Hackathone Day-2 Date

Name: M. Farooq Rehmani
Roll # 00142495

1- TECHNICAL REQUIREMENT:

- FRONT-END REQUIREMENT:

Front-end user interface has been created in order to facilitate customers with sliders and images of outfit. It is kept in mind to make sure it is user friendly and secure in all aspect. The Next.JS is used to create e-commerce website that is empowered with GraphQL or Rest API.

- SANITY CMS AS BACK-END:

Sanity CMS is used to manage data of products, clients details, cart, checkout process, and other commercial. The sanity Schema will not only manage data but it will be used as a database as well.

- THIRD PARTY API:

The third API will be deployed in a website to ensure it feature will work efficiently. The third party API will cover the segment of shipment tracking, payment gateway, and payment processes.

Date _____

2. DESIGN SYSTEM ARCHITECTURE:

. USER LOG-IN:

User will sign-up through frontend using clerk. The registeration details are stored in Sanity CMS

. PRODUCTS BROWSING

The user navigates through product catagory on the front-end. Sanity CMS API fetches product data (name, price, stock, description, image). Dynamic products display on the front-end

. ORDER PLACEMENT:

On the final round, user adds the choosen product into the cart and proceed to checkout. The order details (product, quantity, shipping address) are saved in Sanity CMS.
The payment is proceed through Stripe and a confirmation message is sent to users email and recorded in Sanity CMS

. SHIPPING TRACKING:

Soon after the order has been placed it will re-flect in our dispatching. Customer can Real-time track his order details.

Date

· INVENTORY PROCESS:

Product stock level is managed in Sanity as this will be working as database. Real-time stock products are fetched from Sanity CMS. Out-of-stock will be added to wish-list instead of cart. Similarly, in-stock products will be added in cart.

· API END-POINTS.

END-POINT	METHOD	PURPOSE	RESPONSE EXAMPLE
/products	GET	Fetch all Product details	[{"name": "Product name", "slug": "Product_Slug", "price": 100}]
/order	POST	Submit new order detail	{"orderId": 123, "status": "success"}
/Shipment tracking	GET	Fetch real-time tracking update	{"trackingId": "ABC123", "status": "In Transit"}
/delivery-status	GET	Fetch express delivery informati	{"orderId": 456, "deliveryTime": "30mins"}
/inventory	GET	Fetch real-time stock-level	{"productId": 789, "stock": 50}.

Date _____

. SANITY SCHEMA EXAMPLE

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

export const productType = def
    name: "product"
    title: "Products"
    type: "document"
    icon: TrolleyIcon
    field: [.
        defineFields ({
        name: "name",
        title: "Product name",
        type: "string",
        validation: (Rule) => Rule.required(),
        }),
        defineField ({
        name: "slug",
        title: "slug",
        type: "slug",
        options: "slug",
            source: "name"
            maxlength: 96, },
            validation: (Rule) => Rule.required(),
        }),
        defineField ({
            name: "image"
            title: "Product image",
            type: "image",
            option: {
                hotspot: true, },
                validation: Rule.required(),
                }),
```

Date

```
defineField({
  name: "AdditionalImage"
  title: "Additional Image"
  type: "array",
  of: "[{ type:"image", options:{hotspot:true}}]
  }),
  defineField({
    name: "description",
    title: "Description",
    type: "blockContent", }),

