Day 2 journey

Marketplace Technical Foundation – Furniture(Chairs)

---

1. System Architecture Overview

Core Features

1. **Product Listing:**
   - Display furniture products with images, descriptions, prices, and categories.
   - Include filters for categories, price range, and materials.
2. **User Authentication:**
   - Allow users to sign up, log in, and manage their profiles.
   - Support different roles, e.g., buyers and sellers.
3. **Cart and Checkout:**
   - Add products to a cart with quantity management.
   - Integrate a secure payment gateway for transactions (e.g., Stripe, PayPal).
4. **Admin Dashboard:**
   - Manage products, categories, orders, and user roles.
   - Generate sales and inventory reports.
5. **Search and Filters:**
   - Implement a search bar with suggestions.
   - Add filters for brand, price, color, size, and material.
6. **Responsive Design:**
   - Ensure the platform is mobile-friendly and works on all devices.

Technical Foundation

1. **Frontend:**
   - **Framework:** React, Next.js
   - **Styling:** Tailwind CSS and responsive designs.
2. **Backend:**
   - **API:** REST APIs.
3. **CMS (Content Management System):**
   - Sanity for managing product and category data.
4. **Hosting and Deployment:**
   - **Frontend Hosting:** Vercel.

Advanced Features

1. **Wishlist:** Allow users to save products for later.
2. **Seller Accounts:** Enable sellers to manage their products and orders.

3. **Reviews and Ratings:** Let users review and rate furniture products.
4. **Analytics:** Track user behavior, sales trends, and top-selling products.

---

3. Category-Specific Instructions

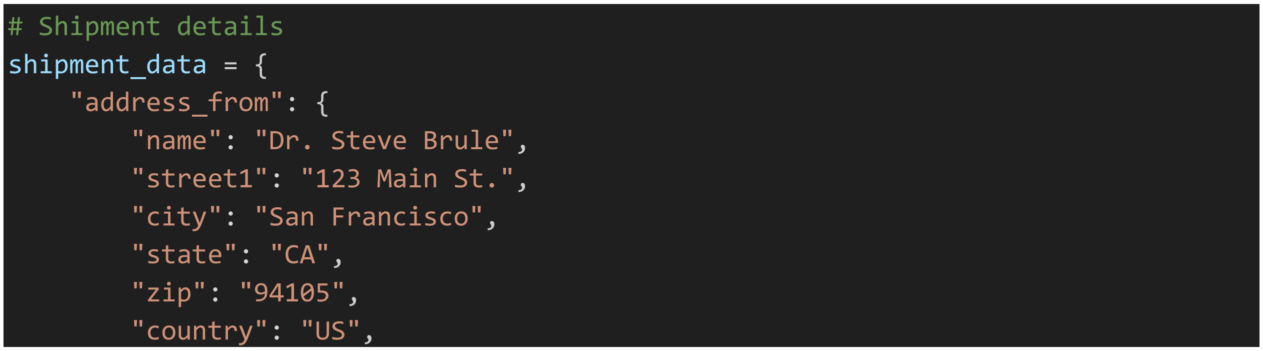
General eCommerce:

- **Product Browsing Workflow**:
  - Query products from /products endpoint with optional filters.
- **Cart Management**:
  - Add, update, or delete items from the cart using RESTful endpoints.
- **Order Placement**:
  - Process orders with payment integration.

