Marketplace Technical Foundation - Cartify

1. Introduction

Cartify is a modern e-commerce platform built with Next.js, featuring a headless CMS (Sanity), authentication via Clerk, and payment processing through Stripe. This report details the system architecture, workflows, and technologies used to ensure a seamless shopping experience.

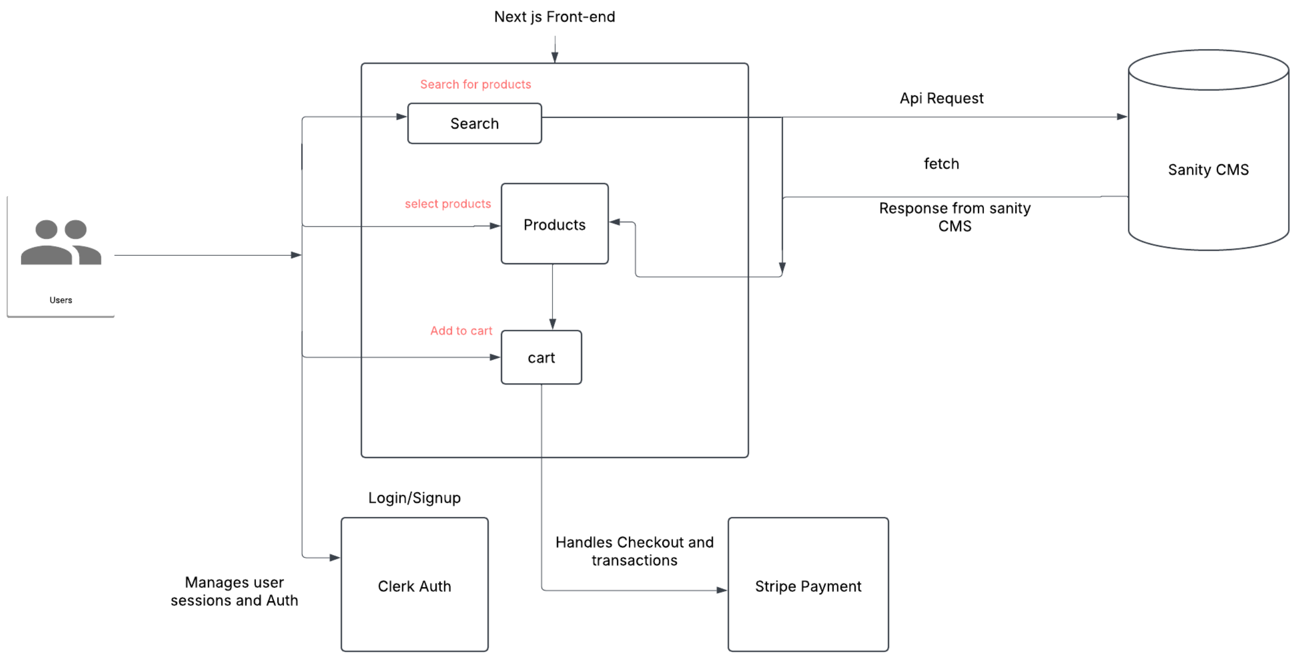
2. System Architecture Overview

Cartify follows a **headless e-commerce architecture**, meaning the frontend and backend are decoupled. Below is a high-level overview of the system:

Key Components:

- **Frontend:** Built with Next.js, Tailwind CSS, and ShadCN for a responsive UI.
- **Backend:** Uses Next.js API routes to interact with Sanity (CMS) and Stripe.
- **Authentication:** Managed via Clerk for secure user login and session handling.
- **Database & Storage:** Sanity CMS stores product data, and Stripe manages transactional data.
- **Deployment:** Hosted on Vercel for scalability and performance optimization.

System Architecture Visualized:



3. Detailed Workflow: How Cartify Works

Step 1: User Browses Products

1. User lands on the homepage.
2. Next.js fetches product data from **Sanity CMS**.
3. The UI dynamically displays available products.

Step 2: User Authentication (Clerk)

1. User clicks “Login” or “Sign Up.”
2. Clerk handles authentication (via Google, GitHub, or email/password).
3. Upon success, Clerk provides a JWT session token.

Step 3: Adding Items to Cart

1. Users browse products and click **“Add to Basket.”**
2. The cart is temporarily stored in local storage or **global state (Zustand/context API).**

Step 4: Checkout Process (Stripe Payments)

1. User proceeds to checkout.
2. Next.js sends cart data to Stripe API.
3. Stripe processes payment & returns a success/failure response.
4. Upon success, an order is stored in **Sanity CMS**.

