Marketplace Technical Foundation

E-Commerce Application Name: Comfort

Overview

Comforty is an e-commerce platform for high-quality and stylish furniture. It leverages modern technologies like Next.js, TypeScript, and Tailwind CSS for the frontend, while Sanity CMS powers its backend. Stripe handles payment processing, and ShipEngine manages real-time shipment tracking, ensuring a seamless and reliable experience for users.

Technical Overview

1. Frontend

Technology Stack:

- **Framework**: Next.js
  - For server-side rendering (SSR) and static site generation (SSG) to optimize performance.
- **Programming Language**: TypeScript
  - Ensures type safety and robust code structure.
- **Styling**: Tailwind CSS
  - Utility-first CSS framework for fast, responsive, and consistent styling.
- **State Management**: Context API
  - Manages global state like cart and user authentication.
- **Hosting and Deployment**: Vercel and Netlify

Pages Overview:

- **Home Page:**
  - Featured products, top categories, navigation buttons, and visually appealing headings.
- **Product Listing Page:**
  - Displays all products with filtering and sorting options.
- **Product Details Page:**
  - Detailed view of a product (description, stock availability, reviews, etc.).
- **Product Category Page:**
  - Show specific category-related products.
- **Cart:**
  - Displays selected products with quantity, total cost, and a remove option.
- **Sign In/Sign Up:**
  - Collects customer details for saving orders and account management.
- **Checkout:**
  - Gathers address, shipment, and payment details from the customer.
- **Order Confirmation Page:**
  - Displays order status and sends email confirmation to the customer.

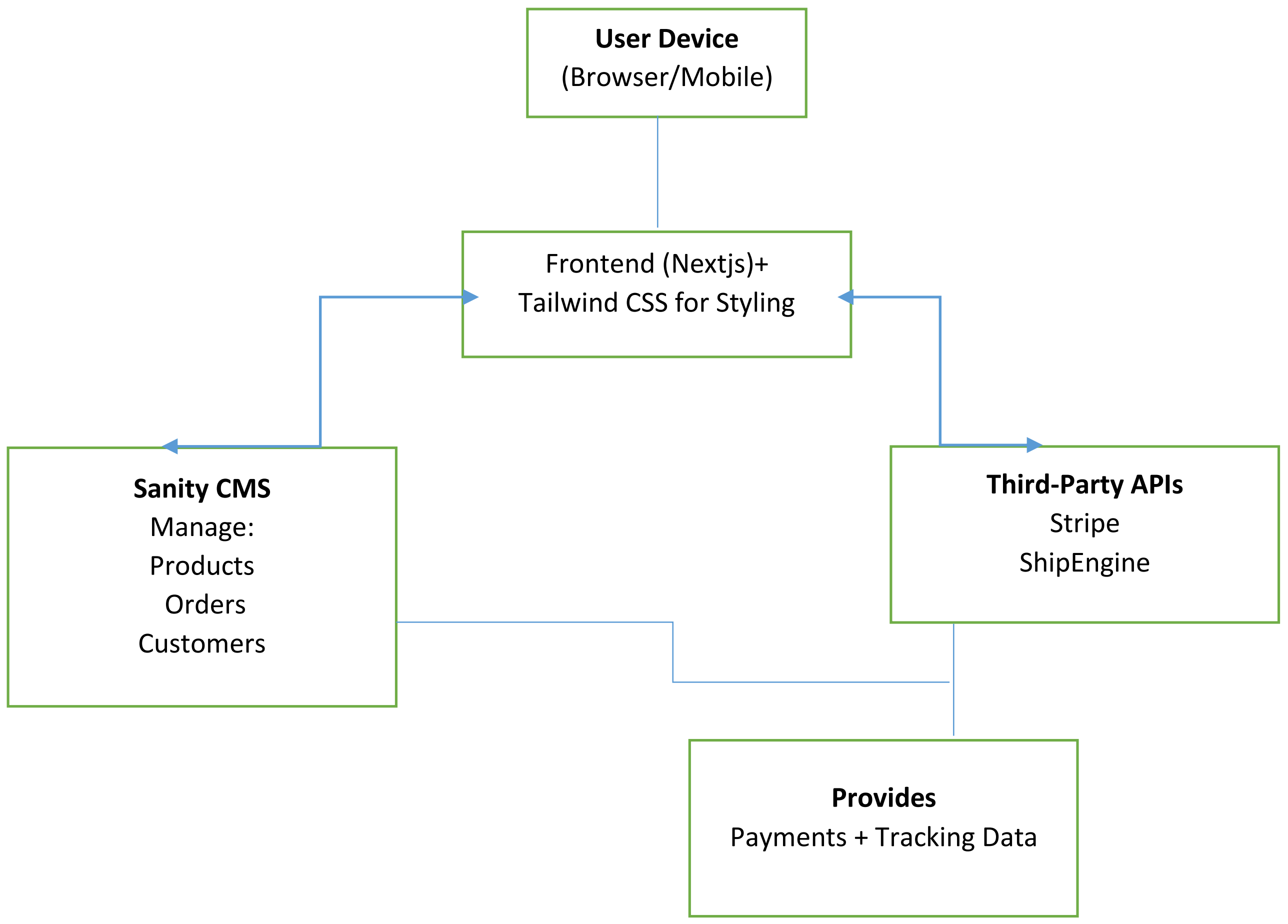
2. Backend

- **CMS: Sanity CMS**
    - Content management system to handle structured data for products, orders, and customer information.
- **RESTful APIs:**
    - Facilitate communication between frontend and backend services.
- **Authentication:**
    - Handles user sign-up, sign-in, and secure session management.

