Marketplace Technical Foundation – **UrbanStitch**

**Technical Plan Aligned with Business Goals**

### ****1. Introduction****

This technical plan outlines the design, development, and implementation of a general eCommerce clothing store using Next.js, Sanity CMS, and ShipEngine API. The primary objective is to build a scalable, user-friendly, and feature-rich platform that aligns with the business goals of attracting customers, increasing sales, and enhancing operational efficiency.

### ****2. Business Goals and Objectives****

#### **Primary Business Goals:**

* Provide a seamless online shopping experience for customers.
* Facilitate efficient management of product inventory and promotions.
* Enable secure and fast order processing, shipping, and tracking.

#### **Technical Objectives:**

* Build a modern, responsive, and SEO-optimized web application.
* Use a headless CMS (Sanity CMS) for flexible and scalable content management.
* Integrate ShipEngine API for real-time shipping rates, label generation, and tracking.
* Ensure high performance, scalability, and robust security.

### ****3. System Architecture Overview****

#### **Frontend:**

* Framework: **Next.js**
  + Server-Side Rendering (SSR) and Incremental Static Regeneration (ISR) for SEO and performance.
  + Dynamic routing for product categories and detail pages.
* Styling: **Tailwind CSS** for rapid UI development.
* State Management: **Context API** or for cart and user session handling.

#### **Backend:**

* API Routes: Built into Next.js to handle server-side logic.
  + /api/products: Fetch product data from **Sanity CMS.**
  + /api/cart: Manage cart operations (add, remove, update).
  + /api/checkout: Process payments using Stripe / or any other payment gateway
  + /api/shipping: Integrate with ShipEngine for shipping operations.
  + /api/orders: Save and retrieve order details.

#### **Content Management:**

* **Sanity CMS**
  + Product catalog management (name, description, pricing, images, inventory).
  + Promotional content (banners, discounts, campaigns).
  + Real-time updates via webhooks.

#### **Shipping:**

* **Ship Engine API**
  + Fetch real-time shipping rates.
  + Generate shipping labels.
  + Provide tracking information to customers.

#### **Payment Processing:**

* **Stripe API**
  + Secure payment handling with support for various **payment methods.**
  + Webhooks to handle payment status updates.

#### **Database and Caching:**

* Primary Data Source: **Sanity CMS** (structured content storage).
* Caching: Redis (for frequently accessed data like product listings).

#### **Hosting and Deployment:**

* Platform: **Vercel**
  + Supports Next.js features like ISR and serverless functions.
  + Built-in CI/CD pipelines for seamless deployments.

#### **Monitoring and Analytics(Optional)**

* Tools: LogRocket and Sentry for error tracking.
* Google Analytics for user behavior insights.

### ****4. Feature Breakdown****

#### **Customer-Facing Features:**

1. **Homepage**
   * Display featured collections, promotions, and trending products.
2. **Product Browsing**
   * Filters and sorting by category, price, size, and color.
3. **Product Detail Pages**
   * High-quality images, detailed descriptions, reviews, and size guides.
4. **Cart Management**
   * Add/remove items, quantity adjustments, and cart summary.
5. **Checkout Process**
   * Secure payment and shipping address collection.
6. **Order Tracking**
   * Customers can view real-time shipment status.

#### **Admin Features:**

1. **Content Management**
   * Manage product details, inventory, and promotions through Sanity Studio.
2. **Order Management**
   * View and update order statuses.
3. **Shipping Management**
   * Generate and manage shipping labels via ShipEngine API.
4. **Analytics Dashboard**
   * Monitor sales, user engagement, and inventory levels.