Step 5: Order Management & History

1. Users can view past orders by fetching data from **Sanity CMS**.
2. The UI updates dynamically based on order status.

- **Technology Stack Summary**

Layer	**Technology**
Frontend	Next.js, React, Tailwind CSS, ShadCN
Backend	Next.js API Routes, Sanity CMS
Auth	Clerk
Database	Sanity CMS
Payment	Stripe
Deployment	Vercel

4. API Endpoints

API Function	**Parameters**	**Purpose**	**Example Response**
getAllProducts()	None	Fetches all available products	[{ "id": "123", "name": "T-Shirt", "price": 29.99 }]
getAllCategories()	None	Fetches all product categories	[{ "id": "1", "name": "Clothing" }]
getProductsByCategory(categorySlug: string)	categorySlug (string)	Fetches products under a category	[{ "id": "123", "name": "Jeans", "price": 49.99 }]
getProductBySlug(slug: string)	slug (string)	Fetches a single product	{ "id": "123", "name": "T-Shirt", "description": "A nice T-shirt" }
searchProductsByName(searchParam: string)	searchParam (string)	Searches for products by name	[{ "id": "456", "name": "Shoes", "price": 79.99 }]

Database Structure (Sanity Schemas)

1. Product Schema

```
import { TrolleyIcon } from "@sanity/icons";
import { defineField, defineType } from "sanity";

export const productType = defineType({
  name: "product",
  title: "Products",
  type: "document",
  icon: TrolleyIcon,
  fields: [
    defineField({
      name: "name",
      title: "Product Name",
      type: "string",
      validation: (Rule) => Rule.required(),
    }),
    defineField({
      name: "slug",
      title: "Slug",
      type: "slug",
      options: {
        source: "name",
        maxLength: 96,
      },
      validation: (Rule) => Rule.required(),
    }),
    defineField({
      name: "image",
      title: "Product Image",
      type: "image",
      options: { hotspot: true },
    }),
    defineField({
      name: "description",
      title: "Description",
      type: "blockContent",
    }),
    defineField({
      name: "price",
      title: "Price",
      type: "number",
      validation: (Rule) => Rule.required(),
    }),
    defineField({
      name: "categories",
      title: "Categories",
      type: "array",
      of: [{ type: "reference", to: { type: "category" } }],
    }),
    defineField({
      name: "stock",
      title: "Stock",
      type: "number",
      validation: (Rule) => Rule.required(),
    }),
  ],
  preview: {
    select: {
      title: "name",
      media: "image",
      price: "price",
    },
    prepare(select) {
      return {
        title: select.title,
        subtitle: `$${select.price}`,
        media: select.media,
      };
    },
  },
});
```

2. Order Schema

```
import { BasketIcon } from "@sanity/icons";
import { defineArrayMember, defineField, defineType } from "sanity";

export const orderType = defineType({
  name: "order",
  title: "Order",
  type: "document",
  icon: BasketIcon,
  fields: [
    defineField({
      name: "orderNumber",
      title: "Order Number",
      type: "string",
      validation: (Rule) => Rule.required(),
    }),
    defineField({
      name: "stripeCheckoutSessionId",
      title: "Stripe Checkout Session ID",
      type: "string",
    }),
    defineField({
      name: "stripeCustomerId",
      title: "Stripe Customer ID",
      type: "string",
      validation: (Rule) => Rule.required(),
    }),
    defineField({
      name: "clerkUserId",
      title: "Store User ID",
      type: "string",
      validation: (Rule) => Rule.required(),
    }),
    defineField({
      name: "customerName",
      title: "Customer Name",
      type: "string",
      validation: (Rule) => Rule.required(),
    }),
defineField({
  name: "email",
  title: "Customer Email",
  type: "string",
  validation: (Rule) => Rule.required(),
}),
defineField({
  name: "products",
  title: "Products",
  type: "array",
  of: [
    defineArrayMember({
      type: "object",
      fields: [
        defineField({
          name: "product",
          title: "Product Bought",
          type: "reference",
          to: [{ type: "product" }],
        }),
        defineField({
          name: "quantity",
          title: "Quantity Purchased",
          type: "number",
        }),
      ],
      preview: {
        select: {
          product: "product.name",
          quantity: "quantity",
          image: "product.image",
          price: "product.price",
        },
        prepare(select) {
          return {
            title: `${select.product} X ${select.quantity}`,
            subtitle: `${select.price} * ${select.quantity}`,
            media: select.image,
          };
        },
        },
      }),
    ],
  }),
  defineField({
    name: "totalPrice",
    title: "Total Price",
    type: "number",
    validation: (Rule) => Rule.required().min(0),
  }),
  defineField({
    name: "status",
    title: "Order Status",
    type: "string",
    options: {
      list: [
        { title: "Pending", value: "pending" },
        { title: "Paid", value: "paid" },
        { title: "Shipped", value: "shipped" },
        { title: "Delivered", value: "delivered" },
        { title: "Cancelled", value: "cancelled" },
      ],
    },
  }),
 ],
});
```

Category Schema

```
import { TagIcon } from "@sanity/icons";
import { defineField, defineType } from "sanity";

export const categoryType = defineType({
  name: "category",
  title: "Category",
  type: "document",
  icon: TagIcon,
  fields: [
    defineField({ name: "title", type: "string" }),
    defineField({
      name: "slug",
      type: "slug",
      options: { source: "title" } }),
  defineField({ name: "description", type: "text" }),
  ],
});
```

Sales Schema

```
import { TagIcon } from "@sanity/icons";
import { defineField, defineType } from "sanity";

export const salesType = defineType({
  name: "sale",
  title: "Sale",
  type: "document",
  icon: TagIcon,
  fields: [
    defineField({ name: "title", type: "string", title: "Sale Title" }),
    defineField({ name: "description", type: "text", title: "Sale Description" }),
    defineField({ name: "discountAmount", type: "number", title: "Discount Amount" }),
    defineField({ name: "couponCode", type: "string", title: "Coupon Code" }),
    defineField({ name: "isActive", type: "boolean", title: "Is Active", initialValue: true }),
  ],
});
```