Day 2

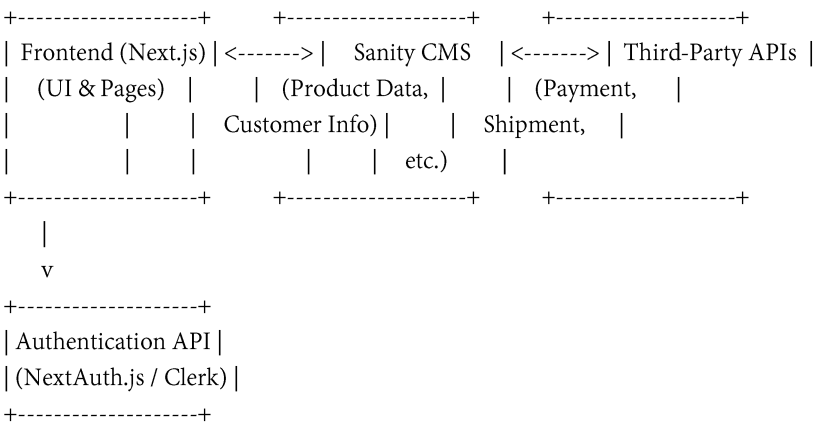
Hekto Ecommerce

Marketplace Technical Foundation - Hekto Ecommerce

1. System Architecture Overview

This section outlines the high-level architecture of the multi-category e-commerce marketplace, including interaction between the frontend (Next.js), backend (Sanity CMS), third-party APIs, and the **authentication system.**

Architecture Diagram



Component Roles:

- **Frontend (Next.js)**:
  - The user interface and application logic for displaying products, managing user authentication, and handling cart and checkout processes.
  - Uses Tailwind CSS for responsive design.
- **Sanity CMS**:
  - Handles product data for various categories (Furniture, Fashion, Electronics, etc.).
  - Manages customer information, orders, and stock levels.
- **Third-Party APIs**:
  - **Payment Gateway (Stripe)**: For secure payment processing.
  - **Shipment Tracking (ShipEngine)**: To provide real-time tracking information.
- **Authentication (NextAuth.js / Clerk)**:
  - **NextAuth.js** or **Clerk** handles user authentication, allowing users to log in, register, and manage sessions for secure access.

2. Authentication System

For managing user authentication in **Hekto Ecommerce**, we can use **NextAuth.js** or **Clerk** based on your preference. Below is the detailed explanation:

Authentication Flow:

- **Registration**:
  - Users can register by providing basic information (email, password, etc.).
  - After registration, users will receive an email confirmation link to activate their account.
- **Login**:
  - Users log in using their credentials (email/password) or social accounts (Google, Facebook, etc.).
  - The system uses **NextAuth.js** or **Clerk** to handle the login flow and manage user sessions.
- **Session Management**:
  - After a successful login, users will receive a session token, allowing them to stay logged in during their visit.
  - Sessions are managed through secure cookies.
- **Profile Management**:
  - Users can update their profile (name, email, shipping information) and view past orders.

Authentication Implementation:

- **NextAuth.js**:
  - **NextAuth.js** is a flexible authentication solution for Next.js applications, allowing social logins and email/password authentication.
  - NextAuth.js can be integrated with **Sanity** to associate user profiles with orders and cart data.
- **Clerk**:
  - **Clerk** is a complete authentication solution with additional features such as user management, session handling, and role-based access.
  - Clerk offers powerful tools for login/signup forms, passwordless login, and social authentication integrations.

Example of NextAuth.js Configuration:

```
import NextAuth from 'next-auth'
import GoogleProvider from 'next-auth/providers/google'
import CredentialsProvider from 'next-auth/providers/credentials'

export default NextAuth({
  providers: [
    GoogleProvider({
```

```
    clientId: process.env.GOOGLE_CLIENT_ID,
    clientSecret: process.env.GOOGLE_CLIENT_SECRET,
   }),
   CredentialsProvider({
    name: 'Credentials',
    credentials: {
     email: { label: 'Email', type: 'email' },
     password: { label: 'Password', type: 'password' },
    },
    async authorize(credentials) {
     const user = await fetchUserByEmail(credentials.email)
     if (user && user.password === credentials.password) {
      return user
     } else {
      return null
     }
    },
   }),
  ],
  pages: {
   signIn: '/auth/signin',  // Custom sign-in page
   error: '/auth/error',    // Error handling page
  },
  session: {
   jwt: true,  // Use JSON Web Tokens for session management
  },
  callbacks: {
   async jwt({ token, user }) {
    if (user) {
     token.id = user.id
     token.email = user.email
    }
    return token
   },
   async session({ session, token }) {
    session.id = token.id
    session.email = token.email
    return session
   },
  },
})
```

