

Marketplace Technical Foundation - FURINO

Day 2 Activities: Transitioning to Technical Planning

Technical Roadmap

This document outlines the system architecture for transitioning into technical planning. The architecture is structured to support a scalable, responsive, and user-friendly e-commerce platform. The system is divided into three key layers:

Front-end Architecture

Technology Stack:

The frontend leverages Tailwind CSS and Next.js to create a modern, responsive, and interactive user interface.

Core Features:

- Intuitive design for smooth product browsing.
- Optimized for both mobile and desktop devices.
- Includes key pages: Home, Product Listing, Product Details, Cart, Checkout, and Order Confirmation.

Responsibilities:

- Rendering dynamic UI components based on user actions and API responses.
- Maintaining high performance and accessibility standards.

Back-end Architecture

Technology Stack:

The backend is powered by Sanity CMS, serving as the primary data management system.

Core Features:

- Centralized handling of product information, customer details, and order data.
- Custom schema design aligned with the business requirements defined in the planning phase.

Responsibilities:

- Efficient storage and organization of all critical data.
- Providing APIs to enable seamless data exchange with the frontend.

Third-Party Integrations

APIs:

The system incorporates third-party services for enhanced functionality:

- Shipment tracking integration for real-time updates.

- Secure payment gateway integration for processing transactions.
- Additional backend services essential for a complete e-commerce experience.

Core Features:

- Smooth data synchronization across the frontend, backend, and external services.
- Ensuring robust security and reliability in all operations.

Responsibilities:

- Enabling backend operations to run efficiently.
- Delivering the necessary data for the frontend to operate seamlessly.

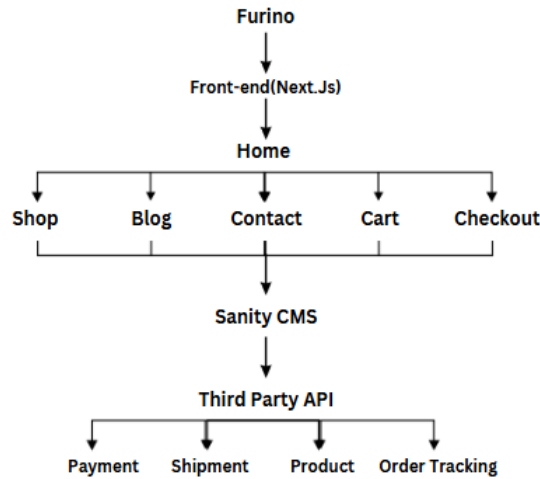
Data Schema

```

mc > ls > schemaTypes > ts products > @product > > fields
1  import { defineType } from "zod"
2
3  export const product = defineType({
4    name: "product",
5    title: "Product",
6    type: "document",
7    fields: [
8      {
9        name: "title",
10       title: "Title",
11       validation: (rule) => rule.required(),
12       type: "string"
13     },
14     {
15       name: "productID",
16       title: "Product ID",
17       validation: (rule) => rule.required(),
18       type: "string"
19     },
20     {
21       name: "description",
22       type: "text",
23       validation: (rule) => rule.required(),
24       title: "Description",
25     },
26     {
27       name: "productImage",
28       type: "image",
29       validation: (rule) => rule.required(),
30       title: "Product Image"
31     },
32     {
33       name: "price",
34       type: "number",
35       validation: (rule) => rule.required(),
36       title: "Price",
37     },
38     {
39       name: "price",
40       type: "number",
41       validation: (rule) => rule.required(),
42       title: "Price",
43     },
44     {
45       name: "tags",
46       type: "array",
47       title: "Tags",
48       of: [{ type: "string" }]
49     },
50     {
51       name: "category",
52       title: "Category",
53       validation: (rule) => rule.required(),
54       type: "string"
55     },
56     {
57       name: "discountPercentage",
58       type: "number",
59       title: "Discount Percentage",
60     },
61     {
62       name: "isNew",
63       type: "boolean",
64       title: "New Badge",
65     },
66     {
67       (property) name: string
68     },
69     {
70       name: "slug",
71       title: "Slug",
72       type: "slug",
73       options: {
74         source: "title",
75       },
76     },
77   ],
78 })

```

System Architecture



Role of Components (Brief Overview)

Furino:

The platform's core brand and framework, creating the first impression and overall structure for the user experience.