### ****5. Implementation Timeline****

| **Phase** | **Tasks** | **Duration** |
| --- | --- | --- |
| **Phase 1: Planning** | Requirements gathering and architecture design. |  |
| **Phase 2: Frontend** | Build responsive UI with Next.js and Tailwind CSS. |  |
| **Phase 3: Backend** | Develop API routes for core functionalities. |  |
| **Phase 4: CMS Setup** | Configure Sanity CMS and data modeling. |  |
| **Phase 5: Integration** | Integrate ShipEngine API and Stripe. |  |
| **Phase 6: Testing** | Conduct functional, performance, and security tests. |  |
| **Phase 7: Deployment** | Launch on Vercel with CI/CD setup. |  |

### ****6. Security Measures (optional if time permits)****

* **Authentication and Authorization:**
  + Use JWT for user sessions and admin access.
* **Data Encryption:**
  + TLS for data in transit and AES-256 for sensitive data storage.
* **Input Validation:**
  + Sanitize user inputs to prevent SQL injection and XSS attacks.
* **Rate Limiting:**
  + Implement API rate limiting to prevent abuse.

### ****7. Performance Optimization (optional if time permits)****

* **Caching:** Use Redis for quick retrieval of popular content.
* **CDN:** Deploy static assets via Cloudflare or Vercel’s built-in CDN.
* **Lazy Loading:** Defer loading of images and non-critical components.
* **Database Optimization:** Optimize GROQ queries for Sanity CMS.

### ****8. Monitoring and Maintenance (optional if time permits)****

* **Error Monitoring:** Use Sentry for real-time error tracking.
* **Performance Monitoring:** Employ LogRocket for frontend performance insights.
* **Regular Updates:** Keep dependencies updated to mitigate vulnerabilities.

### ****9. Key Performance Indicators (KPIs) (optional if time permits)****

* **Customer Metrics:**
  + Conversion rate, average order value, and cart abandonment rate.
* **Operational Metrics:**
  + Order processing time, shipping accuracy, and content update frequency.
* **Technical Metrics:**
  + Page load speed, API response time, and uptime percentage.

**10. Conclusion**

This technical plan ensures that the eCommerce platform is aligned with the business’s goals of providing an exceptional shopping experience, efficient operations, and scalability. By leveraging Next.js, Sanity CMS, and ShipEngine API, the platform will be robust, secure, and future-proof.

**2. System Architecture Visualized**

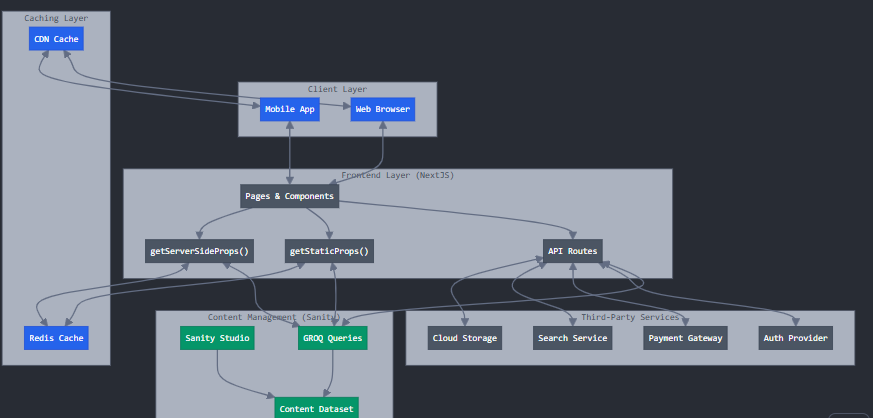
**[Frontend (Next.js)] -- API Requests --> [Sanity CMS]**

**| ^**

**| |**

**v |**

**[Third-Party APIs] <-- Interactions -- [Sanity CMS]**

****

**Data Flow**

I've created a comprehensive data flow diagram that illustrates how data moves through the system. Here's a breakdown of the key data flows:

1. Product Data Flow

* Admins manage products and content through the Admin Panel
* Content updates flow through Sanity CMS to the product pages
* Product data is served to users through Next.js API

1. User Session Flow

* Authentication data is stored in MongoDB (persistent)
* Session data is cached in Redis (temporary)
* Cart data is maintained in Redis for performance

1. Order Processing Flow

* Cart data converts to order data
* Payment processing through payment gateway
* Shipping calculations via ShipEngine
* Order data stored in MongoDB

1. Real-time Updates

* Shipping status updates from ShipEngine
* Order status updates to users
* Cart updates in real-time

