Frontend Requirements Flow Chart



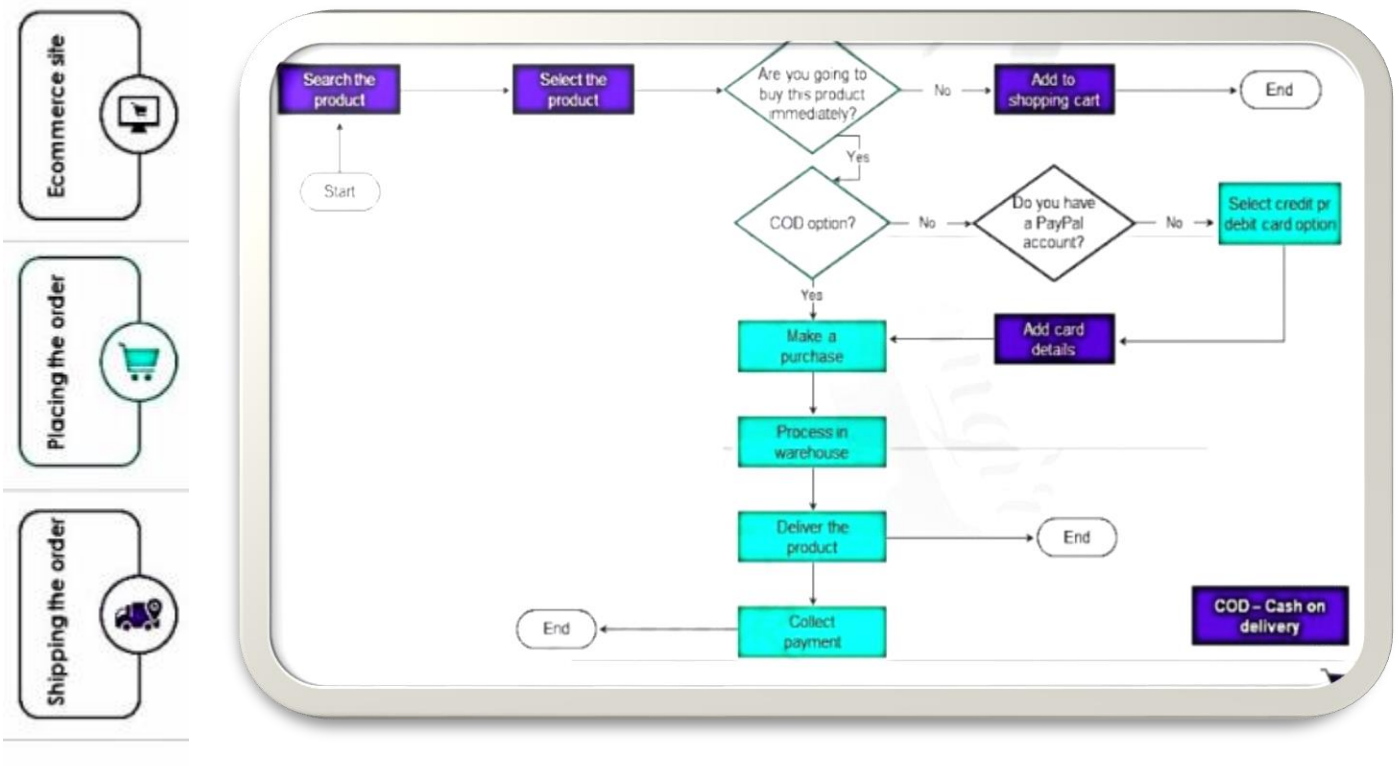
Design System Architecture

System Architecture Overview

Diagram:



System Architecture



Components and Roles:

Frontend (Next.js):

1. Displays the user interface for browsing products, managing the cart, and placing orders.
2. Handles user interactions and communicates with backend services via APIs.

Sanity CMS:

1. Acts as the primary backend to manage product data, customer details, and order records.
2. Provides APIs for the frontend to fetch and update data.

Product Data API:

Provides endpoints to fetch product listings, details, and inventory status.