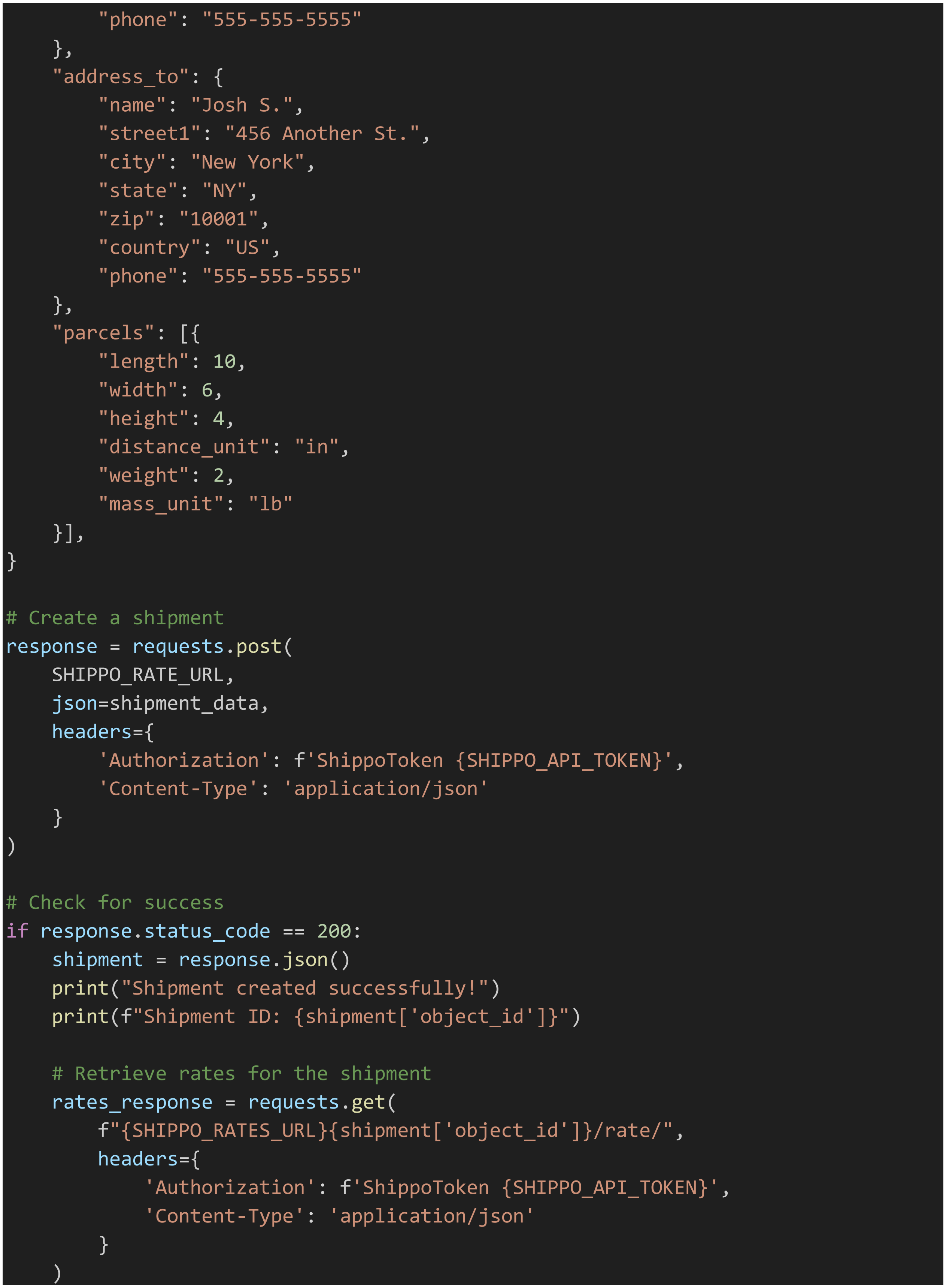
4.API Endpoints

Endpoint	Method	Purpose	Response Example
/products	GET	Fetches all product details	{ "id": 1, "name": "Product A", "price": 100, "stock": 50 }
/categories	GET	Fetches all product categories	{ "id": "123", "title": "Furniture" }
/cart	POST	Adds an item to the cart	{ "success": true, "message": "Item added to cart." }
/checkout	POST	Processes the user's order	{ "orderId": "ABC123", "status": "Processing" }
/order-status	GET	Retrieves the status of an order	{ "orderId": "ABC123", "status": "Shipped", "ETA": "2 days" }

