HACKATHON 3

DAY 2 PLANNING THE TECHNICAL FOUNDATION

Marketplace Technical Foundation - [Hekto: Building the Future of Furniture Shopping]

Table of Contents

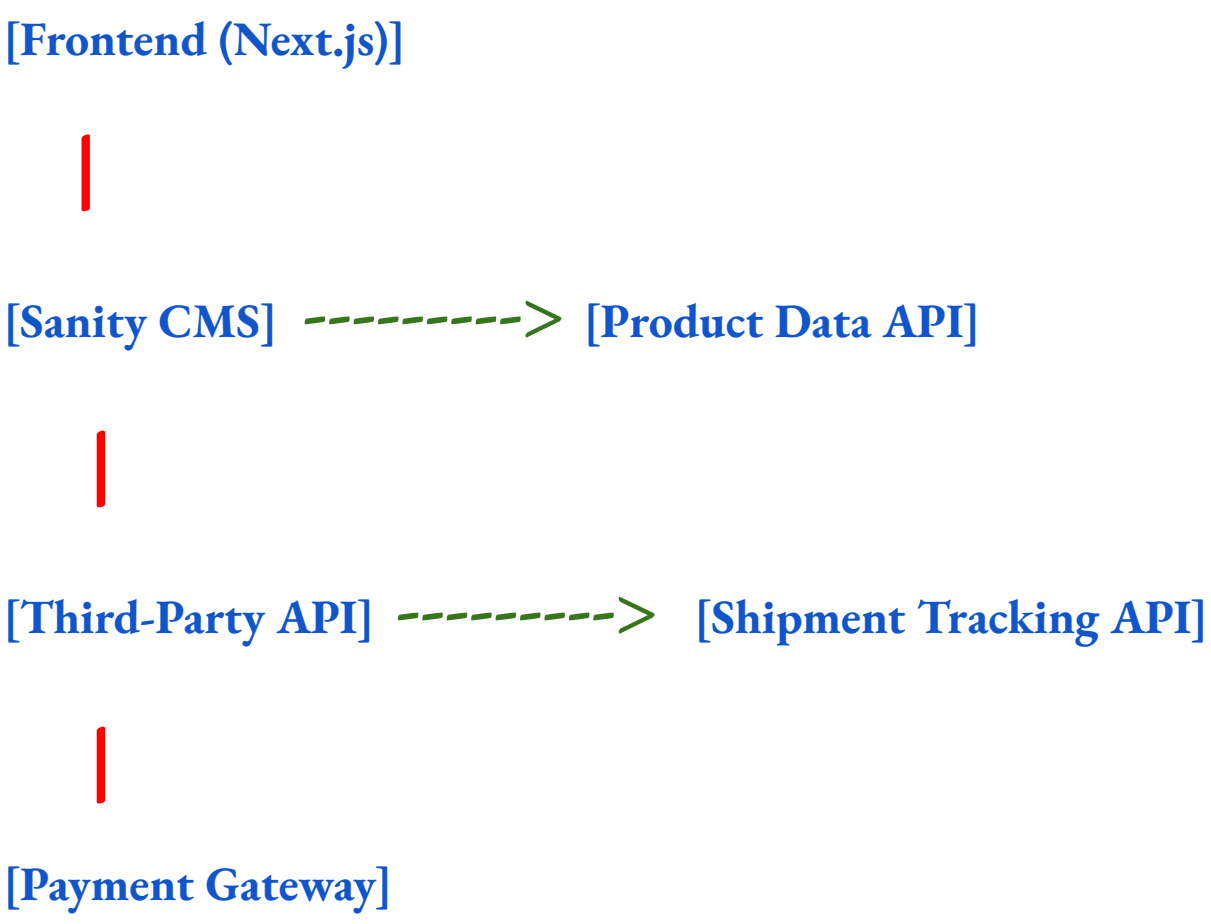
1. Overview

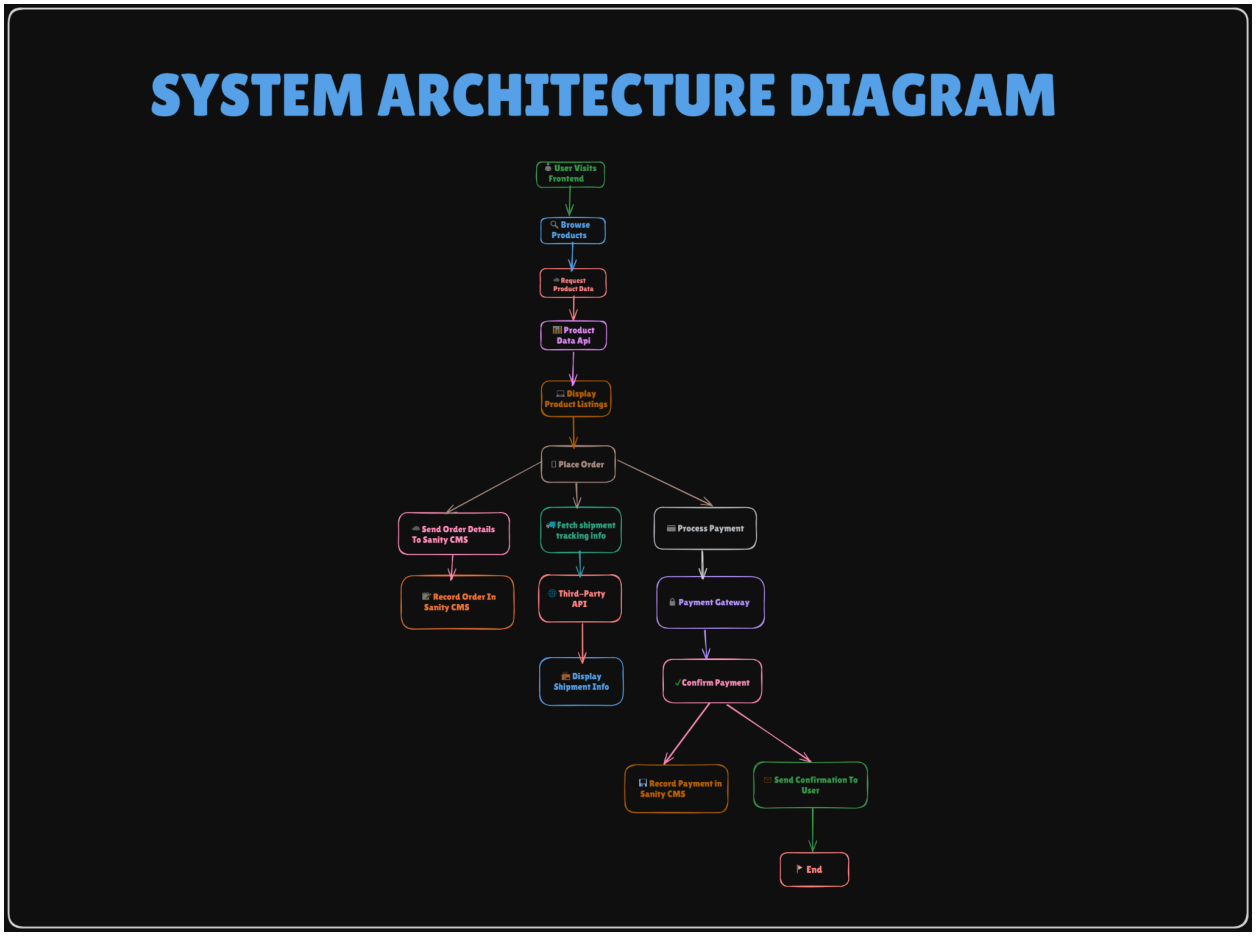
The Furniture E-Commerce Platform is designed to provide a seamless online shopping experience for customers to browse,

purchase, and track furniture items, such as sofas, chairs, and tables. The platform utilizes a Next.js frontend, a Sanity CMS backend for content management, and integrates third-party APIs for payment processing and shipment tracking.

2. System Architecture

High-Level Diagram





Component Interactions

1. **Frontend (Next.js):** Provides a dynamic and responsive user interface for browsing products, managing the shopping cart, and completing purchases.
2. **Sanity CMS:** Manages product data, customer information, order details, and content. It provides an API for accessing product data and storing order information.

3. **Third-Party APIs:** Handle external functionalities such as secure payment processing and real-time shipment tracking.
4. **Payment Gateway:** Responsible for processing user payments securely and providing transaction confirmation.

