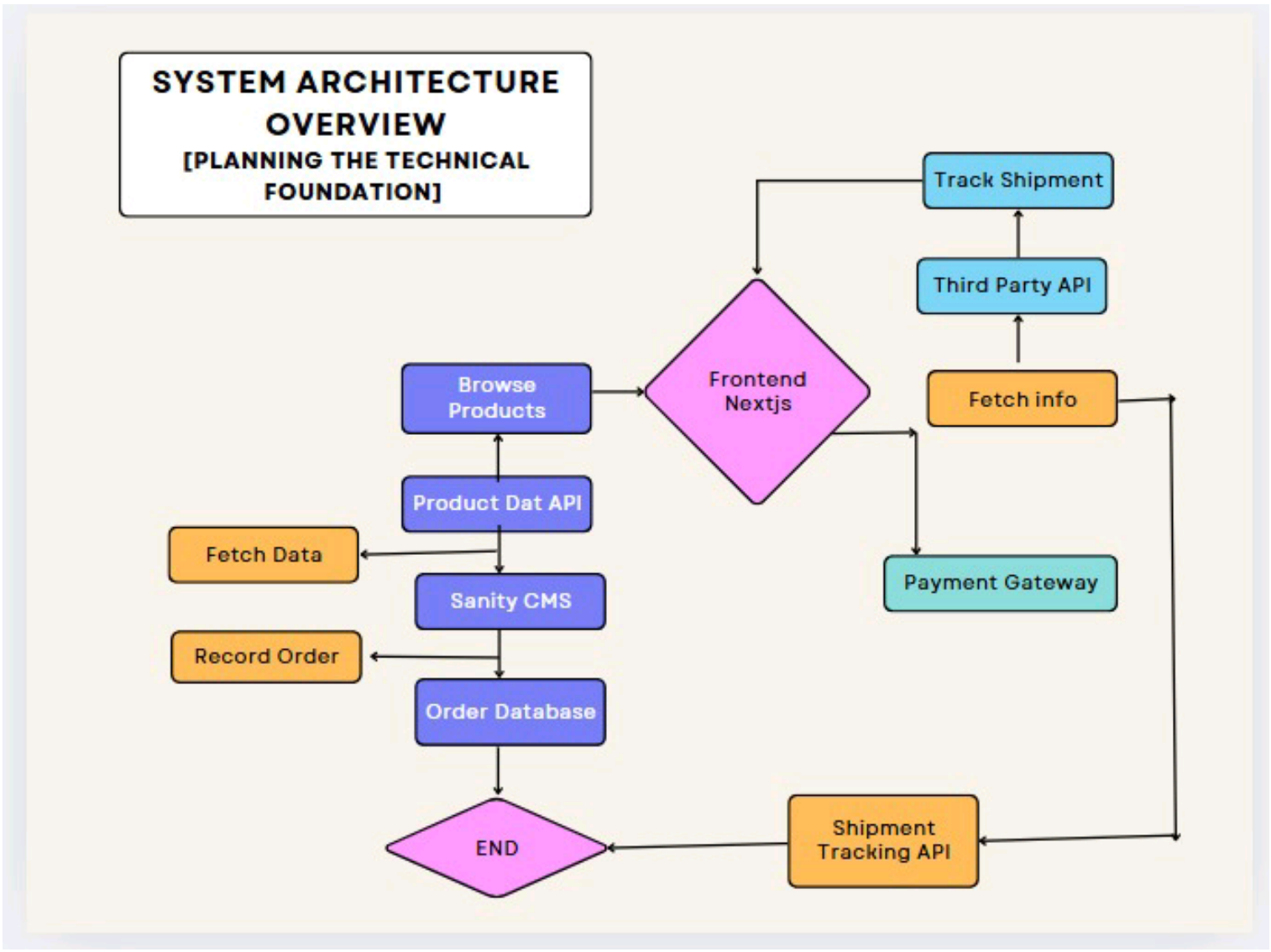