Example of Clerk Integration:

To use Clerk for authentication, you will need to install Clerk SDK and configure it in your Next.js application:

```
npm install @clerk/clerk-sdk
```

Then, in your Next.js app, you can configure Clerk as follows:

```
import { ClerkProvider, RedirectToSignIn } from '@clerk/clerk-sdk'

export default function MyApp({ Component, pageProps }) {
 return (
  <ClerkProvider>
   <Component {...pageProps} />
  </ClerkProvider>
 )
}
```

For login and signup:

```
import { SignUp } from '@clerk/clerk-sdk'

function SignupPage() {
 return (
  <div>
   <SignUp path="/sign-up" routing="path" />
  </div>
 )
}
```

---

3. Key Workflows

1. User Registers / Logs In:

- **Step 1**: User either registers or logs in via email/password or social login.
- **Step 2**: User's session is created using **NextAuth.js** or **Clerk**.
- **Step 3**: User is redirected to the homepage/dashboard after successful authentication.

2. User Browses Products by Category:

- **Step 1**: User visits the homepage with categories (Furniture, Fashion, Electronics).
- **Step 2**: User clicks on a specific category to view products.
- **Step 3**: Products are displayed with images, prices, and stock availability.
- **Step 4**: User adds products to the cart or clicks for more details.

3. User Adds Product to Cart:

- **Step 1**: User selects a product and clicks "Add to Cart".
- **Step 2**: The cart is updated, and the item (with quantity) is stored.
- **Step 3**: User can review the cart and proceed to checkout.

4. User Places an Order:

- **Step 1**: User fills out shipping information and selects a payment method.
- **Step 2**: Order is created in the backend (Sanity CMS) with all product details and customer info.
- **Step 3**: Payment is processed through Stripe.
- **Step 4**: Order confirmation and shipment tracking details are provided to the user.

---

4. API Endpoints

Endpoint	Method	Purpose	Response Example
/products	GET	Fetch all available products by category.	{ "id": 1, "name": "Sofa Set", "price": 500, "category": "Furniture" }
/product/{id}	GET	Fetch specific product details.	{ "id": 1, "name": "Sofa Set", "price": 500, "category": "Furniture" }
/orders	POST	Create a new order with customer and product details.	{ "orderId": 987, "status": "Order Created" }
/shipment/{orderId}	GET	Track the shipment for the specific order.	{ "orderId": 987, "status": "Shipped", "estimatedDelivery": "2 days" }
/payment	POST	Process payment using Stripe.	{ "status": "Success", "transactionId": "xyz789" }

---

5. Sanity Schema Example

Here is an example schema for a product in your Sanity CMS, which can be used for all categories like Furniture, Fashion, etc.

```
export default {
  name: 'product',
  type: 'document',
  fields: [
    { name: 'name', type: 'string', title: 'Product Name' },
    { name: 'price', type: 'number', title: 'Price' },
    { name: 'stock', type: 'number', title: 'Stock Level' },
    { name: 'description', type: 'text', title: 'Product Description' },
```

```
    { name: 'category', type: 'string', title: 'Category' }, // e.g., "Furniture", "Fashion"
    { name: 'image', type: 'image', title: 'Product Image' },
    { name: 'brand', type: 'string', title: 'Brand' }, // Optional for some categories
    { name: 'rating', type: 'number', title: 'Rating' }, // Optional for reviews
  ]
};
```

---

6. Technical Roadmap

Milestones & Deliverables:

1. **Milestone 1 - Project Setup**:
    - Set up Next.js project with Tailwind CSS.
    - Install and configure San

ity CMS for data management.

- Set up authentication using **NextAuth.js** or **Clerk**.

2. **Milestone 2 - Frontend Development**:
    - Develop homepage with product categories.
    - Implement product listing pages with pagination and filters.
    - Integrate dynamic product cards with add-to-cart functionality.
3. **Milestone 3 - Backend Development**:
    - Set up Sanity schemas for product and order management.
    - Configure API endpoints for product fetching, orders, and shipment tracking.
    - Integrate payment gateway for secure checkout.
4. **Milestone 4 - Testing & Optimization**:
    - Conduct user acceptance testing (UAT).
    - Optimize app for performance (load time, SEO).
    - Ensure full mobile responsiveness.