Example Code: Getting Shipping Rates and Creating a Shipment with Shippo:



```
# Shipment details
shipment_data = {
    "address_from": {
        "name": "Dr. Steve Brule",
        "street1": "123 Main St.",
        "city": "San Francisco",
        "state": "CA",
        "zip": "94105",
        "country": "US",
```



```
        "phone": "555-555-5555"
    },
    "address_to": {
        "name": "Josh S.",
        "street1": "456 Another St.",
        "city": "New York",
        "state": "NY",
        "zip": "10001",
        "country": "US",
        "phone": "555-555-5555"
    },
    "parcels": [{
        "length": 10,
        "width": 6,
        "height": 4,
        "distance_unit": "in",
        "weight": 2,
        "mass_unit": "lb"
    }],
}

# Create a shipment
response = requests.post(
    SHIPPO_RATE_URL,
    json=shipment_data,
    headers={
        'Authorization': f'ShippoToken {SHIPPO_API_TOKEN}',
        'Content-Type': 'application/json'
    }
)

# Check for success
if response.status_code == 200:
    shipment = response.json()
    print("Shipment created successfully!")
    print(f"Shipment ID: {shipment['object_id']}")

    # Retrieve rates for the shipment
    rates_response = requests.get(
        f"{SHIPPO_RATES_URL}{shipment['object_id']}/rate/",
        headers={
            'Authorization': f'ShippoToken {SHIPPO_API_TOKEN}',
            'Content-Type': 'application/json'
        }
    )
    if rates_response.status_code == 200:
        rates = rates_response.json()
        print("Available Rates:")
        for rate in rates['rates']:
            print(f"Carrier: {rate['carrier']} - Cost: ${rate['amount']}")
    else:
        print("Failed to retrieve rates:", rates_response.json())
else:
    print("Failed to create shipment:", response.json())
```

---

5. Sanity Schema Example

Product Schema:

```
import { defineType } from "sanity";

export const productSchema = defineType({
  name: "products",
  title: "Products",
  type: "document",
  fields: [
    {
      name: "title",
      title: "Product Title",
      type: "string",
    },
    {
      name: "price",
      title: "Price",
      type: "number",
    },
    {
      title: "Price without Discount",
      name: "priceWithoutDiscount",
      type: "number",
    },
    {
      name: "badge",
      title: "Badge",
      type: "string",
    },
    {
      name: "image",
      title: "Product Image",
      type: "image",
    },
    {
      name: "category",
      title: "Category",
      type: "reference",
      to: [{ type: "categories" }],
    },
    {
      name: "description",
      title: "Product Description",
      type: "text",
    },
    {
      name: "inventory",
      title: "Inventory Management",
      type: "number",
    },
    {
      name: "tags",
      title: "Tags",
      type: "array",
      of: [{ type: "string" }],
      options: {
        list: [
          { title: "Featured", value: "featured" },
          {
            title: "Follow products and discounts on Instagram",
            value: "instagram",
          },
          { title: "Gallery", value: "gallery" },
        ],
      },
    },
  ],
});
```

Category Schema:

```
import { defineType } from "sanity";
```

```
export const categorySchema = defineType({
  name: 'categories',
  title: 'Categories',
  type: 'document',
  fields: [
    {
      name: 'title',
      title: 'Category Title',
      type: 'string',
    },
    {
      name: 'image',
      title: 'Category Image',
      type: 'image',
    },
    {
      title: 'Number of Products',
      name: 'products',
      type: 'number',
    }
  ],
});
```

6.Technical Roadmap

Phase 1: Core Development

- Build and test APIs for products, categories, cart, and checkout.
- Implement frontend structure using Next.js and Tailwind CSS.

Phase 2: Feature Enhancements

- Add user authentication and dashboard.
- Integrate third-party payment and delivery services.