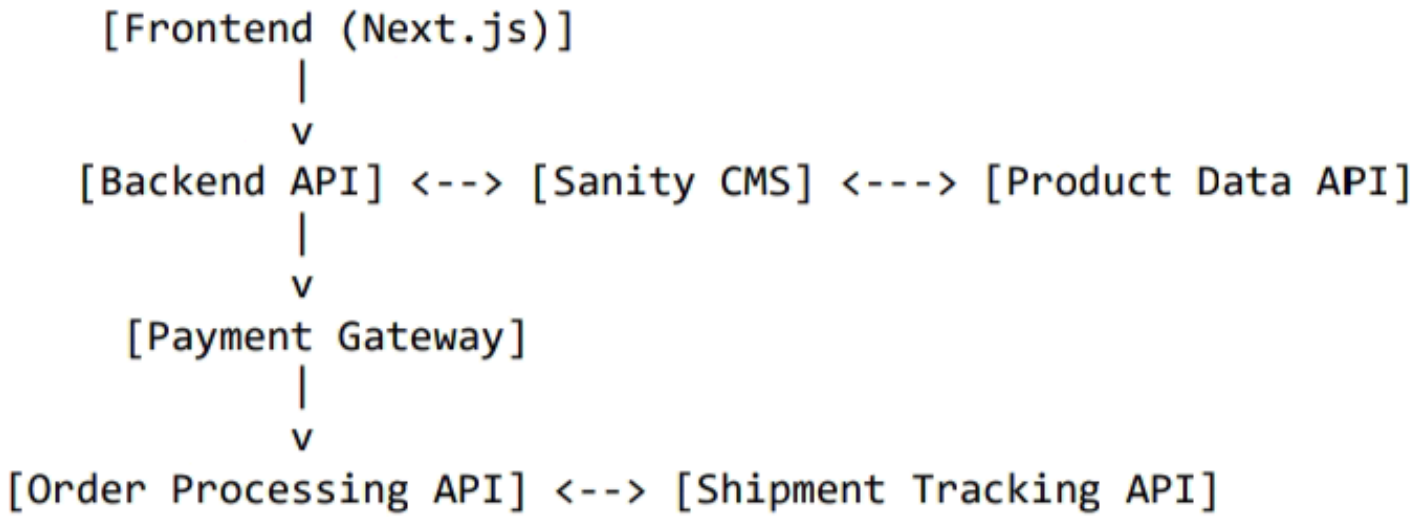
DAY2-
HACKATHON