3. Technical Requirements

Frontend Requirements

- **User-Friendly Interface:**
    - Browsing and filtering categories: Sofas, chairs, tables, etc.
    - Dynamic product listings with filtering options based on price, category, and availability.
- **Pages to Implement:**

1.Home Page:

- Overview of products, promotions, and featured items.
- Includes a banner for special offers and seasonal promotions.
- Display of trending or best-selling products.

---

2. Product Listing Page:

- A catalog of products with filtering options such as categories, price ranges, and ratings.

- Sorting options like "Price: Low to High," "New Arrivals," "Best Sellers," etc.
- Users can view and browse products to select items for purchase.

---

3. Product Details Page:

- Detailed view of a single product, including:
    - Price, description, and specifications.
    - High-quality images or videos of the product.
    - Availability (in stock, out of stock).
    - Option to select quantity and size (if applicable).
    - Add to Cart button.

---

4. Cart Page:

- Allows users to view and manage their cart:
    - List of selected products, including quantities and prices.
    - Options to update quantities or remove products.
    - Display of estimated total price and taxes.
    - Option to proceed to checkout.

---

5. Checkout Page:

- User enters payment information and proceeds with the purchase:
    - Address form (shipping address).
    - Payment options (credit card, PayPal, etc.).
    - Order summary with a final price breakdown.
    - Place Order button.

---

6. Order Confirmation Page:

- Confirmation of the order with transaction details:
    - Order number, confirmation message.
    - Details of the purchased items and shipping address.
    - Estimated delivery date.
    - A thank-you note and options to continue shopping or view the order status.

---

7. Blog Page:

- Articles, product tips, and updates on promotions or new arrivals.
- Engaging content related to the products or industry.
- Option to share or comment on blog posts.

---

8. Wishlist Page:

- Allows users to save products they are interested in purchasing later.
- Option to move items to the cart or remove them from the wishlist.
- A helpful reminder for products that are on sale or about to run out of stock.
- Users can keep track of products they like but are not yet ready to purchase.

---

9. Shop Page:

- Displays all available products for immediate purchase.
- Products can be filtered or sorted by categories, price, popularity, etc.
- Users can browse through different products and add them directly to their cart.

---

10. About Page:

- Information about the store, company mission, and values.
- Brief history or background of the business.
- Contact information or links to social media profiles.

---

11. Contact Page:

- Contact form for inquiries, returns, or feedback.
- Store address, phone number, and email.
- Embedded map for the store location (if applicable).
- Social media links for additional contact methods.
    -
- **Responsive Design:** Ensures compatibility with mobile, tablet, and desktop devices.

Backend Requirements (Sanity CMS)

- **Data Management:**
    - **Products:** Store product information such as name, description, price, stock level, category, and images.
    - **Orders:** Manage customer details, ordered products, status, and timestamps.
    - **Categories:** Organize products into categories (e.g., sofas, chairs, tables).
- **Schema Design:**
    - **Product Schema:** Includes fields for product attributes (name, description, price, category, stock, and images).
    - **Order Schema:** Manages customer and order details such as status, items, and order date.