3. Third-Party API Integrations

1. **Payment Gateway: Stripe**
    - For secure and seamless payment processing.
2. **Shipment and Order Tracking: ShipEngine**
    - Provides real-time tracking details and order delivery status.
    -

System Architecture Workflow Overview



Key Workflows

1. Browsing and Product Management

- **Trigger**: User visits Home/Product Listing Page.
- **Steps**:
    1. User clicks to browse products.
    2. Frontend calls GET /api/products from Sanity CMS.
    3. Data for products is fetched and displayed using dynamic components in **Next.js**.
    4. User selects a category, and GET /api/products/category/{categoryName} endpoint is used to display products from that category.

---

2. Adding Items to Cart

- **Trigger**: User adds a product to the cart.
- **Steps**:
    1. User selects a product and clicks **Add to Cart**.
    2. Frontend updates the **Context API state** to store cart data temporarily.
    3. If the user is authenticated, the cart data is optionally synced with Sanity CMS via POST /api/cart.

---

3. User Authentication

- **Trigger**: User logs in or signs up.
- **Steps**:
    1. User enters their email and password.
    2. **Frontend** sends a request to Firebase using its authentication SDK or **POST /api/auth/signin** for login.
    3. User session is created securely, and a token is returned for subsequent requests.

---

4. Checkout Process

- **Trigger**: User initiates checkout.
- **Steps**:
    1. Cart data, user address, and payment details are submitted via POST /api/checkout.
    2. **Stripe API** is called for payment processing.
    3. If the payment is successful, the order is stored in **Sanity CMS** and marked as complete.
    4. Confirmation email is sent to the user.

---

5. Shipment Tracking

- **Trigger**: User tracks an order.
- **Steps**:
  1. User views order details via the order confirmation page.
  2. The frontend sends a request to the **ShipEngine API** using the GET /api/shipment/{orderId} endpoint.
  3. Real-time delivery updates are fetched and displayed in a user-friendly interface.

API Endpoints

Feature	Endpoint	Method	Description	Request Payload	Response Example
Products Management	/api/products	GET	Fetches all products for the product listing page.	N/A	[{"id": 1, "name": "Sofa Set", "price": 30000, "stock": 10}, {"id": 2, "name": "Dining Table", "price": 25000, "stock": 5}]
	/api/products/{id}	GET	Fetches details of a specific product.	N/A	{"id": 1, "name": "Sofa Set", "description": "Stylish and comfortable sofa.", "price": 30000, "stock": 10, "reviews": [{"user": "John", "rating": 4}, {"user": "Jane", "rating": 5}]}
	/api/products/category/{name}	GET	Fetches products by category.	N/A	[{"id": 1, "name": "Sofa Set", "category": "Living Room"},

					{"id": 2, "name": "Dining Table", "category": "Dining Room"}]
Cart Management	/api/cart	POST	Adds a product to the cart.	{"productId": 1, "quantity": 2}	{"message": "Product added to cart successfully."}
	/api/cart	GET	Fetches all items in the cart.	N/A	[{"productId": 1, "name": "Sofa Set", "quantity": 2, "price": 30000}, {"productId": 2, "name": "Dining Table", "quantity": 1, "price": 25000}]
	/api/cart/{productId}	DELETE	Removes a product from the cart.	N/A	{"message": "Product removed from cart successfully."}
User Authentication	/api/auth/signin	POST	Authenticates a user.	{"email": "user@example.com", "password": "password123"}	{"message": "Login successful.", "token": "jwt-token"}
	/api/auth/signup	POST	Registers a new user.	{"name": "John Doe", "email": "john@example.com", "password": "securePassword"}	{"message": "User registered successfully."}
Order Management	/api/orders	POST	Places an order for the items in the cart.	{"userId": 1, "address": "123 Main St", "paymentStatus": "Paid", "items": [{"productId": 1, "quantity": 2}]}	{"message": "Order placed successfully.", "orderId": 101}
	/api/orders/{orderId}	GET	Fetches details of a	N/A	{"orderId": 101, "userId": 1,

			specific order.		"items": [{"productId": 1, "name": "Sofa Set", "quantity": 2, "price": 30000}], "totalAmount": 60000, "orderStatus": "Processing" }
Shipment Tracking	/api/shipment/{orderId}	GET	Provides real-time tracking details for an order.	N/A	{"orderId": 101, "status": "In Transit", "ETA": "2025-01-20"}