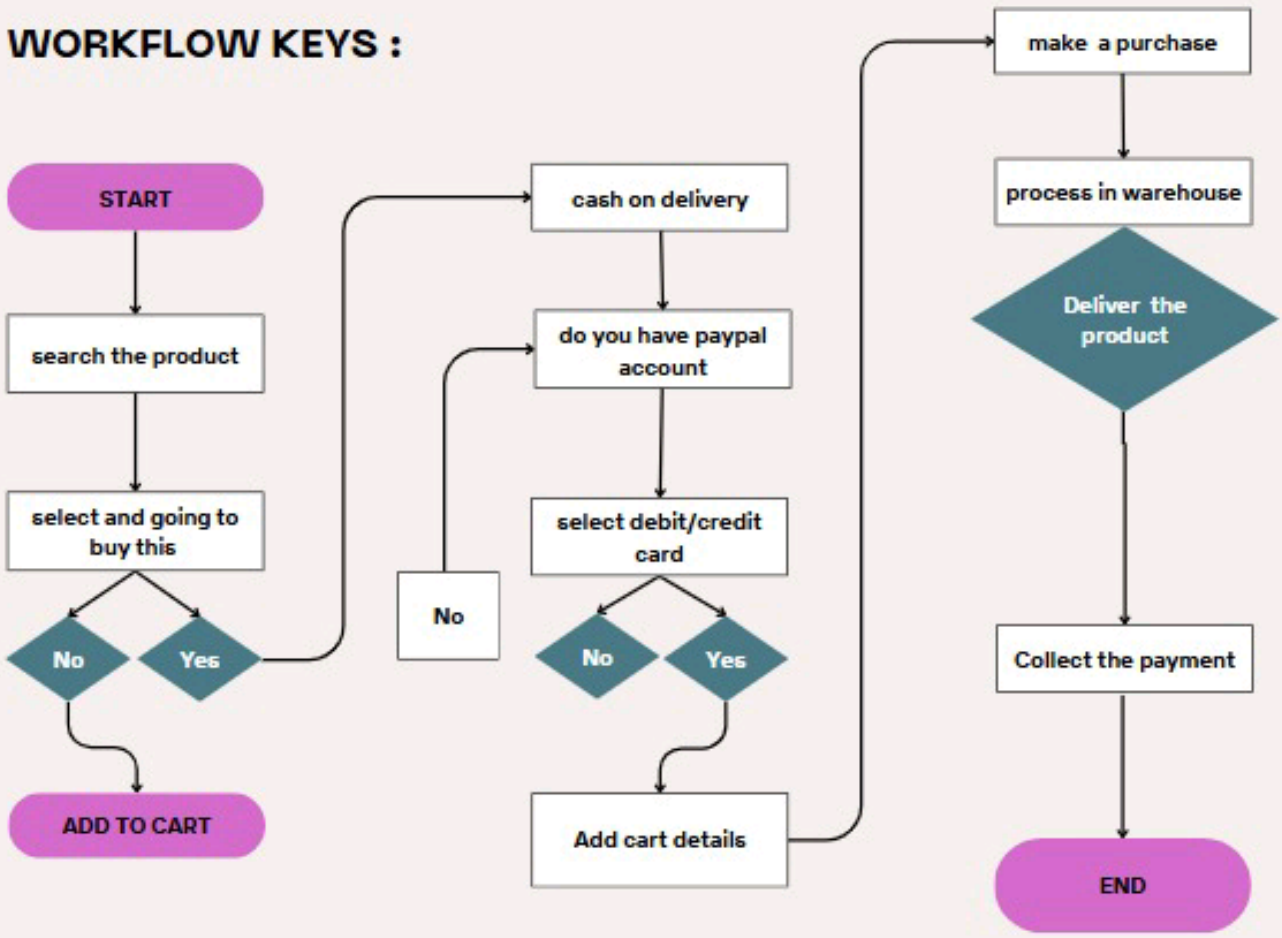
MARKETPLACE TECHNICAL-FOUNDATION [YOUR SHOP.CO MARKETPLACE]