Third-Party API Integrations

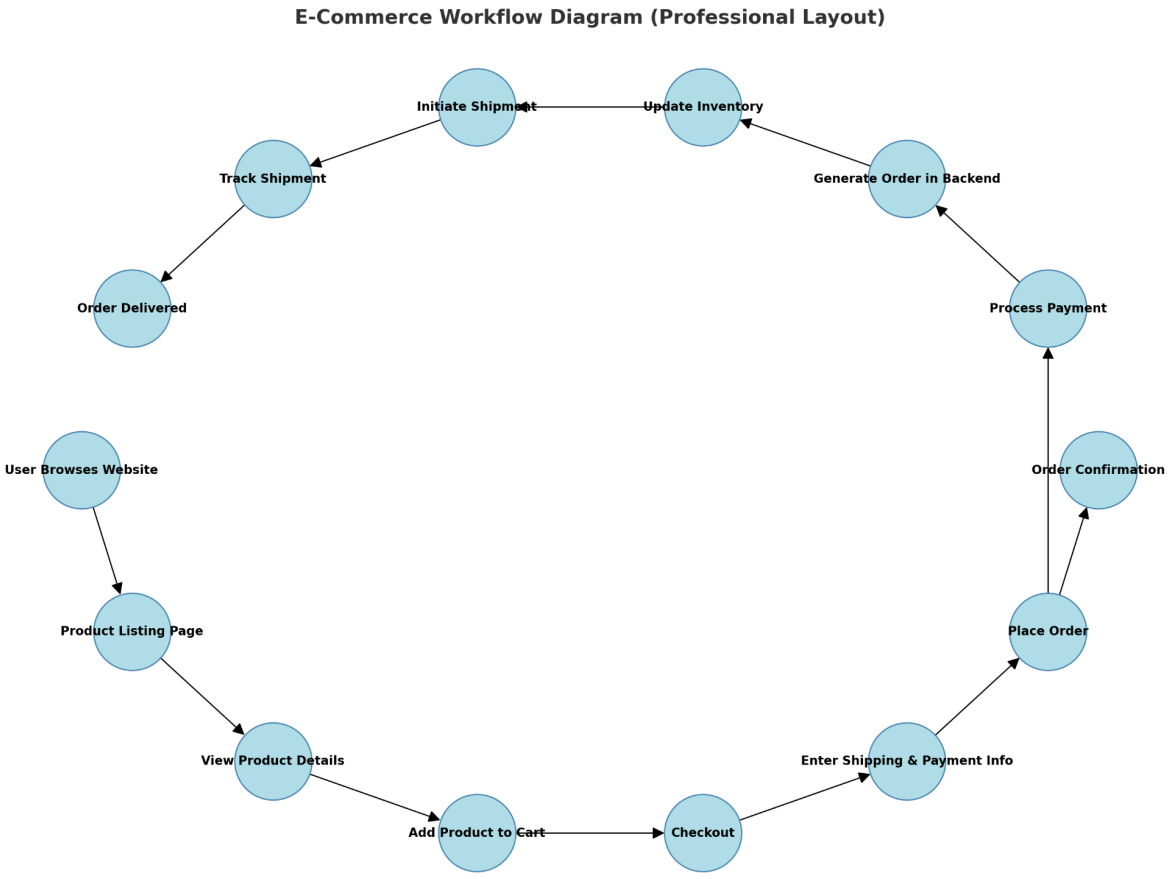
1. **Payment Gateway API:**
    - Secure payment processing for transactions.
    - Sends confirmation to both users and the backend after payment is successful.
2. **Shipment Tracking API:**
    - Provides real-time delivery updates.
    - Integrates with the frontend to display shipment progress to the user.

4. Workflows

Key Workflows

1. **User Registration:**
    - User provides details (name, email, password).
    - Data is stored in Sanity CMS.
    - Confirmation email is sent to the user.
2. **Product Browsing:**
    - Users browse products on the frontend.
    - Product data is fetched from the Sanity CMS API.
    - Products are displayed dynamically with filtering options.
3. **Order Placement:**
    - User adds items to the cart and proceeds to checkout.
    - Order details are stored in the Sanity CMS.
    - A payment gateway is used to process the transaction.

4. **Shipment Tracking:**
    - After order placement, the shipment API provides real-time tracking updates.
    - Shipment status (e.g., "Shipped," "Out for Delivery") is displayed to the user.
5. **Payment Processing:**
    - Payment information is securely transmitted to the payment gateway.
    - After payment is successful, a confirmation is sent to the user and recorded in the system.



