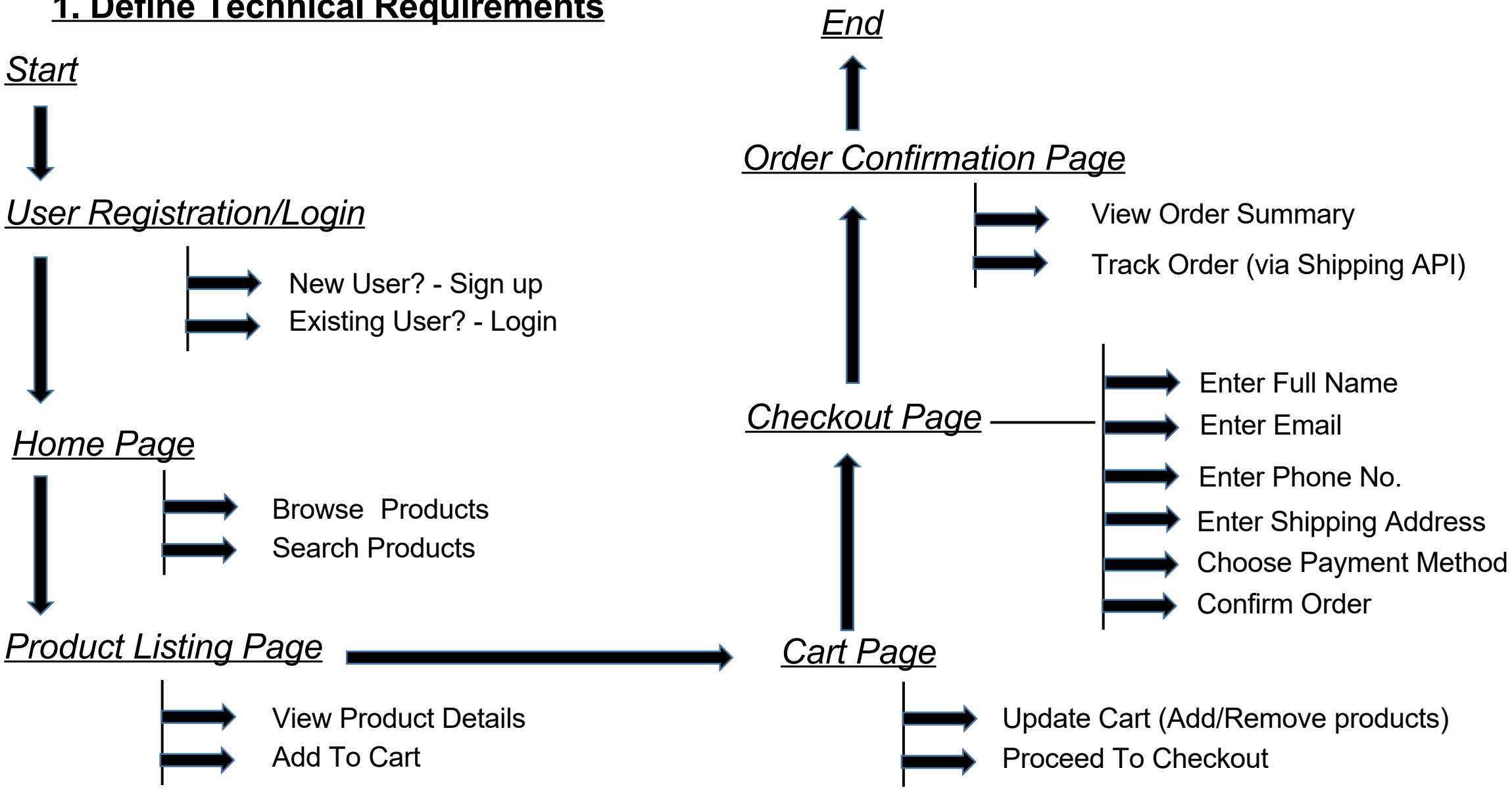
# General E-Commerce

## Marketplace Technical Foundation

## "LuxeSeaters"

**Owner: "Syed Hamail"**

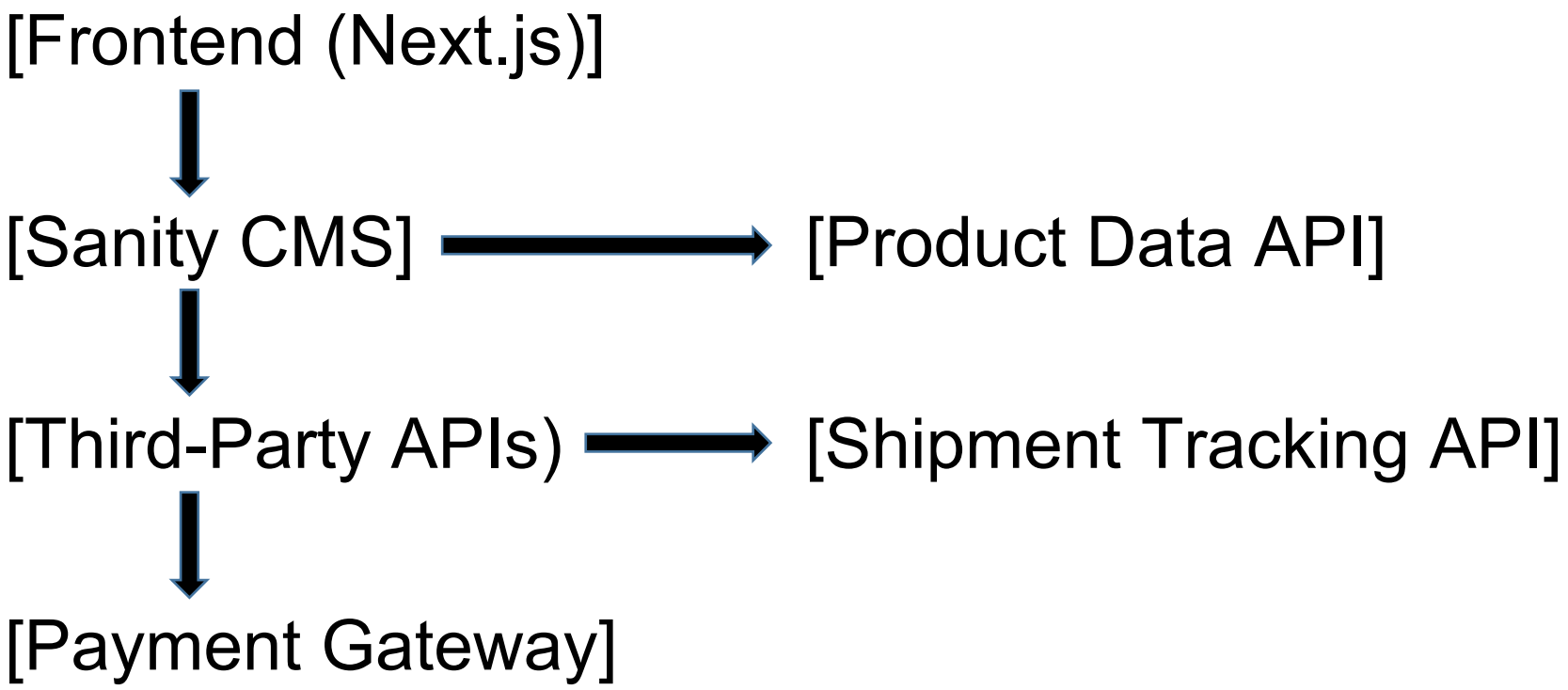
# 1. Define Technical Requirements



# Explanation of the Flowchart

1. **Start:** The user begins their journey on the platform.
2. **User Registration/Login:**
    - New users sign up.
    - Existing users log in.
3. **Home Page**:
    - Users can browse or search for products.
4. **Product Listing Page:**
    - Users view product details and add items to the cart.
5. **Cart Page:**
    - Users can update their cart (add/remove items) and proceed to checkout.
6. **Checkout Page:**
    - Users enter their fullName, email, phoneNo, shipping address, choose a payment method, and confirm the order.
7. **Order Confirmation Page:**
    - Users view their order summary and track the shipment using the shipping API.

## 2. Design System Architecture



# Key Workflows in the Diagram

1. **User Registration/Login:**
    - User signs up/logs in -> Data stored in Sanity.
2. **Product Browsing:**
    - Frontend fetches product data from Sanity via Product Data API.
3. **Cart Management:**
    - User adds/removes items from the cart (handled by Frontend).
4. **Checkout and Payment:**
    - Order details sent to Sanity -> Payment processed via Payment Gateway.
5. **Order Confirmation and Tracking:**
    - Shipment tracking info fetched from 3rd-party API -> Displayed to user.