## **3. Detailed API Requirements Document**

### ****1. Overview****

* **Platform Name**: UrbanStitch Marketplace API
* **Frontend**: Next.js
* **Backend & Content Management**: Sanity CMS
* **Shipping Integration**: ShipEngine API
* **Payment Integration**: Stripe API
* **Authentication**: JSON Web Tokens (JWT) or API Key (based on user roles)
* **Base URL**:
  + Development & Production: vercel link here ………….
* **Content Type**: JSON (application/json)

### ****2. Authentication****

#### **Login User**

**Endpoint**: /auth/login

* **Method**: POST
* **Request Body**:
* {
* "email": "user@example.com",
* "password": "password123"
* }
* **Response (200)**:
* {
* "token": "ey12345exampletoken",
* "expires\_in": 3600,
* "user": {
* "id": "u1",
* "name": "Jane Doe",
* "email": "user@example.com",
* "role": "customer"
* }
* }
* **Response (401)**:
* {
* "error": "Invalid email or password"
* }

### ****3. Product Management (Sanity CMS)****

#### **Get All Products**

**Endpoint**: /products

* **Method**: GET
* **Response (200)**:
* [
* {
* "id": "p1",
* "name": "Casual T-Shirt",
* "description": "100% cotton unisex t-shirt",
* "price": 19.99,
* "category": "T-Shirts",
* "sizes": ["S", "M", "L", "XL"],
* "colors": ["White", "Black"],
* "image": "https://cdn.example.com/tshirt1.jpg",
* "in\_stock": true
* },
* {
* "id": "p2",
* "name": "Denim Jacket",
* "description": "Premium quality denim jacket",
* "price": 59.99,
* "category": "Jackets",
* "sizes": ["M", "L"],
* "colors": ["Blue"],
* "image": "https://cdn.example.com/jacket1.jpg",
* "in\_stock": false
* }
* ]

#### **Get Product by ID**

**Endpoint**: /products/{id}

* **Method**: GET
* **Response (200)**:
* {
* "id": "p1",
* "name": "Casual T-Shirt",
* "description": "100% cotton unisex t-shirt",
* "price": 19.99,
* "category": "T-Shirts",
* "sizes": ["S", "M", "L", "XL"],
* "colors": ["White", "Black"],
* "image": "https://cdn.example.com/tshirt1.jpg",
* "in\_stock": true
* }

#### **Create Product (Admin Only)**

**Endpoint**: /products

* **Method**: POST
* **Request Body**:
* {
* "name": "Leather Boots",
* "description": "Genuine leather boots for men and women",
* "price": 129.99,
* "category": "Footwear",
* "sizes": ["8", "9", "10", "11"],
* "colors": ["Brown", "Black"],
* "image": "https://cdn.example.com/boots1.jpg",
* "in\_stock": true
* }
* **Response (201)**:
* {
* "message": "Product created successfully",
* "product": {
* "id": "p3",
* "name": "Leather Boots",
* "price": 129.99
* }
* }

### ****4. Category Management****

#### **Get All Categories**

**Endpoint**: /categories

* **Method**: GET
* **Response (200)**:
* [
* {
* "id": "c1",
* "name": "T-Shirts"
* },
* {
* "id": "c2",
* "name": "Jackets"
* },
* {
* "id": "c3",
* "name": "Footwear"
* }
* ]

#### **Create Category (Admin Only)**

**Endpoint**: /categories

* **Method**: POST
* **Request Body**:
* {
* "name": "Accessories"
* }
* **Response (201)**:
* {
* "message": "Category created successfully",
* "category": {
* "id": "c4",
* "name": "Accessories"
* }
* }

### ****5. Cart Management****

#### **Add to Cart**

**Endpoint**: /cart/add

* **Method**: POST
* **Request Body**:
* {
* "userId": "u1",
* "productId": "p1",
* "quantity": 2
* }
* **Response (200)**:
* {
* "message": "Product added to cart successfully",
* "cart": {
* "userId": "u1",
* "items": [
* {
* "productId": "p1",
* "quantity": 2
* }
* ]
* }
* }