5. Data Schema Design

1. Schema for Products (Sanity CMS)

Documentation/SanitySchema.js

```
export default {
  name: "product",
  title: "Product",
  type: "document",
```

```
  fields: [
    {
      name: "name",
      title: "Name",
      type: "string",
      validation: (Rule) => Rule.required().max(100).warning("Keep the
name short!"),
    },
    {
      name: "description",
      title: "Description",
      type: "text",
      validation: (Rule) => Rule.required().min(20).max(500),
    },
    {
      name: "rating",
      title: "Rating",
      type: "number",
```

```
    validation: (Rule) =>
      Rule.required()
        .min(0)
        .max(5)
        .warning("Rating must be between 0 and 5."),
  },
  {
    name: "price",
    title: "Price",
    type: "number",
    validation: (Rule) => Rule.required().min(0).warning("Price
cannot be negative."),
  },
  {
    name: "discountedPrice",
    title: "Discounted Price",
    type: "number",
    validation: (Rule) =>
```

```
    Rule.min(0)
      .custom((discountedPrice, context) =>
        discountedPrice > context.document.price
          ? "Discounted price cannot be greater than the original price."
          : true
      ),
},
{
  name: "stockQuantity",
  title: "Stock Quantity",
  type: "number",
  validation: (Rule) =>
    Rule.required()
      .min(0)
      .warning("Stock quantity cannot be negative."),
},
{
```

```
    name: "brand",
    title: "Brand",
    type: "string",
    validation: (Rule) => Rule.required().max(50).warning("Brand name should be short."),
  },
  {
    name: "dimensions",
    title: "Dimensions / Size",
    type: "object",
    fields: [
      {
        name: "width",
        title: "Width",
        type: "number",
        validation: (Rule) => Rule.min(0).warning("Width cannot be negative."),
      },
```

```
      {
        name: "height",
        title: "Height",
        type: "number",
        validation: (Rule) => Rule.min(0).warning("Height cannot be
negative."),
      },
      {
        name: "depth",
        title: "Depth",
        type: "number",
        validation: (Rule) => Rule.min(0).warning("Depth cannot be
negative."),
      },
    ],
    options: { collapsible: true },
  },
  {
```

```
  name: "colors",
  title: "Colors",
  type: "array",
  of: [{ type: "string" }],
  options: {
    layout: "tags",
  },
},
{
  name: "categories",
  title: "Categories",
  type: "array",
  of: [
    {
      type: "reference",
      to: [{ type: "category" }],
    },
```

```
  ],
},
{
  name: "tags",
  title: "Tags",
  type: "array",
  of: [{ type: "string" }],
  options: {
    layout: "tags",
  },
},
{
  name: "image",
  title: "Image",
  type: "image",
  options: {
    hotspot: true,
```

```
    },
    fields: [
     {
       name: "alt",
       title: "Alt Text",
       type: "string",
       validation: (Rule) => Rule.required().warning("Alt text is
important for accessibility."),
      },
    ],
   },
 ],
};
```

2. Schema for Orders (Sanity CMS)

Documentation/SanitySchema.js

```
export default {
  name: "order",
  title: "Order",
```

```
  type: "document",
  fields: [
    {
      name: "customerId",
      title: "Customer ID",
      type: "reference",
      to: [{ type: "customer" }], // Ensure you have a `customer` schema defined
      validation: (Rule) => Rule.required().error("Customer ID is required."),
    },
    {
      name: "customerName",
      title: "Customer Name",
      type: "string",
      validation: (Rule) =>
        Rule.required()
```

```
      .min(2)
      .max(100)
      .warning("Name should be between 2 to 100 characters."),
},
{
  name: "customerEmail",
  title: "Customer Email",
  type: "string",
  validation: (Rule) =>
    Rule.required().email().error("Must be a valid email address."),
},
{
  name: "items",
  title: "Ordered Items",
  type: "array",
  of: [
```

```
      {
        type: "object",
        fields: [
          {
            name: "productId",
            title: "Product ID",
            type: "reference",
            to: [{ type: "product" }], // Ensure you have a `product`
schema defined
            validation: (Rule) =>
              Rule.required().error("Each item must include a product."),
          },
          {
            name: "quantity",
            title: "Quantity",
            type: "number",
            validation: (Rule) =>
```

```
            Rule.required()
              .min(1)
              .error("Quantity must be at least 1."),
          },
        ],
      },
    ],
    validation: (Rule) =>
      Rule.required().min(1).error("Order must include at least one item."),
  },
  {
    name: "totalPrice",
    title: "Total Price",
    type: "number",
    validation: (Rule) =>
      Rule.required()
```

```
      .min(0)
      .error("Total price must be a positive value."),
},
{
  name: "orderStatus",
  title: "Order Status",
  type: "string",
  options: {
    list: [
      { title: "Pending", value: "Pending" },
      { title: "Processing", value: "Processing" },
      { title: "Shipped", value: "Shipped" },
      { title: "Delivered", value: "Delivered" },
      { title: "Cancelled", value: "Cancelled" },
    ],
    layout: "dropdown",
```

```
  },
  initialValue: "Pending",
  validation: (Rule) => Rule.required(),
},
{
  name: "paymentStatus",
  title: "Payment Status",
  type: "string",
  options: {
    list: [
      { title: "Unpaid", value: "Unpaid" },
      { title: "Paid", value: "Paid" },
      { title: "Refunded", value: "Refunded" },
    ],
    layout: "dropdown",
  },
```

```
    initialValue: "Unpaid",
    validation: (Rule) => Rule.required(),
  },
  {
    name: "deliveryAddress",
    title: "Delivery Address",
    type: "object",
    fields: [
      {
        name: "street",
        title: "Street",
        type: "string",
        validation: (Rule) => Rule.required().error("Street address is
required."),
      },
      {
        name: "city",
```

```
  title: "City",
  type: "string",
  validation: (Rule) => Rule.required().error("City is required."),
},
{
  name: "state",
  title: "State",
  type: "string",
  validation: (Rule) => Rule.required().error("State is required."),
},
{
  name: "postalCode",
  title: "Postal Code",
  type: "string",
  validation: (Rule) =>
    Rule.required().error("Postal code is required."),
```

```
      },
      {
        name: "country",
        title: "Country",
        type: "string",
        validation: (Rule) => Rule.required().error("Country is
required."),
      },
    ],
  },
  {
    name: "timestamp",
    title: "Order Timestamp",
    type: "datetime",
    options: {
      dateFormat: "YYYY-MM-DD",
      timeFormat: "HH:mm",
```

```
      calendarTodayLabel: "Today",
    },
    initialValue: () => new Date().toISOString(),
    validation: (Rule) => Rule.required(),
  },
 ],
};
```