PREPARED BY
SADIA TARIQ

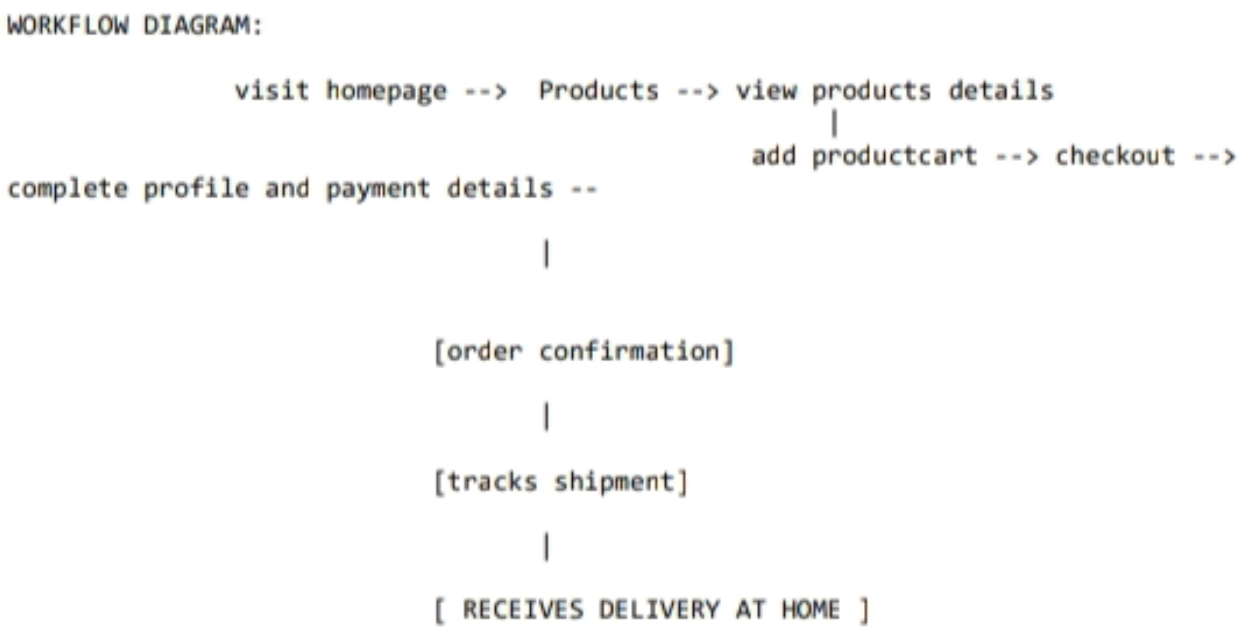

SYSTEM ARCHITECTURE OVERVIEW
[PLANNING THE TECHNICAL FOUNDATION]
Track Shipment
Third Party API
Frontend Nextjs
Browse Products
Fetch info
Product Dat API
Fetch Data
Payment Gateway
Sanity CMS
Record Order
Order Database
END
Shipment Tracking API


[Frontend (Next.js)]
|
v
[Backend API] <--> [Sanity CMS] <---> [Product Data API]
|
v
[Payment Gateway]
|
v
[Order Processing API] <--> [Shipment Tracking API]

WORKFLOW KEYS :



WORKFLOW DIAGRAM:



→ API ENDPOINTS:

ENDPOINT NAME	METHOD	DESCRIPTION	PAYLOAD/QUERY	RESPONSE EXAMPLE
/API/PRODUCTS	GET	Fetch all available products.	None	[{ "id": 1, "name": "Shirt", "price": 1500, "stock": 20 }]
/API/PRODUCTS/:ID	GET	Fetch details of a specific product by ID.	None	{ "id": 1, "name": "Shirt", "price": 1500, "stock": 20 }
/API/PRODUCTS	POST	Add a new product (Admin only).	{ "name": "Shirt", "price": 1500, "stock": 20 }	{ "id": 1, "status": "Product added successfully" }
/API/ORDERS	POST	Create a new order.	{ "customerId": 101, "products": [{ "id": 1, "qty": 2 }], "paymentStatus": "Paid"	{ "orderId": 201, "status": "Order placed successfully" }
/API/ORDERS/:ID	GET	Fetch details of a specific order.	None	{ "orderId": 201, "products": [...], "status": "Processing" }
/API/SHIPMENT	GET	Track order status via third-party API.	{ "orderId": 201 }	{ "shipmentId": 301, "status": "In Transit", "ETA": "15 mins" }

```
export default {
  name: 'product',
  type: 'document',
  title: 'Product',
  fields: [
    {
      name: 'name',
      type: 'string',
      title: 'Product Name',
      validation: (Rule) =>
        Rule.required()
          .max(100)
          .error('Product name is required and cannot exceed 100 characters.'),
    },
    {
      name: 'description',
      type: 'text',
      title: 'Description',
      validation: (Rule) =>
        Rule.required()
          .min(20)
          .max(500)
          .error('Description must be between 20 and 500 characters.'),
    },
    {
      name: 'price',
      type: 'number',
      title: 'Product Price',
      validation: (Rule) =>
        Rule.required()
          .min(0)
          .error('Product price must be a positive value.'),
    },
    {
      name: 'oldPrice',
      type: 'number',
      title: 'Old Price',
      description: 'The original price of the product before any discount.',
      validation: (Rule) =>
        Rule.min(0).error('Old price must be a positive value or zero.'),
    },
    {
      name: 'rating',
      type: 'number',
      title: 'Rating',
      description: 'Average rating of the product.',
      validation: (Rule) =>
```

```
        Rule.min(0)
          .max(5)
          .precision(1)
          .error('Rating must be between 0 and 5.'),
    },
    {
      name: 'sizes',
      type: 'array',
      title: 'Sizes',
      of: [{ type: 'string' }],
      options: {
        layout: 'tags',
      },
    },
    {
      name: 'image',
      type: 'image',
      title: 'Product Image',
      options: {
        hotspot: true,
      },
      validation: (Rule) => Rule.required().error('Product image is required.'),
    },
  ],
};
```