HACKATHON.

Dated: DAY 2 - PLANNING THE TECHNICAL FOUNDATION:

DAY 2 - GOAL: TRANSITIONING TO TECHNICAL PLANNING.

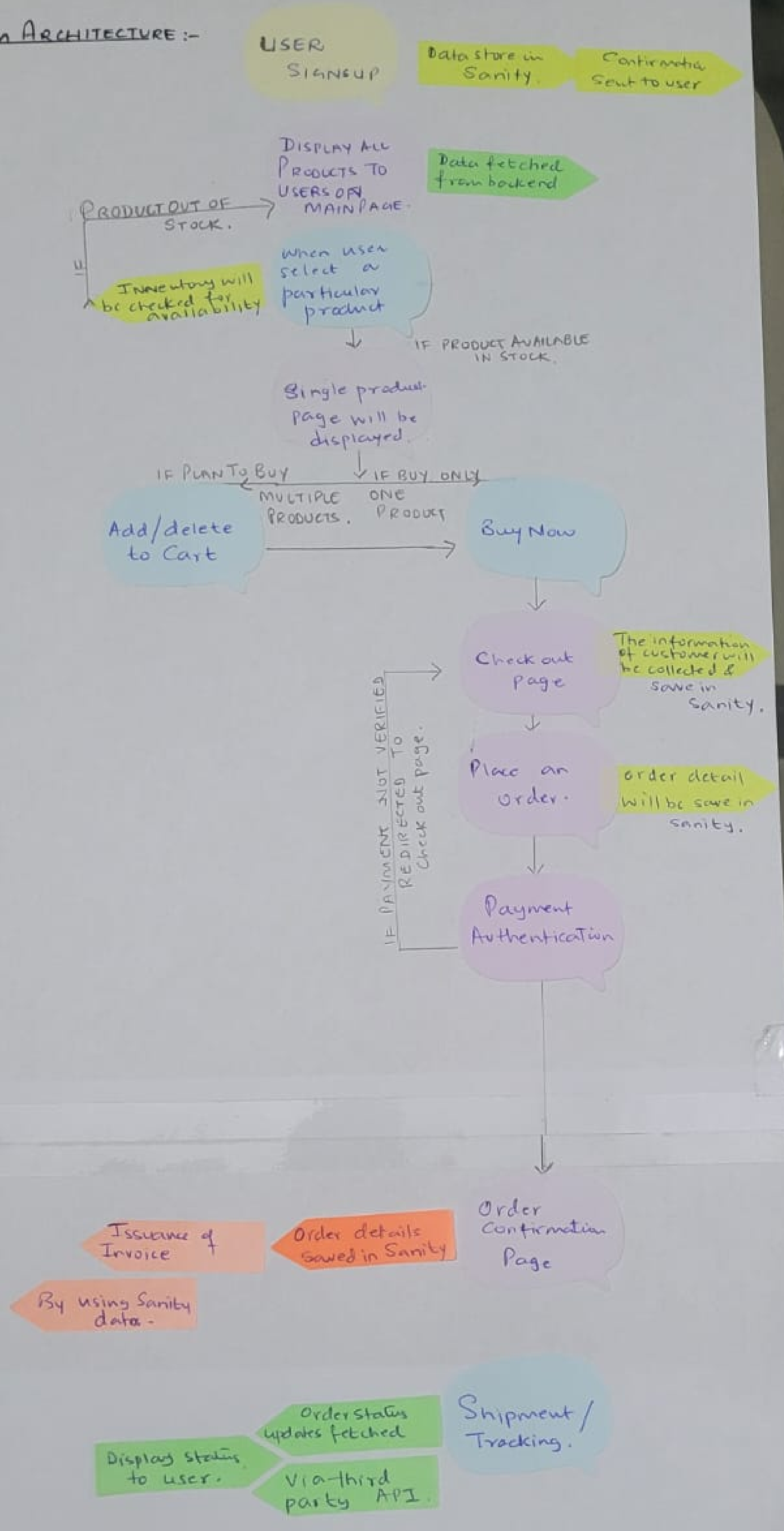
1. FRONT END REQUIREMENTS:

For my E-commerce project I shall use Next js 14, react, Typescript and css tailwind. The design of the website would be responsive from mobile to laptop. The website would have essential pages of Home, Shop, single product page, Cart, Checkout, Confirmation page, Invalid credentials of payment page, Shipment/tracking page, shipment delayed page/shipment returned page, Delivery status/Thanks/customer loyalty card page.