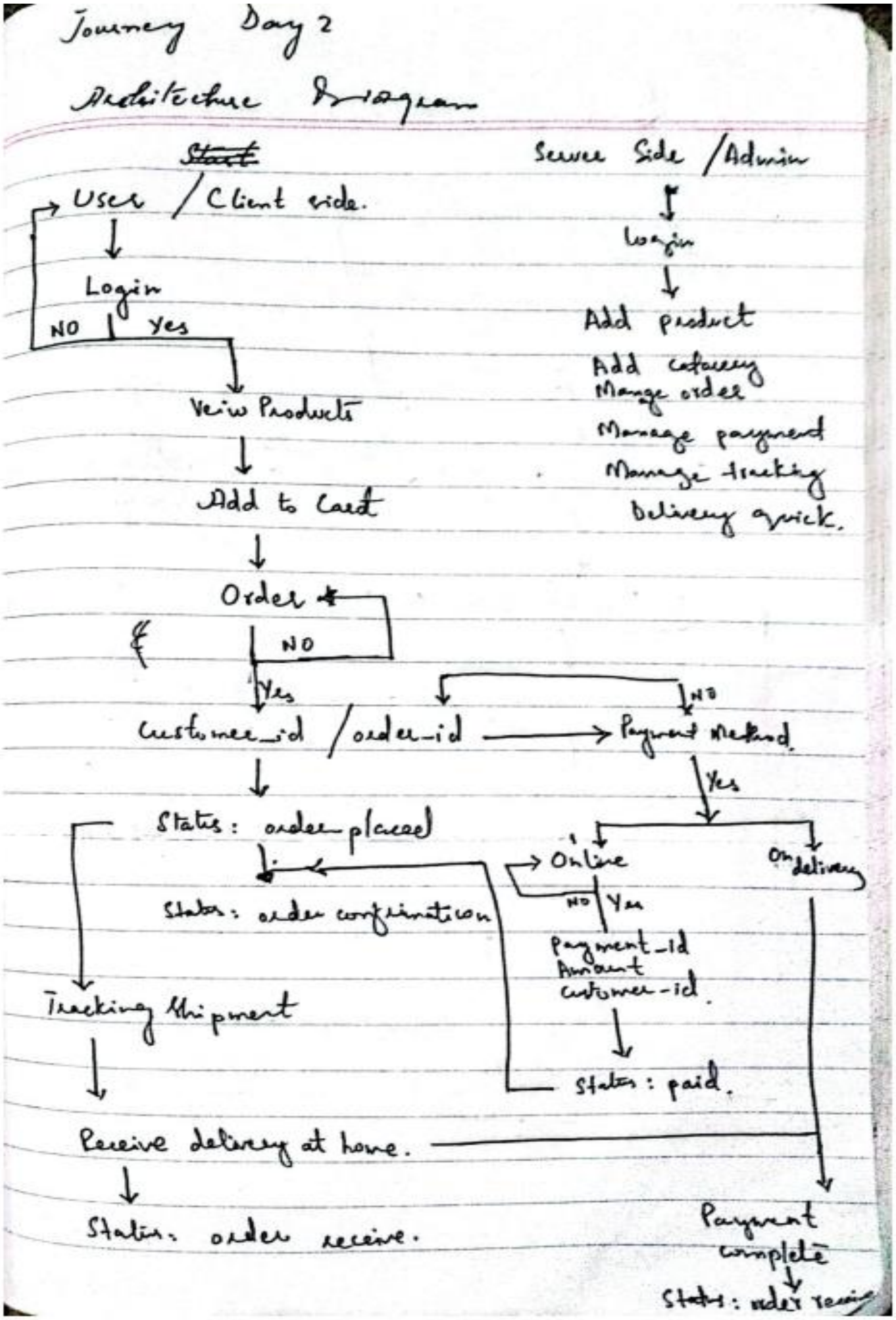
Phase 3: Scalability and Optimization

- Optimize database queries and implement caching.
- Add real-time features like live ,stock updates.

2. Key Workflows

Example Workflow: "User Adds Products to Cart"