## 3. Plan API Requirements

### 1. User Management

**Endpoint Name:** /api/users/register

**Method:** POST

**Description:** Register a new user.

**Response:** User ID and confirmation message.

**Endpoint Name:** /api/users/login

**Method:** POST

**Description:** Log in an existing user.

**Response:** JWT (JSON Web Token) token and login confirmation.

## 2. Product Management

**Endpoint Name:** /api/products

**Method:** GET

**Description:** Fetch all available products from Sanity.

**Response:** Product details (ID, name, price, stock, image, category).

## 3. Order Management

**Endpoint Name:** /api/orders/place

**Method:** POST

**Description:** Place an order for items in the cart.

**Response:** Order ID and confirmation message.

**Endpoint Name:** /api/orders/:order_id

**Method:** GET

**Description:** Fetch details of a specific order by ID.

**Response:** Order details (ID, products, total amount, status).

## 4. Payment and Shipping

**Endpoint Name:** /api/payments/process

**Method:** POST

**Description:** Process payment for an order.

**Response:** Payment ID and confirmation message.

**Endpoint Name:** /api/shipments/track/:order_id

**Method:** GET

**Description:** Fetch shipment tracking information for an order.

**Response:** Tracking details (tracking number, status, estimated delivery).

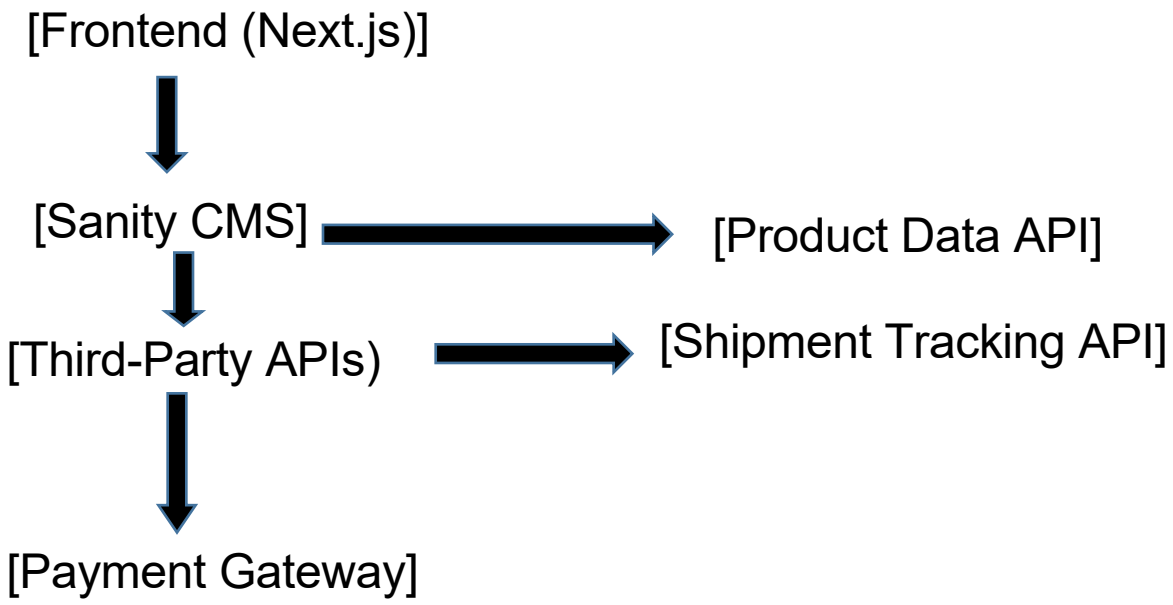
# 4. Write Technical Documentation

## LuxeSeaters Marketplace Documentation

This documentation provides a **comprehensive overview** of the LuxeSeaters marketplace's system architecture, key workflows, and API endpoints. It ensures seamless interaction between the frontend, Sanity CMS, and third-party APIs for a smooth user experience.

### 1. System Architecture Overview

#### Diagram



## Component Roles

### 1. Frontend (Next.js):

- The user interface where customers browse products, add items to the cart, and place orders.
- Communicates with Sanity CMS and third-party APIs to fetch and send data.

### 2. Sanity CMS:

- Acts as the backend database.
- Stores product details, user information, and order records.

### 3. Third-Party APIs:

- **Shipment Tracking API:** Provides real-time tracking information for orders.
- **Payment Gateway:** Processes payments securely.

## 2. Key Workflows

### Workflow 1: User Adds Products to Cart

- User browses products on the **Frontend**.
- Frontend fetches product data from **Sanity CMS** via the **Product Data API**.
- User selects a product and clicks "Add to Cart".
- Frontend updates the cart and stores the data locally or in **Sanity CMS**.

### Workflow 2: User Places an Order

- User proceeds to checkout from the cart page.
- Frontend sends order details to **Sanity CMS** for storage.
- Frontend communicates with the **Payment Gateway** to process payment.
- After successful payment, Frontend fetches shipment tracking details from the **Shipment Tracking API**.
- User receives an order confirmation with tracking information

## 3. Category-Specific Instructions

## General E-Commerce Workflows

### 1. Product Browsing:

- Users can browse products by category (e.g., casual, sports, formal).
- Example Endpoint: GET /products

### 2. Cart Management:

- Users can add/remove products from the cart.
- Example Endpoint: POST /cart/add

## 3. Order Placement:

- Users can place orders and track shipments.
- Example Endpoint: POST /orders/place