3. Schema for Shipments (Sanity CMS)

Documentation/SanitySchema.js

```
export default {
 name: "shipment",
 title: "Shipment",
 type: "document",
 fields: [
  {
   name: "trackingNumber",
   title: "Tracking Number",
   type: "string",
   validation: (Rule) =>
    Rule.required()
     .min(5)
```

```
        .max(50)
        .warning("Tracking number should be between 5 to 50
characters."),
    },
    {
      name: "order",
      title: "Associated Order",
      type: "reference",
      to: [{ type: "order" }], // Link to the `order` schema
      validation: (Rule) => Rule.required().error("A shipment must be
associated with an order."),
    },
    {
      name: "carrier",
      title: "Carrier",
      type: "string",
      options: {
        list: [
          { title: "FedEx", value: "FedEx" },
          { title: "UPS", value: "UPS" },
          { title: "DHL", value: "DHL" },
          { title: "USPS", value: "USPS" },
        ],
        layout: "dropdown",
      },
```

```
  validation: (Rule) => Rule.required().error("Carrier is required."),
},
{
  name: "status",
  title: "Shipment Status",
  type: "string",
  options: {
    list: [
      { title: "In Transit", value: "In Transit" },
      { title: "Out for Delivery", value: "Out for Delivery" },
      { title: "Delivered", value: "Delivered" },
      { title: "Pending", value: "Pending" },
    ],
    layout: "dropdown",
  },
  initialValue: "Pending",
  validation: (Rule) => Rule.required(),
},
{
  name: "estimatedDeliveryDate",
  title: "Estimated Delivery Date",
  type: "datetime",
  options: {
    dateFormat: "YYYY-MM-DD",
    timeFormat: "HH:mm",
```

```
      calendarTodayLabel: "Today",
    },
    validation: (Rule) => Rule.required().error("Estimated delivery
date is required."),
  },
  {
    name: "actualDeliveryDate",
    title: "Actual Delivery Date",
    type: "datetime",
    options: {
      dateFormat: "YYYY-MM-DD",
      timeFormat: "HH:mm",
      calendarTodayLabel: "Today",
    },
  },
  {
    name: "shipmentNotes",
    title: "Shipment Notes",
    type: "text",
    description: "Optional notes about the shipment.",
  },
 ],
};
```

6. API Endpoints

This document provides a detailed overview of the API endpoints required for the Furniture E-Commerce Platform. These endpoints are essential for interacting with products, orders, and shipment tracking within the platform.

1. Fetch All Available Products

Endpoint Name: `/products`

- **Method:** GET
- **Description:** Retrieves a list of all products available in the system. This data is sourced from Sanity CMS and includes product details such as ID, name, price, stock availability, and image URL.