2. SANITY CMS as Backend:- I will use Sanity.io as content management system. Sanity will act as a database for our e-commerce Furniro store.

Third party API:

SYSTEM ARCHITECTURE :-


USER SIGNUP
Data store in Sanity.
Confirmation Sent to user
DISPLAY ALL PRODUCTS TO USERS ON MAIN PAGE.
Data fetched from backend
PRODUCT OUT OF STOCK.
When user select a particular product
Inventory will be checked for availability
IF PRODUCT AVAILABLE IN STOCK.
Single product page will be displayed.
IF PLAN TO BUY MULTIPLE PRODUCTS.
IF BUY ONLY ONE PRODUCT
Add/delete to Cart
Buy Now
Check out page
The information of customer will be collected & save in Sanity.
Place an order.
order detail will be save in sanity.
IF PAYMENT NOT VERIFIED REDIRECTED TO Check out page.
Payment Authentication
Order Confirmation Page
Order details saved in Sanity
Issuance of Invoice
By using Sanity data.
Shipment / Tracking.
Order status updates fetched
Via third party API.
Display status to user.

Dated:

3. PLAN API REQUIREMENTS:-

GENERAL E-COMMERCE PROJECT:

END POINT: / products.
Method : GET.
Description: Fetch all product details.

END POINTS: / orders. Method: POST
Description: Product id, order id, customer info, payment status

END POINTS. / shipment Method: GET.
Description: shipment id, order id, order status - Track order status via third party API.
END POINTS: / customer Method:- POST + GET

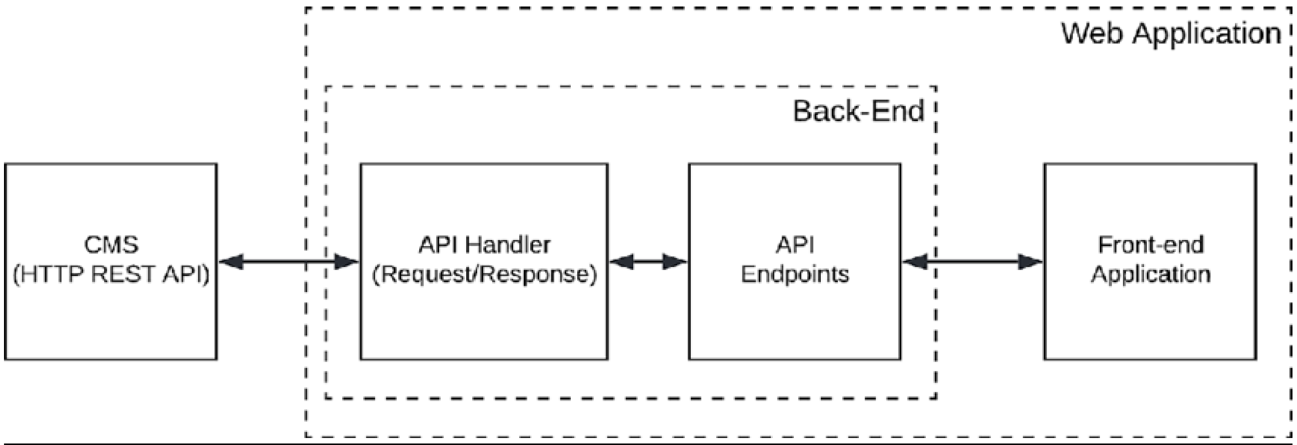
End points: / inventory Method:-

System Architectural Document

“Market place Technical Foundation-[E-Commerce store Furniro Furnishers]”

1-System Agricultural Overview:

Please see the diagram showing how front end interacts with sanity CMS & third-party API.



Please see the brief description about the following terms

Front-End:

Front-end is the user interface and this term is used interchangeably as client-side or browser depending upon the situation.

Sanity: I have already database in Sanity and more records will be added manually. Each time the data change in Sanity it would be updated by running a simple command i.e.

npm create sanity@latest -- --project 0jizi7pu --dataset production --template clean

API:Api stands for application programing interface. Api is in url form. Our local server may be our api. Whenever we hit on the Api we get the message. There are five methods of api through which we can send/receive data to and from server.

1. GET: It retrieves data from the server/backend.
2. POST: Send data to the server to create a new resource.
3. DELETE: Remove a resource from the server.

4. PUT: Update an existing resource to create a new resource
5. PATCH:It partially update and existing resource.

In the first phase the first two methods will be used in coding and then after we will use delete in second phase.

2-Key Work Flows:

User Work flow: (step by step interaction of user with different components:

3-Category Specific Instructions:

General E-Commerce:

In a rapidly changing world the need of trusted market place is growing.