### ****6. Order Management****

#### **Create Order**

**Endpoint**: /orders

* **Method**: POST
* **Request Body**:
* {
* "userId": "u1",
* "products": [
* { "productId": "p1", "quantity": 2 },
* { "productId": "p2", "quantity": 1 }
* ],
* "shippingAddress": {
* "line1": "123 Fashion St",
* "city": "Los Angeles",
* "state": "CA",
* "zip": "90001",
* "country": "US"
* }
* }
* **Response (201)**:
* {
* "message": "Order created successfully",
* "order": {
* "id": "o1",
* "totalAmount": 99.97
* }
* }

### ****7. Payment Processing (Stripe API)****

#### **Create Payment Intent**

**Endpoint**: /payments/create-intent

* **Method**: POST
* **Request Body**:
* {
* "amount": 99.97,
* "currency": "USD"
* }
* **Response (200)**:
* {
* "clientSecret": "pi\_123456789\_secret\_abcdefgh"
* }

### ****8. Shipping Management (ShipEngine API)****

#### **Get Shipping Rates**

**Endpoint**: /shipping/rates

* **Method**: POST
* **Request Body**:
* {
* "from": {
* "postalCode": "90001",
* "countryCode": "US"
* },
* "to": {
* "postalCode": "10001",
* "countryCode": "US"
* },
* "weight": {
* "value": 2.5,
* "unit": "pounds"
* }
* }
* **Response (200)**:
* [
* {
* "carrier": "UPS",
* "service": "Ground",
* "rate": 12.99,
* "estimatedDeliveryDays": 3
* },
* {
* "carrier": "FedEx",
* "service": "Express",
* "rate": 24.99,
* "estimatedDeliveryDays": 1
* }
* ]

#### **Create Shipping Label**

**Endpoint**: /shipping/create-label

* **Method**: POST
* **Request Body**:
* {
* "shipment": {
* "fromAddress": {
* "line1": "Warehouse 1",
* "city": "Los Angeles",
* "state": "CA",
* "zip": "90001",
* "country": "US"
* },
* "toAddress": {
* "line1": "123 Fashion St",
* "city": "New York",
* "state": "NY",
* "zip": "10001",
* "country": "US"
* },
* "weight": {
* "value": 2.5,
* "unit": "pounds"
* },
* "carrier": "UPS",
* "serviceCode": "ground"
* }
* }
* **Response (200)**:
* {
* "label": {
* "labelUrl": "https://cdn.example.com/shipping\_label.pdf",
* "trackingNumber": "1Z12345E1234567890"
* }
* }

### ****9. Error Handling****

#### General Error Format

* **Response (4xx/5xx)**:
* {
* "error": "Error description",
* "details": "Optional details about the error"
* }

**4. Sanity Schemas Drafted**

### ****1. Product Schema****

Defines the structure for products in the marketplace, including key attributes like name, price, category, sizes, and colors.

export default {

name: "product",

type: "document",

title: "Product",

fields: [

{

name: "name",

type: "string",

title: "Product Name",

validation: (Rule) => Rule.required().max(100),

},

{

name: "slug",

type: "slug",

title: "Slug",

options: {

source: "name",

maxLength: 96,

},

validation: (Rule) => Rule.required(),

},

{

name: "description",

type: "text",

title: "Description",

validation: (Rule) => Rule.required().max(500),

},

{

name: "price",

type: "number",

title: "Price",

validation: (Rule) => Rule.required().min(0),

},

{

name: "category",

type: "reference",

to: [{ type: "category" }],

title: "Category",

validation: (Rule) => Rule.required(),

},

{

name: "sizes",

type: "array",

title: "Available Sizes",

of: [{ type: "string" }],

options: {

list: ["XS", "S", "M", "L", "XL", "XXL"],

},

},

{

name: "colors",

type: "array",

title: "Available Colors",

of: [{ type: "string" }],

},

{

name: "inStock",

type: "boolean",

title: "In Stock",

initialValue: true,

},

{

name: "images",

type: "array",

title: "Product Images",

of: [{ type: "image", options: { hotspot: true } }],

},

],

};

### ****2. Category Schema****

Organizes products into categories, such as "T-Shirts," "Jackets," or "Footwear."