Frontend (Next.js):

- Technology: Built with Next.js for a fast, responsive, and interactive user interface.
- Pages and Features:
 - *Home*: Main landing page guiding users to key sections.
 - *Shop*: Displays products for browsing, filtering, and selection.
 - *Blog*: Shares articles and updates to boost engagement.
 - *Contact*: Allows users to send inquiries or feedback.
 - *Cart*: Summarizes selected items for review.
 - *Checkout*: Handles payments, shipping, and order confirmation.

Sanity CMS:

- Manages backend data like product details, blog posts, user profiles, and orders.
- Acts as the central hub for organizing platform data.

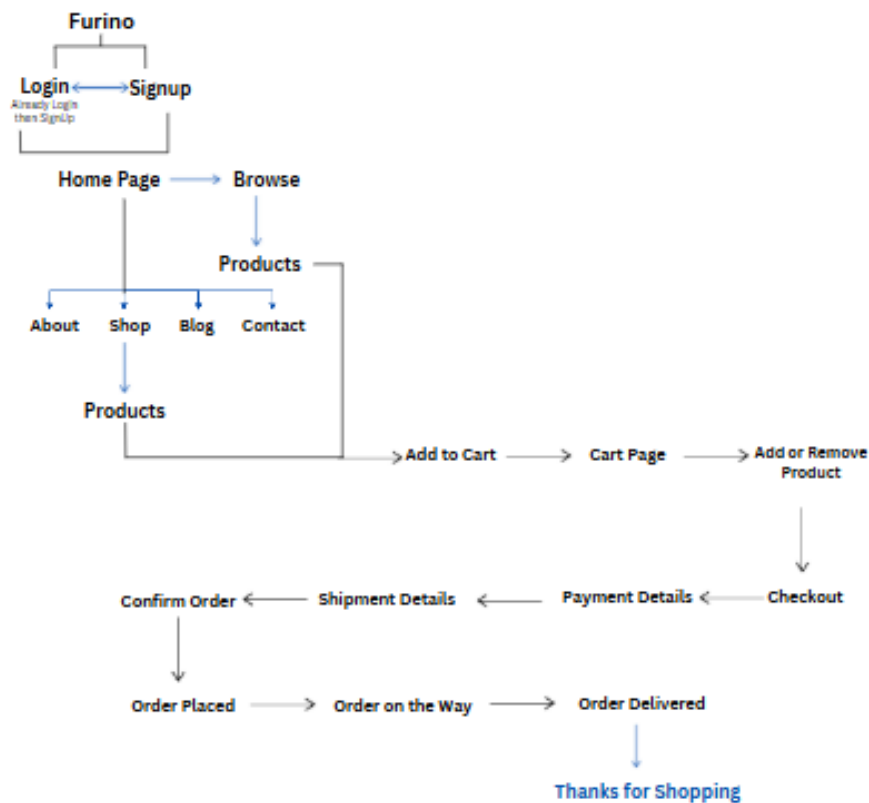
Third-Party API Integrations:

- Payment Gateways: Secure and smooth transaction processing.
- Shipping Management: Tracks and coordinates deliveries.
- Product Syncing: Ensures up-to-date product data across systems.
- Order Tracking: Provides real-time updates on order status.

API Specification

Endpoint	Method	Purpose	Response			
/api/products	GET	Fetch all products	Product list			
/api/products/{id}	GET	Fetch product details	ID, name, price, image			
/api/cart	GET	Retrieve cart items	Products, Quantity, Total Bill			
/api/shipment	POST	Add shipping details	Shipment ID, order ID, status, expected delivery date			
/api/payment	POST	Process payment	Card details, Order ID, Price			
/api/orders	GET	Get user order history	Customer info, product details, payment status			

Work Flow Diagram



By: Maida Murtaza