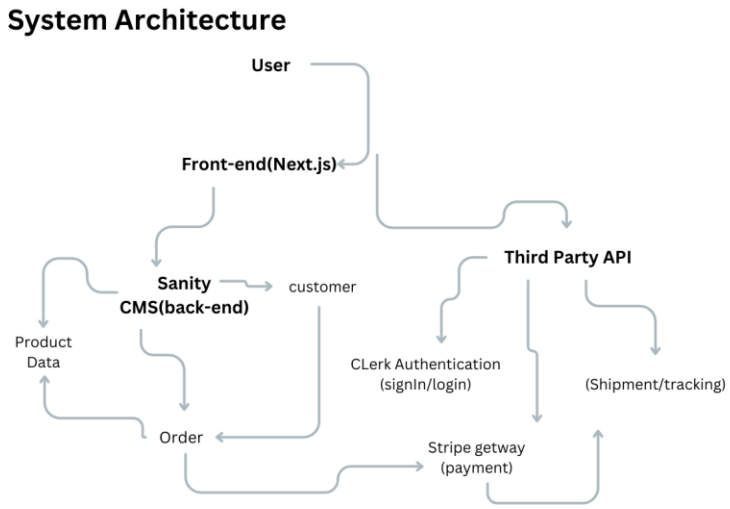
Day:02 **Marketplace Technical Foundation**

Market type: Q commerce

1:High Level Diagram:



2: Archietecture Overview

Components:

1: Frontend (Next.js):

- Delivers a responsive UI for product browsing, order management, and user authentication.

- Fetches and displays real-time data from backend APIs.

2: Sanity CMS:

- Centralized backend for managing product info, user data, orders, and inventory.

- Provides APIs for seamless frontend communication.

3: Order Placement:

- User adds items to the carts => proceeds to checkout => order details saved in sanity

4: Third-party API:

- Authentication (clerk): Handles user registration, login, and session management.

- Payment Getway (stripe): Securely processes transactions and confirms payments.

- Shipment tracking (ship-engine): Provides real-time shipping updates and tracking details.

Key Workflow:

User Registration: Frontend ↔ Clerk

Product Browsing: Frontend ↔ Sanity CMS

Order Placement: Frontend ↔ Sanity CMS

Payment Getway: Frontend ↔ Stripe

Shipment Tracking: Frontend ↔ ShipEngine

3: Plan API Requirement

Q-Commerce

1: Endpoint Name:/product

Method: GET

Description: Fetch ALL product Details dynamicaly

Response JSON: { "name": "Product Name", "slug": "product-slug", "image": "https://productimage.png", "description": "Product Description", "price": $25, "discountPrice": $18, "inStock": true, "stock": 50}

2: EndPoint Name : /Order

Method : POST

Description:post all order details

Response JSON : {“orderId” : 201,”customerID” : 3001,”quantity” : 2, }

3: EndPoint Name : /Customer

Method : POST

Description: post all order details

Response JSON : { “customerId” : 3001 , “name” : “Aleema” , “contact” : 0312-1113997 , “address” : “street 123 karachi-pakistan, }

Sanity Schema

Product Schema

```
import { defineField, defineType } from 'sanity';


export const productTypes = defineType({
  name: 'product',
  title: 'Products',
  type: 'document',
  icon: () => '🛒', // You can replace this with an icon component like TrolleyIcon
  fields: [
    defineField({
      name: 'name',
      title: 'Product Name',
      type: 'string',
      description: 'The name of the product.',
```

```
  validation: (Rule) => Rule.required(),
}),
defineField({
  name: 'slug',
  title: 'Slug',
  type: 'slug',
  description: 'URL-friendly identifier for the product.',
  options: {
    source: 'name', // Automatically generate slug from the product name
    maxLength: 96,
  },
  validation: (Rule) => Rule.required(),
}),
defineField({
  name: 'image',
  title: 'Main Image',
  type: 'image',
  description: 'The main image of the product.',
  options: {
    hotspot: true, // Enable image cropping
  },
  validation: (Rule) => Rule.required(),
}),
defineField({
  name: 'description',
  title: 'Description',
  type: 'text',
```

```
  description: 'A detailed description of the product.',
  validation: (Rule) => Rule.required(),
}),
defineField({
  name: 'price',
  title: 'Price',
  type: 'number',
  description: 'The original price of the product.',
  validation: (Rule) => Rule.required().positive(),
}),
defineField({
  name: 'discountPrice',
  title: 'Discount Price',
  type: 'number',
  description: 'The discounted price of the product (if applicable).',
  validation: (Rule) => Rule.positive(),
}),
defineField({
  name: 'inStock',
  title: 'In Stock',
  type: 'boolean',
  description: 'Indicates whether the product is in stock.',
  initialValue: true, // Default value
}),
defineField({
  name: 'stock',
  title: 'Stock',
```

```
      type: 'number',
      description: 'The current stock quantity of the product.',
      validation: (Rule) => Rule.required().integer().min(0),
    }),
    defineField({
      name: 'sizes',
      title: 'Sizes',
      type: 'array',
      description: 'Available sizes for the product.',
      of: [{ type: 'string' }], // Array of strings
      validation: (Rule) => Rule.required().min(1),
    }),
  ],
});
```