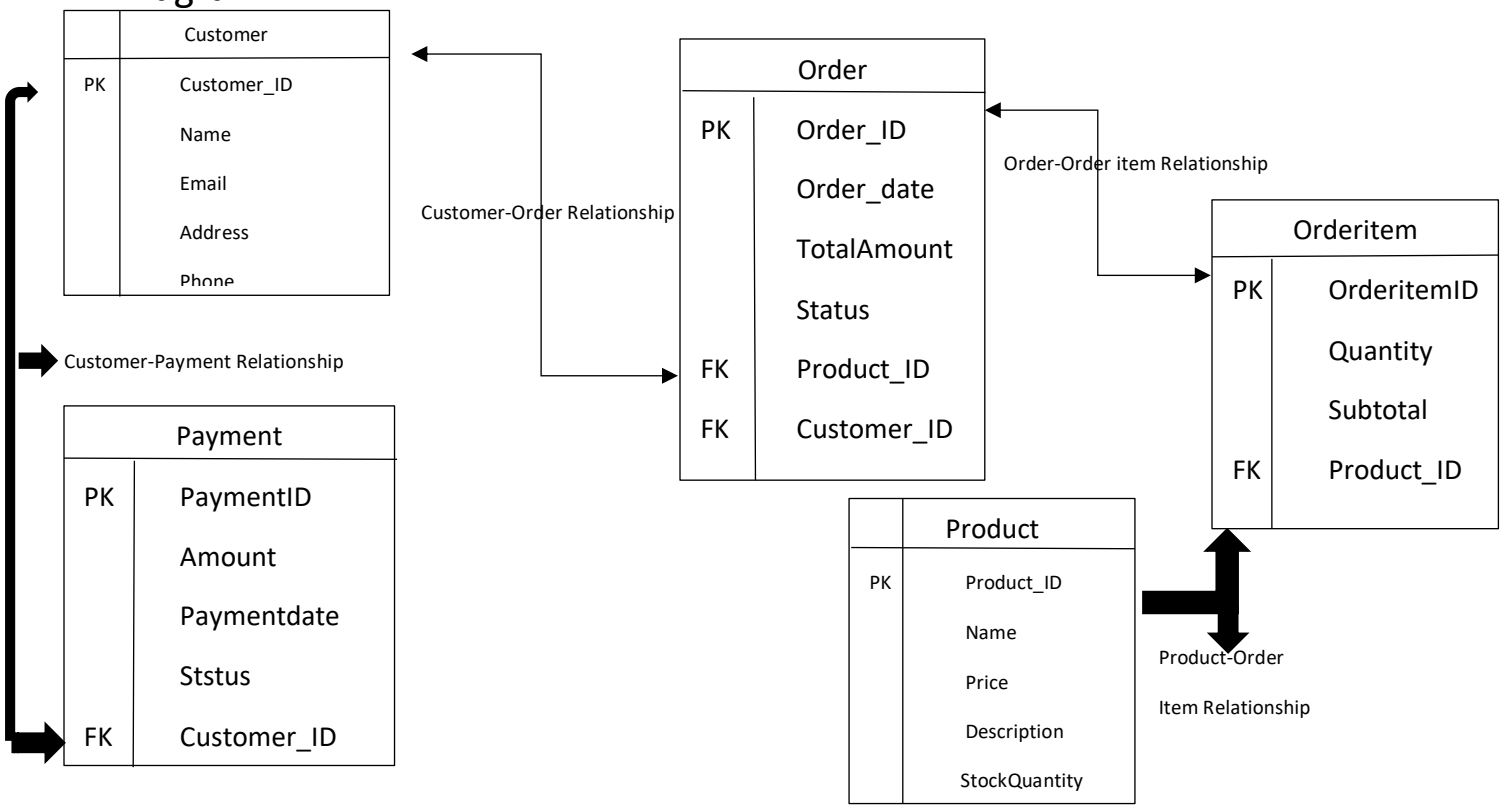
Third-Party APIs:

Integrates services like shipment tracking and payment processing.

Payment Gateway:

Processes user payments securely and provides transaction confirmation.