## 4. API Endpoints

| Endpoint | Method | Purpose | Response Example |
|---|---|---|---|
| **/products** | GET | Fetch all product details | { "id": 1, "name": "Tribu Elio Chair", "price": 1200, "stock": 25, "image": "url" } |
| **/products/:id** | GET | Fetch details of a specific product | { "id": 1, "name": "Tribu Elio Chair", "price": 1200, "stock": 25, "image": "url" } |
| **/cart/add** | POST | Add a product to the cart | { "message": "Product added to cart" } |
| **/cart/remove** | DELETE | Remove a product from the cart | { "message": "Product remove to cart" } |
| **/orders/place** | POST | Place an order | { "order_id": 456, "message": "Order placed successfully" } |
| **/orders/:id** | GET | Fetch details of a specific order | { "order_id": 456, "products": [{"id": 1, "name": "Tribu Elio Chair", "quantity": 1}], "total": 12000 } |
| **/payments/process** | POST | Process payment for an order | { "payment_id": 789, "message": "Payment processed successfully" } |
| **/shipments/track/:id** | GET | Fetch shipment tracking details | { "order_id": 456, "tracking_number": "ABC123", "status": "shipped" } |

# 5. Sanity Schema Example:

## 1. Product Schema

```
export default {
  name: 'product',
  type: 'document',
  title: 'Product',
  fields: [
    {
      name: 'name',
      type: 'string',
      title: 'Name',
      validation: (Rule: any) =>
Rule.required().error('Name is required'),
    },
    {
      name: 'image',
      type: 'image',
      title: 'Image',
      options: {
        hotspot: true,
      },
      description: 'Upload an image of the
product.',
    },
    {
      name: 'price',
      type: 'string',
      title: 'Price',
      validation: (Rule: any) =>
Rule.required().error('Price is required'),
    },
    {
      name: 'description',
      type: 'text',
      title: 'Description',
      validation: (Rule: any) =>
        Rule.max(150).warning('Keep the
description under 150 characters.'),
    },
```

```
{
      name: 'discountPercentage',
      type: 'number',
      title: 'Discount Percentage',
      validation: (Rule: any) =>
        Rule.min(0).max(100).warning('Discount must be between
0 and 100.'),
    },
    {
      name: 'isFeaturedProduct',
      type: 'boolean',
      title: 'Is Featured Product',
    },
    {
      name: 'stockLevel',
      type: 'number',
      title: 'Stock Level',
      validation: (Rule: any) => Rule.min(0).error('Stock level must
be a positive number.'),
    },
    {
      name: 'category',
      type: 'string',
      title: 'Category',
      options: {
        list: [
          { title: 'Chair', value: 'Chair' },
          { title: 'Sofa', value: 'Sofa' },
        ],
      },
      validation: (Rule: any) => Rule.required().error('Category is
required'),
    },
  ],
};
```

## 2. User Schema

```
export default {
    name: 'user',
    type: 'document',
    title: 'User',
    fields: [
        {
            name: 'name',
            type: 'string',
            title: 'Name',
        },
        {
            name: 'email',
            type: 'string',
            title: 'Email',
        },
        {
            name: 'password',
            type: 'string',
            title: 'Password',
        },
        {
            name: 'address',
            type: 'string',
            title: 'Address',
        },
        {
            name: 'phone_number',
            type: 'string',
            title: 'Phone Number',
        },
    ],
};
```

## 3. Order Schema

```
export default {
    name: 'order',
    type: 'document',
    title: 'Order',
    fields: [
        {
            name: 'user',
            type: 'reference',
            to: [{ type: 'user' }],
            title: 'User',
        },
        {
            name: 'products',
            type: 'array',
            title: 'Products',
            of: [
                {
                    type: 'reference',
                    to: [{ type: 'product' }],
                },
            ],
        },
        {
            name: 'total_amount',
            type: 'number',
            title: 'Total Amount',
        },
        {
            name: 'status',
            type: 'string',
            title: 'Status',
            options: {
                list: [
                    { title: 'Pending', value: 'pending' },
                    { title: 'Shipped', value: 'shipped' },
                    { title: 'Delivered', value: 'delivered' },
                ],
            },
        },
    ],
};
```

## 4. Delivery Zone Schema

```
export default {
    name: 'deliveryZone',
    type: 'document',
    title: 'Delivery Zone',
    fields: [
        {
            name: 'zone_name',
            type: 'string',
            title: 'Zone Name',
        },
        {
            name: 'shipping_cost',
            type: 'number',
            title: 'Shipping Cost',
        },
        {
            name: 'estimated_delivery_time',
            type: 'string',
            title: 'Estimated Delivery Time',
        },
    ],
};
```

# Hackathon 03

# Day 02

# Completed