The local market places of Daraz and Darwaza is still trying to create its place due to lack of trust by customers. The concept of selling furnishers online is not a new concept but defining the requirement of the customers through a new angle can change the game. There we are to introduce state of the art furnisher which is compact, stylish and economical at the same time.

4-API Endpoints:

Initially my api end points would be as follows:

1. product
   http://localhost:3000/studio/structure/product

2. order

   http://localhost:3000/studio/structure/order

3. customer
   http://localhost:3000/studio/structure/customer

Keeping in view of the requirement at later stage I will add a new resource of “inventory” and “delivery zone” to stream line the product inflow and outflow.

3-Sanity Schema:

For product:

```
export default {
    name: 'product',
    title: 'Product',
    type: 'document',
    fields: [
      {
        name: 'productId',
        title: 'Product Id',
        type: 'string',

      },

      {
        name: 'productName',
        title: 'Product Name',
        type: 'string',
        /
      },

      {
        name: 'productDimension',
        title: 'Product Dimension',
        type: 'object',
        fields: [
          {
            name: 'width',
            title: 'Width',
            type: 'number'
          },
          {
            name: 'height',
            title: 'Height',
            type: 'number'
          },
          {
            name: 'depth',
            title: 'Depth',
            type: 'number'
          }
        ]

      },

      {
        name: 'slug',
        title: 'Slug',
        type: 'slug',
        options: {
          source: 'productName',
          maxLength: 200
        },

      },
      {
        name: 'price',
        title: 'Price',
        type: 'number',

      },
      {
        name: 'fullPrice',
        title: 'Full Price',
        type: 'number',

      },
      {
        name: 'image',
        title: 'Image',
        type: 'image',
        options: {
          hotspot: true
        },

      },
      {
        name: 'color',
        title: 'Color',
        type: 'string',

      },
      {
        name: 'quantity',
        title: 'Quantity Sold',
        type: 'number',

      }
    ]
  }
```

For order:

```
export default {
    name: 'order',
    title: 'Order',
    type: 'document',
    fields: [
      {
        name: 'orderId',
        title: 'Order ID',
        type: 'string',

      },

      {
        name: 'productId',
        title: 'Product ID',
        type: 'string',

      },

      {
        name: 'stock',
        title: 'Stock',
        type: 'number',
        /
      },

      {
        name: 'quantitySold',
        title: 'Quantity Sold',
        type: 'number',
        // validation: Rule => Rule.required().min(1).max(1000)
      },

      {
        name: 'paymentStatus',
        title: 'Payment Status',
        type: 'string',
        options: {
          list: [
            { title: 'Pending', value: 'pending' },
            { title: 'Completed', value: 'completed' },
            { title: 'Failed', value: 'failed' }
          ],
          layout: 'radio'  // Makes it a radio button group
        },
        // validation: Rule => Rule.required().min(1).max(50)
      },

      {
        name: 'productDetail',
        title: 'Product Detail',
        type: 'text',

      },

      {
        name: 'customerName',
        title: 'Customer Name',
        type: 'string',

      },
      {
        name: 'shippingAddress',
        title: 'Shipping Address',
        type: 'object',
        fields: [
          {
            name: 'houseAddress',
            title: 'House Address',
            type: 'string',

          },
          {
            name: 'area',
            title: 'Area',
            type: 'string',
          },
          {
            name: 'city',
            title: 'City',
            type: 'string',
          }
        ]
      },

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

      }
    ]
  }
```

For customer:

```
export default {
    name: 'customer',
    title: 'Customer',
    type: 'document',
    fields: [
      {
        name: 'customerId',
        title: 'Customer ID',
        type: 'string',

      },

      {
        name: 'customerName',
        title: 'Customer Name',
        type: 'string',

      },

      {
        name: 'customerEmail',
        title: 'Customer Email',
        type: 'string',

      },

      {
        name: 'customerAddress',
        title: 'Customer Address',
        type: 'object',
        fields: [
          {
            name: 'houseAddress',
            title: 'House Address',
            type: 'string',
          },
          {
            name: 'area',
            title: 'Area',
            type: 'string',

          },
          {
            name: 'city',
            title: 'City',
            type: 'string',

          }
        ]
      },

      {
        name: 'customerMobileNumber',
        title: 'Customer Mobile Number',
        type: 'string',

      },

      {
        name: 'invoiceNumber',
        title: 'Invoice Number',
        type: 'string',

      }
    ]
  }
```