MarketPlace Techinacal Fooundtaion

Revised System Architecture for Nike-like E-Commerce Website Using Next.js, Sanity CMS, Stripe, ShipEngine, Clerk, and More

1

. User Layer (Client Side) Components:

Next.js Frontend:

React-based UI built with Next.js for SSR (Server-Side Rendering) and SSG (Static Site Generation) to ensure fast page loads and SEO optimization. Handles pages

like: Home Page: Displays featured products and categories (shoes, shirts). Product Pages: Dynamic pages for individual products fetched from Sanity CMS. Order

Tracking: Fetches and displays the real-time order and shipment status from ShipEngine API. JWT Authentication with Clerk for user registration, login, and secure

access to order history and personal account pages. Mobile Application:

You can build a mobile app using React Native or Expo, reusing components and logic from the Next.js web app for consistent user experience across platforms. 2.

Application Layer (Backend) Components:

Next.js API Routes: API Routes in Next.js will handle backend tasks like order creation, user authentication, and querying product data from Sanity CMS. Key API

Routes: /api/products: Fetches all products from Sanity CMS based on categories (e.g., shoes, shirts). Fetches metadata like name, price, stock, description, and

image. /api/orders: Handles order creation, updating order status, and interacting with the Stripe payment gateway to process payments. Integrates with Clerk for

managing customer sessions and ensuring users are authenticated before placing an order. /api/order-tracking/[orderId]: Queries the ShipEngine API for real-time

shipment tracking. Provides details such as the current location of the shipment, tracking number, carrier info, and expected delivery date. Authentication with Clerk:

Clerk handles all user authentication tasks:

User registration and login via Clerk’s SDK. Secure token management (JWT) to ensure authenticated access to sensitive data like orders and user profiles. Clerk also

provides a user dashboard for managing profiles and viewing order history. Order Creation:

Once the customer is authenticated, they can place an order. The Order Management API checks product availability and processes the payment via Stripe. 3. Data

Layer (Databases & Storage) Components:

Sanity CMS:

Sanity CMS serves as the headless content management system for managing product data. Product details such as name, description, pricing, images, and stock

levels are stored here and fetched dynamically in your Next.js app. Sanity Studio is used by admins to manage products, promotions, and other content. It's an easy-

to-use interface to keep the product catalog up-to-date. Relational Database (e.g., PostgreSQL or MySQL):

Stores user information, orders, and payment details. Keeps records of past orders, user preferences, and addresses. Cloud Storage (e.g., AWS S3):

Stores product images and other media, linked to products managed through Sanity CMS. Fetches images dynamically when products are displayed on the website.

4. Third-Party Integrations Components:

Stripe (Payment Gateway):

Stripe processes payments securely by interacting with the Order API to charge customers for their orders. Supports different payment methods such as credit cards,

Apple Pay, Google Pay, etc. After payment confirmation, the system creates an order in the database and sends an order confirmation to the customer. ShipEngine

(Shipment Tracking):

ShipEngine API is used to track shipments once the order is processed and shipped. Fetches real-time tracking data like the current location of the package, delivery

status, and expected delivery date. Displays shipment tracking details in the customer’s dashboard via the Order Tracking API. Clerk (Authentication):

Handles all user management, authentication, and session management. Provides JWT-based authentication tokens to manage secure login, registration, and access

to orders and customer profiles. 5. Cloud Infrastructure & Deployment Components:

Vercel (for Next.js Deployment):

Vercel is the preferred platform for deploying Next.js applications. It handles serverless functions (API routes) for backend logic and automatically optimizes the

application for performance. Vercel CDN provides fast delivery of static content like images, stylesheets, and JavaScript, ensuring low-latency access to users

worldwide. AWS / GCP (for backend and databases):

PostgreSQL or MySQL for storing user accounts, orders, and transaction data. AWS S3 or Google Cloud Storage for storing product images, files, and assets.

Serverless Functions:

API routes in Next.js are deployed as serverless functions, ensuring cost-effective scalability and reduced operational overhead. 6. Monitoring and Analytics

Components:

Datadog / New Relic:

Monitor the performance of the entire stack, including the Next.js app, API routes, Stripe payments, and order tracking functionality. Track page load times, request

errors, and database performance. Google Analytics / Mixpanel:

Integrate with your Next.js app to track user behavior, interactions, and conversion rates. Helps analyze traffic, identify popular products, and optimize user

experience. 7. Security Components:

Clerk Authentication:

Clerk ensures secure user authentication and session management with JWT tokens. Role-based access control (RBAC) is implemented for admins and sellers to

manage products and orders. Payment Security with Stripe:

Stripe handles payment information securely using encryption and tokenization. PCI-compliant to ensure secure transactions. SSL/TLS Encryption:

All communication between the client and server is encrypted using HTTPS to ensure data security. Flow Overview Customer Browsing:

The customer browses the product catalog (shoes, shirts) on the Next.js frontend. Product data is fetched dynamically from Sanity CMS via API routes in Next.js.

Authentication:

The customer logs in or registers using Clerk, which returns a JWT token for secure access. All authenticated routes, like order history and account settings, are

protected with JWT. Order Creation:

Once authenticated, the customer selects products, adds them to the cart, and proceeds to checkout. The Order API handles order creation and initiates payment

processing via Stripe. If payment is successful, the order is stored in the database, and the customer is notified. Shipment Tracking:

After the order is shipped, the ShipEngine API provides real-time shipment tracking updates. The Order Tracking API displays shipment status to the customer in their

dashboard. Notifications:

The Notification API (could integrate with services like SendGrid or Twilio) sends order confirmations, shipping updates, and promotional notifications to customers.

Conclusion By using Sanity CMS for content management, Clerk for authentication, Stripe for payment processing, and ShipEngine for shipment tracking, this

architecture allows you to build a fully-featured e-commerce platform. You can take advantage of Next.js's flexibility for both frontend and API routes, providing high

performance and scalability. With secure authentication, efficient payment handling, and real-time shipment tracking, this architecture ensures a seamless and secure

shopping experience for customers.