Order Schema:

```
import { defineField, defineType } from 'sanity';

export const orderTypes = defineType({
  name: 'order',
  title: 'Orders',
  type: 'document',
  icon: () => '📦', // You can replace this with an icon component
  fields: [
    defineField({
      name: 'orderId',
      title: 'Order ID',
      type: 'string',
```

```
  description: 'A unique identifier for the order.',
  validation: (Rule) => Rule.required(),
}),
defineField({
  name: 'customerId',
  title: 'Customer ID',
  type: 'string',
  description: 'The ID of the customer who placed the order.',
  validation: (Rule) => Rule.required(),
}),
defineField({
  name: 'products',
  title: 'Products',
  type: 'array',
  description: 'List of products in the order.',
  of: [
   {
    type: 'object',
    fields: [
     defineField({
      name: 'productId',
      title: 'Product ID',
      type: 'string',
      description: 'The ID of the product.',
      validation: (Rule) => Rule.required(),
     }),
     defineField({
```

```
          name: 'quantity',
          title: 'Quantity',
          type: 'number',
          description: 'The quantity of the product ordered.',
          validation: (Rule) => Rule.required().integer().min(1),
        }),
      ],
    },
  ],
  validation: (Rule) => Rule.required().min(1),
}),
defineField({
  name: 'totalAmount',
  title: 'Total Amount',
  type: 'number',
  description: 'The total amount of the order.',
  validation: (Rule) => Rule.required().positive(),
}),
defineField({
  name: 'status',
  title: 'Order Status',
  type: 'string',
  description: 'The current status of the order (e.g., Processing, Shipped, Delivered).',
  options: {
    list: [
      { title: 'Processing', value: 'processing' },
      { title: 'Shipped', value: 'shipped' },
```

```
          { title: 'Delivered', value: 'delivered' },
          { title: 'Cancelled', value: 'cancelled' },
        ],
      },
      validation: (Rule) => Rule.required(),
    }),
    defineField({
      name: 'orderDate',
      title: 'Order Date',
      type: 'datetime',
      description: 'The date and time when the order was placed.',
      validation: (Rule) => Rule.required(),
    }),
  ],
});
```

Customer Schema:

```
import { defineField, defineType } from 'sanity';

export const customerTypes = defineType({
  name: 'customer',
  title: 'Customers',
  type: 'document',
  icon: () => ' 👤 ', // You can replace this with an icon component
  fields: [
    defineField({
      name: 'customerId',
      title: 'Customer ID',
```

```
  type: 'string',
  description: 'A unique identifier for the customer.',
  validation: (Rule) => Rule.required(),
}),
defineField({
  name: 'name',
  title: 'Name',
  type: 'string',
  description: 'The full name of the customer.',
  validation: (Rule) => Rule.required(),
}),
defineField({
  name: 'contact',
  title: 'Contact',
  type: 'object',
  description: 'Contact details of the customer.',
  fields: [
    defineField({
      name: 'email',
      title: 'Email',
      type: 'string',
      description: 'The email address of the customer.',
      validation: (Rule) => Rule.required().email(),
    }),
    defineField({
      name: 'phone',
      title: 'Phone',
```

```
      type: 'string',
      description: 'The phone number of the customer.',
      validation: (Rule) => Rule.required(),
    }),
  ],
  validation: (Rule) => Rule.required(),
}),
defineField({
  name: 'address',
  title: 'Address',
  type: 'object',
  description: 'The address of the customer.',
  fields: [
    defineField({
      name: 'street',
      title: 'Street',
      type: 'string',
      description: 'The street address of the customer.',
      validation: (Rule) => Rule.required(),
    }),
    defineField({
      name: 'city',
      title: 'City',
      type: 'string',
      description: 'The city of the customer.',
      validation: (Rule) => Rule.required(),
    }),
```

```
        defineField({
          name: 'state',
          title: 'State',
          type: 'string',
          description: 'The state or region of the customer.',
          validation: (Rule) => Rule.required(),
        }),
        defineField({
          name: 'zipCode',
          title: 'Zip Code',
          type: 'string',
          description: 'The postal code of the customer.',
          validation: (Rule) => Rule.required(),
        }),
        defineField({
          name: 'country',
          title: 'Country',
          type: 'string',
          description: 'The country of the customer.',
          validation: (Rule) => Rule.required(),
        }),
      ],
      validation: (Rule) => Rule.required(),
    }),
  ],
});
```

Technical Roadmap

Development Phase

Authentication

: Implement user registration and login using Clerk.

: Integrate Clerk with Sanity CMS for user data storage.

Product managment

Create mock API for product data.

: Store product data in Sanity CMS.

: Fetch and display product data on dynamic frontend pages.

: Cart and Wishlist

: Implement add-to-cart functionality with real-time stock checks.

: Allow out-of-stock products to be added to wishlist.

: Display total bill and proceed to checkout button on cart page.

Payment integration

: Integrate Stripe for secure payments.

: Use Stripe test account for development.

: Handle payment success and failure scenarios

Shipment Tracking

: Integrate ShipEngine for shipment tracking.

: Generate tracking numbers and display on the frontend.

: Allow users to track their orders in real-time.

Testing Phase

End to End Testing

: Test all workflows, including user registration, product browsing, cart management, checkout, and shipment tracking.

: Validate API responses and ensure data accuracy.

Security Audits

: Conduct security audits for sensitive data handling, including user authentication and payment processing.

Launch Phase

Deployment

: Deploy the platform on a cloud hosting service (e.g., Vercel, Netlify).

: Monitor user feedback and optimize for performance.

Post-Launch

: Collect user feedback for continuous improvement.

: Optimize API performance and frontend loading times.

: Scale infrastructure based on traffic and demand

Conclution

This technical foundation outlines the architecture, workflows, and API endpoints for the Bandage Online Shopping platform. By following the enhanced workflow and technical roadmap, the platform will provide a seamless eCommerce experience with robust authentication, inventory management, and shipment tracking features