Response Example:


```
[
 {
  "id": 1,
  "name": "Modern Sofa",
  "price": 599.99,
  "stock": 25,
  "image": "https://example.com/images/sofa.jpg"
 },
 {
  "id": 2,
```

```
  "name": "Wooden Dining Table",
  "price": 349.99,
  "stock": 15,
  "image": "https://example.com/images/dining-table.jpg"
 }
]
```

2. Create a New Order

Endpoint Name: `/orders`

- **Method:** POST
- **Description: Creates a new order by collecting the customer's details, product selections, and payment status. The order is stored in Sanity CMS.**

Payload Example:

```
{
 "customerName": "John Doe",
 "customerEmail": "johndoe@example.com",
 "items": [
  {
   "productId": 1,
   "quantity": 2,
   "price": 599.99
  },
  {
   "productId": 2,
   "quantity": 1,
   "price": 349.99
  }
 ],
```

```
 "paymentStatus": "Paid"
}
```

Response Example:

```
{
 "orderId": "12345",
 "customerName": "John Doe",
 "totalPrice": 1549.97,
 "status": "Pending"
}
```

3. Track Shipment Status

Endpoint Name: `/shipment`

- **Method:** `GET`
- **Description: Fetches the shipment status for an order using the `orderId`. This data is sourced from a third-party shipment tracking API to provide customers with real-time delivery information.**

Query Parameter Example:

```
/shipment?orderId=12345
```

Response Example:

```
{
 "shipmentId": "67890",
 "orderId": "12345",
 "status": "Shipped",
 "expectedDeliveryDate": "2025-02-01"
}
```

API Endpoint Summary

Endpoint	Method	Description	Response Fields
/products	GET	Fetch all available products from Sanity CMS.	id, name, price, stock, image
/product/:id	GET	Fetch a specific product by ID to view detailed info.	id, name, description, price, stock, images
/orders	POST	Create a new order in Sanity CMS.	orderId, customerName, totalPrice, status
/shipment	GET	Track order shipment status via third-party API.	shipmentId, orderId, status, expectedDeliveryDate

Business Workflows

1. Product Browsing and Selection:

- **The `/products` endpoint allows customers to browse the catalog of products. The data fetched from this endpoint is displayed dynamically on the product listing page.**
- **The `/product/:id` endpoint is used when a user clicks on a specific product to view more detailed information such as its description, price, and available stock.**

2. Order Creation:

- **Once customers have selected items, they can place an order by submitting the order details to the `/orders` endpoint. The backend captures customer information, order items, and payment status, and stores the order in Sanity CMS.**

3. Shipment Tracking:

- **After placing an order, customers can use the `/shipment` endpoint to track their shipment status in real-time. The system fetches updates from the third-party shipment tracking API and displays the current status and expected delivery date to the user.**

Conclusion

This API documentation outlines the key endpoints required for the Furniture E-Commerce Platform. These APIs enable seamless product browsing, order creation, and shipment tracking, ensuring a smooth and efficient user experience. The endpoints are designed to integrate seamlessly with the frontend and backend, and can be easily extended or modified as needed.

7. Technical Roadmap

Phase 1: Setup and Configuration

- **Set up Next.js for frontend development.**
- **Configure Sanity CMS with required schemas for products, categories, and orders.**

Phase 2: Frontend Development

- **Design and implement essential pages: Home, Product Listing, Product Details, Cart, Checkout, Order Confirmation.**
- **Implement responsive design for a seamless user experience on all devices.**

Phase 3: Backend Integration

- **Integrate Sanity CMS for dynamic content management.**
- **Implement third-party API integrations for payment processing and shipment tracking.**

Phase 4: Testing and Deployment

- **Conduct comprehensive unit and integration testing for all components.**
- **Deploy the frontend to a platform like Vercel and the CMS to Sanity.**
- **Ensure scalability and reliability of both frontend and backend components.**

Conclusion

This documentation outlines the comprehensive design, technical specifications, and workflows for the Furniture E-Commerce Platform. The platform is built to ensure optimal performance, scalability, and a user-friendly experience. The combination of Next.js for frontend development, Sanity CMS for backend content management, and third-party API integrations ensures a seamless experience for both customers and the business. Adjustments and updates can be made as the implementation progresses to meet the evolving needs of the business and its users.

Prepared by: Areeba Bano

Senior Student

GIAIC - 00221204