  defineField({
    name: "price",
    title: "Price",
    type: "number",
    validation: (Rule) => Rule.required.min(0),
    }),
  defineField({
    name: "discountPrice",
    title: "Discount Price",
    type: "number",
    description: "Discount Price of the product (optional)",
    validation: (Rule) =>
    Rule.custome(discountPrice, context) => {
    Const doc = context.document;
    if(doc && typeof doc.price === "number"){
    if(discountPrice && discountPrice >= doc.price){
    return "Discount price must be less than the original Price";
    }}
    return true; }),
    }),
```

Date _____

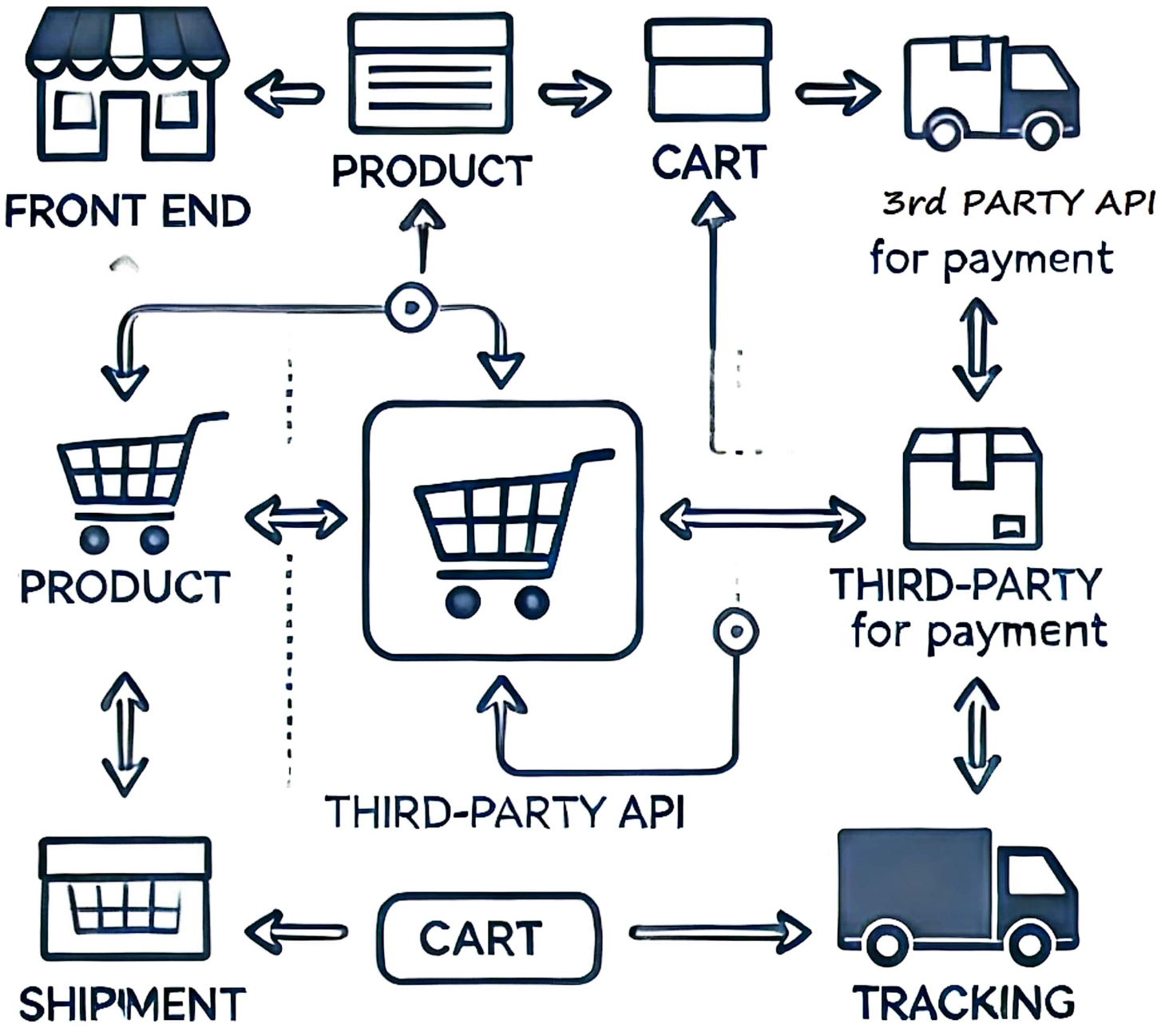
```
defineField ({
  name : "inStock",
  title : "In Stock",
  type : "boolean",
  description: "Indicate whether the product is in stock",
  initialValue : true,
  validation : (Rule) => Rule.required( ), }),

defineField ({
  name: "stock"
  title : "Stock Quantity"
  type : "number",
  description : "Number of item available in-stock",
  Validation : (Rule) => Rule.required.min(0), }),

defineField ({
  name : "rating",
  title : "Rating",
  type : "number",
  description: "Number of reviews of the product",
  validation : (Rule) => Rule.min(0).max(5).precision(1
                , })
  defineField ({
  name : "review",
  title : "Review Count",
  type : "number",
  description: "Number of reviews of the product",
  validation : (Rule) => Rule.required ( ).min(0), }),
  ],
```

Date

```
Preview : {
select : {
title : "name",
media: "image",
Subtitle: "price",
inStock : "inStock",
stock : "stock", },
prepare ( { title, subtitle, media, inStock, stock }) {
    return {
        title,
        subtitle : ${subtitle} | ${inStock ? `in Stock
                (${stock})` : "out of Stock" }`,
                media,
    };
    },
    },
});
```


FRONT END
PRODUCT
CART
3rd PARTY API
for payment
PRODUCT
THIRD-PARTY
for payment
THIRD-PARTY API
SHIPMENT
CART
TRACKING