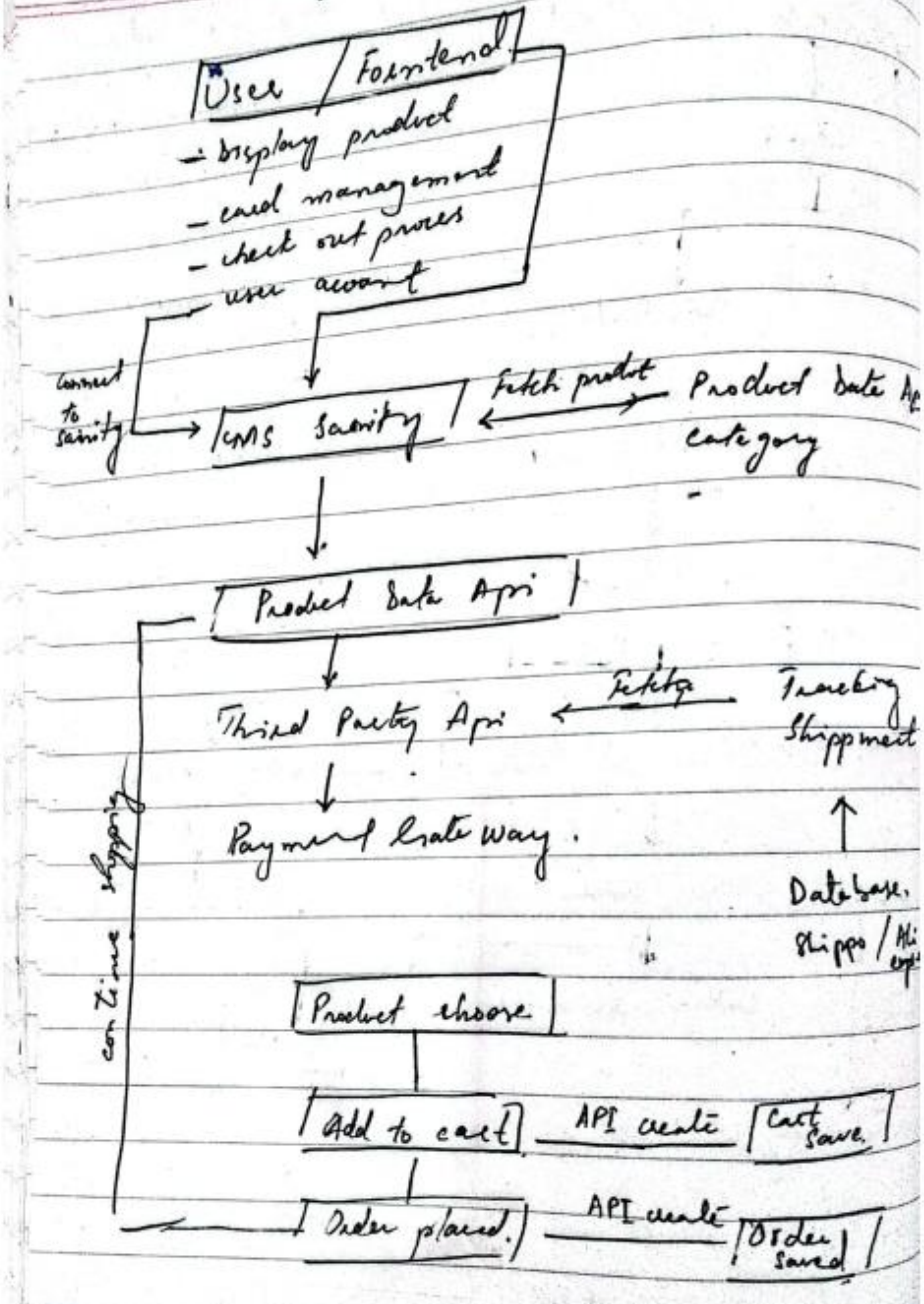
1. **User Interaction**:
   - The user selects a product and clicks "Add to Cart."
2. **Frontend Request**:
   - The frontend sends a POST request to the /cart endpoint.
3. **Backend Processing**:
   - The backend validates the product's availability via Sanity CMS.
4. **Cart Update**:
   - The updated cart is stored in the database and returned to the frontend.
5. **Frontend Update**:
   - The cart page displays the updated items and total price.



Journey Day 2

Architecture Diagram

System Architecture


User / Forntend
- Display product
- card management
- check out proces
- user account
Connect to sanity
CMS Sanity
Fetch product
Product Data Ap
category
Product Data Api
Third Party Api
Fetch
Tracking Shippment
Payment Gate way.
Database.
Shippo / Ali exp
on Time shipping
Product choose
Add to cart
API create
Cart Save
Order placed.
API create
Order Saved