ER Diagram:



API Endpoint.xisx

Endpoint	Method	Description	Parameters	Response Example
/api/bikes	GET	Fetch all food items	None	{ id: !, name: “Honda 125”}
/api/bikes/:id	GET	Fetch a single bike item	Id(Path)	{id: 1, name “Honda125”}
/api/bikes	POST	Add a new bike item	Name,price, category (Body)	{success:true,id: 5}
/api/bikes/:id	PUT	Update a bike item	Id (Path), name, Price (Body)	{success: true}
/api/bike/:id	DELETE	Delete a bike item	Id (Path)	{success: true}
/api/categories	GET	Fetch all bike Categories’	None	{categories: [“CD 70”]}

SanitySchema.js

```
export default {
  // Define the document type and its name
  name: 'product',
  type: 'document',
  title: 'Product', // The title displayed in the CMS for this document
  fields: [
    {
      // Field for the product's name
      name: 'name',
      type: 'string', // Basic text field
      title: 'Product Name', // Label for the field in the CMS
      validation: (Rule) =>
        Rule.required() // Makes this field mandatory
          .max(100) // Restricts the maximum length to 100 characters
          .error('Product name is required and cannot exceed 100 characters.'),
    },
    {
      // Slug field for generating a URL-friendly identifier
      name: 'slug',
      type: 'slug',
      title: 'Slug',
      description: 'URL-friendly identifier for the product.', // Helper text in the CMS
      options: {
        source: 'name', // Automatically generates the slug based on the product name
        maxLength: 200, // Limits the slug length
      },
      validation: (Rule) =>
        Rule.required().error('Slug is required for product identification.'),
    },
```

```
{
  // Field for a detailed description of the product
  name: 'description',
  type: 'text', // Multi-line text field
  title: 'Description',
  description: 'Detailed description of the product.',
  validation: (Rule) =>
    Rule.required() // Makes this field mandatory
      .min(20) // Requires at least 20 characters
      .max(500) // Limits to a maximum of 500 characters
      .error('Description must be between 20 and 500 characters.'),
},
{
  // Field for the product's price
  name: 'price',
  type: 'number', // Numeric field
  title: 'Product Price',
  validation: (Rule) =>
    Rule.required() // Makes this field mandatory
      .min(0) // Ensures the price is non-negative
      .error('Product price must be a positive value.'),
},
{
  // Field for the discount percentage
  name: 'discountPercentage',
  type: 'number',
  title: 'Discount Percentage',
  description: 'Percentage discount on the product.',
  validation: (Rule) =>
    Rule.min(0) // Ensures the discount is not negative
      .max(100) // Ensures the discount does not exceed 100%
      .error('Discount percentage must be between 0 and 100.'),
},
```

```
    {
      // Array field for tags associated with the product
      name: 'tags',
      type: 'array',
      title: 'Tags',
      of: [{ type: 'string' }], // Each tag is a string
      options: {
        layout: 'tags', // Allows for tag-style input
      },
      description: 'Add tags such as "new arrival", "bestseller", or "limited edition".',
    },
    {
      // Array field for available product sizes
      name: 'sizes',
      type: 'array',
      title: 'Sizes',
      of: [{ type: 'string' }], // Each size is a string
      options: {
        layout: 'tags', // Allows for tag-style input
      },
      description: 'Available sizes for the product (e.g., S, M, L, XL, XXL).',
    },
    {
      // Field for the product image
      name: 'image',
      type: 'image',
      title: 'Product Image',
      description: 'High-quality image of the product.',
      options: {
        hotspot: true, // Enables cropping and focal point selection
      },
      validation: (Rule) => Rule.required().error('Product image is required.'),
    },
    {
      // Field for the SEO-friendly title
      name: 'seoTitle',
      type: 'string',
      title: 'SEO Title',
      description: 'Title for SEO optimization (max 60 characters).',
      validation: (Rule) => Rule.max(60).error('SEO title cannot exceed 60 characters.'),
    },
    {
      // Field for the SEO-friendly description
      name: 'seoDescription',
      type: 'text',
      title: 'SEO Description',
      description: 'Meta description for SEO optimization (max 160 characters).',
      validation: (Rule) => Rule.max(160).error('SEO description cannot exceed 160 characters.'),
    },
  ],
};
```