export default {

name: "category",

type: "document",

title: "Category",

fields: [

{

name: "name",

type: "string",

title: "Category Name",

validation: (Rule) => Rule.required().max(50),

},

{

name: "slug",

type: "slug",

title: "Slug",

options: {

source: "name",

maxLength: 96,

},

validation: (Rule) => Rule.required(),

},

{

name: "description",

type: "text",

title: "Description",

validation: (Rule) => Rule.max(200),

},

],

};

### ****3. Customer Schema****

Manages customer profiles, including contact information and addresses.

export default {

name: "customer",

type: "document",

title: "Customer",

fields: [

{

name: "name",

type: "string",

title: "Full Name",

validation: (Rule) => Rule.required(),

},

{

name: "email",

type: "string",

title: "Email",

validation: (Rule) => Rule.required().email(),

},

{

name: "phone",

type: "string",

title: "Phone Number",

validation: (Rule) =>

Rule.required().regex(/^\+?\d{10,15}$/, {

name: "phone number",

}),

},

{

name: "addresses",

type: "array",

title: "Addresses",

of: [

{

type: "object",

fields: [

{ name: "line1", type: "string", title: "Address Line 1" },

{ name: "city", type: "string", title: "City" },

{ name: "state", type: "string", title: "State" },

{ name: "zip", type: "string", title: "Postal Code" },

{ name: "country", type: "string", title: "Country" },

],

},

],

},

],

};

### ****4. Order Schema****

Tracks customer orders, including product details, shipping, and payment status.

export default {

name: "order",

type: "document",

title: "Order",

fields: [

{

name: "orderId",

type: "string",

title: "Order ID",

validation: (Rule) => Rule.required(),

},

{

name: "customer",

type: "reference",

to: [{ type: "customer" }],

title: "Customer",

validation: (Rule) => Rule.required(),

},

{

name: "products",

type: "array",

title: "Products",

of: [

{

type: "object",

fields: [

{

name: "product",

type: "reference",

to: [{ type: "product" }],

title: "Product",

},

{ name: "quantity", type: "number", title: "Quantity" },

{ name: "price", type: "number", title: "Price Per Unit" },

],

},

],

},

{

name: "totalAmount",

type: "number",

title: "Total Amount",

validation: (Rule) => Rule.required().min(0),

},

{

name: "shippingAddress",

type: "object",

title: "Shipping Address",

fields: [

{ name: "line1", type: "string", title: "Address Line 1" },

{ name: "city", type: "string", title: "City" },

{ name: "state", type: "string", title: "State" },

{ name: "zip", type: "string", title: "Postal Code" },

{ name: "country", type: "string", title: "Country" },

],

},

{

name: "status",

type: "string",

title: "Order Status",

options: {

list: ["Pending", "Processing", "Shipped", "Delivered", "Canceled"],

},

initialValue: "Pending",

},

{

name: "paymentStatus",

type: "string",

title: "Payment Status",

options: {

list: ["Pending", "Paid", "Failed", "Refunded"],

},

initialValue: "Pending",

},

],

};

### ****5. Review Schema****

Stores customer reviews for products.

export default {

name: "review",

type: "document",

title: "Review",

fields: [

{

name: "product",

type: "reference",

to: [{ type: "product" }],

title: "Product",

validation: (Rule) => Rule.required(),

},

{

name: "customer",

type: "reference",

to: [{ type: "customer" }],

title: "Customer",

validation: (Rule) => Rule.required(),

},

{

name: "rating",

type: "number",

title: "Rating",

validation: (Rule) => Rule.required().min(1).max(5),

},

{

name: "reviewText",

type: "text",

title: "Review Text",

validation: (Rule) => Rule.max(300),

},

],

};

### ****6. Additional Fields****

#### **Banner Schema (for Homepage Promotions)**

export default {

name: "banner",

type: "document",

title: "Banner",

fields: [

{

name: "title",

type: "string",

title: "Title",

validation: (Rule) => Rule.required(),

},

{

name: "image",

type: "image",

title: "Banner Image",

options: { hotspot: true },

},

{

name: "link",

type: "url",

title: "Redirect Link",

},

],

};