Technical Plan Overview: E-Commerce

Marketplace

This document outlines the technical plan for developing an e-commerce marketplace that will allow small businesses and individuals to sell products online. This plan incorporates recommendations from Hackathon Day 2.

Key Technologies:

* Frontend: Next.js and Tailwind CSS for styling.
* CMS: Sanity (for dynamic content management).
* Order Tracking: ShipEngine (for real-time shipment updates).
* Payment Gateway: Stripe (for secure payment processing).

System Overview:

1. Frontend (Next.js):
   1. Client-side rendering to enhance speed and responsiveness.

○ Server-side rendering for SEO and product page preloading.

○ Tailwind CSS for styling.

1. CMS (Sanity):
   1. Manages dynamic content like banners and blog posts.

○ Integrates seamlessly with Next.js via the Sanity API.

1. Order Tracking (ShipEngine):
   1. Real-time shipment updates via ShipEngine.
2. Payment Processing (Stripe):
   1. Stripe handles payments securely, supporting multiple payment gateways.

System Components and Workflow:

1. User Registration/Login:
   1. Input: User credentials (email, password).

○ Outcome: A token will be generated for session management.

1. Content Management (Sanity CMS):
   1. Admins can manage product listings, banners, and blog content.

○ Sanity API integration is used to fetch content.

1. Product Browsing and Checkout:
   1. Users can browse products, add them to the cart, and checkout.

○ Product management is kept dynamic without an admin panel (self-service for sellers).

1. Order Management:
   1. Users can place orders, which are tracked through ShipEngine.
2. Shipment Tracking:
   1. Real-time shipment tracking via the ShipEngine API.
3. Payment Processing:
   1. Payments will be processed securely via Stripe once the order is confirmed.

API Endpoints:

1. User Management:
   1. POST /api/auth/register

○ POST /api/auth/login

○ GET /api/users/profile ○ PUT /api/users/update

1. Product Management:
   1. GET /api/products

○ GET /api/products/:id

○ POST /api/products (sellers only)

○ PUT /api/products/:id (sellers only) ○ DELETE /api/products/:id (sellers only)

1. Order Management:
   1. POST /api/orders

○ GET /api/orders ○ GET /api/orders/:id

1. Payment Management:
   1. POST /api/payments ○ GET /api/payments/status
2. Shipment Management:
   1. POST /api/shipments

○ GET /api/shipments/track

SCHEMA

Product Schema

export const Product = { name: 'product', title: 'Product', type: 'document', fields: [

{ name: 'id', title: 'Product ID', type: 'string', validation: (Rule) => Rule.required() },

{ name: 'name', title: 'Name', type: 'string', validation: (Rule) => Rule.required() },

{ name: 'description', title: 'Description', type:

'text' },

{ name: 'price', title: 'Price', type: 'number', validation: (Rule) => Rule.min(0).required() },

{ name: 'category', title: 'Category', type:

'string' },

{ name: 'mainImage', title: 'Main Image', type:

'image', validation: (Rule) => Rule.required() },

{ name: 'stock', title: 'Stock Count', type:

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

],

};

Order Schema

export const Order = {

name: 'order',

title: 'Customer Order', type: 'document', fields: [

{ name: 'orderId', title: 'Order ID', type:

'string', validation: (Rule) => Rule.required() },

{ name: 'customer', type: 'reference', to: [{ type:

'customer' }], validation: (Rule) => Rule.required() },

{ name: 'items', type: 'array', of: [{ type:

'object', fields: [

{ name: 'product', type: 'reference', to: [{ type: 'product' }] },

{ name: 'quantity', type: 'number', validation:

(Rule) => Rule.min(1).required() }

]}] },

{ name: 'totalAmount', title: 'Total Amount', type:

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

{ name: 'shippingAddress', type: 'object', fields:

[

{ name: 'street', type: 'string', validation:

(Rule) => Rule.required() },

{ name: 'city', type: 'string', validation:

(Rule) => Rule.required() },

{ name: 'zipCode', type: 'string', validation:

(Rule) => Rule.required() }

] },

{ name: 'orderDate', title: 'Order Date', type:

'datetime', validation: (Rule) => Rule.required() },

{ name: 'status', title: 'Status', type: 'string', options: { list: ['Pending', 'Shipped', 'Delivered'], layout: 'dropdown' }, validation: (Rule) => Rule.required() },

],

};

Customer Schema export const Customer = { name: 'customer', title: 'Customer', type: 'document',

fields: [

{ name: 'id', type: 'string', title: 'Customer ID', validation: (Rule) => Rule.required() },

{ name: 'name', type: 'string', title: 'Full Name', validation: (Rule) => Rule.required() },

{ name: 'email', type: 'string', title: 'Email

Address', validation: (Rule) => Rule.required().email() },

{ name: 'phone', type: 'string', title: 'Phone

Number', validation: (Rule) =>

Rule.regex(/^\+?[0-9]{10,15}$/).required() },

{ name: 'address', type: 'object', fields: [

{ name: 'street', type: 'string', validation:

(Rule) => Rule.required() },

{ name: 'city', type: 'string', validation:

(Rule) => Rule.required() },

{ name: 'zipCode', type: 'string', validation:

(Rule) => Rule.required() }

] },

{ name: 'orders', type: 'array', of: [{ type:

'reference', to: [{ type: 'order' }] }] },

],

};

Shipment Schema export const Shipment = { name: 'shipment', title: 'Shipment', type: 'document', fields: [

{ name: 'shipmentId', title: 'Shipment ID', type:

'string', validation: (Rule) => Rule.required() },

{ name: 'orderId', type: 'reference', to: [{ type:

'order' }], validation: (Rule) => Rule.required() },

{ name: 'shippingCarrier', type: 'string', title: 'Shipping Carrier', validation: (Rule) =>

Rule.required() },

{ name: 'trackingNumber', type: 'string', title:

'Tracking Number' },

{ name: 'shipmentDate', type: 'datetime', title: 'Shipment Date', validation: (Rule) =>

Rule.required() },

{ name: 'shipmentStatus', type: 'string', options:

{ list: ['Shipped', 'In Transit', 'Delivered'], layout:

'dropdown' }, validation: (Rule) => Rule.required() },

],

};

Conclusion:

This plan ensures a scalable and robust e-commerce platform using modern technologies like Next.js, Tailwind CSS, Sanity CMS, ShipEngine, and Stripe for payment processing. It allows small businesses to manage products and orders efficiently while providing a seamless user experience.



