Submission Title: " Day2- Market Technical Foundation (Bandage)

Name: Zakia Begum

Role # : 00053035

Date: 16-01-2025

Introduction :

This document describes the user journey flow for an E Commerce Marketplace,focusing on keys teps andtheir technical implementation. It is tailored for building a robust and user-friendly platform.

Flowchart Overview:

Below is the complete user journey flow for an E-Commerce marketplace:

Home Page:

User lands on the homepage.

Displays featured categories,popular products,and asearch bar.

Product Browsing:

User selects a category or uses the search bar.

Products are displayed with options to filter by price,brand, or rating.

Product Details:

User clicks on aproduct to view its details.

Page includes product description, price, availability, and user reviews.

Add to Cart:

User adds the product to the cart.

Cart updates dynamically with quantity and price.

Checkout:

User proceeds to the checkout page.

Provides shipping address and selects a delivery option.

Payment:

User enters payment details.

Secure payment gateway processes the transaction.

Order Confirmation:

Order details are displayed and sent via email.

Order status is updated in the backend.

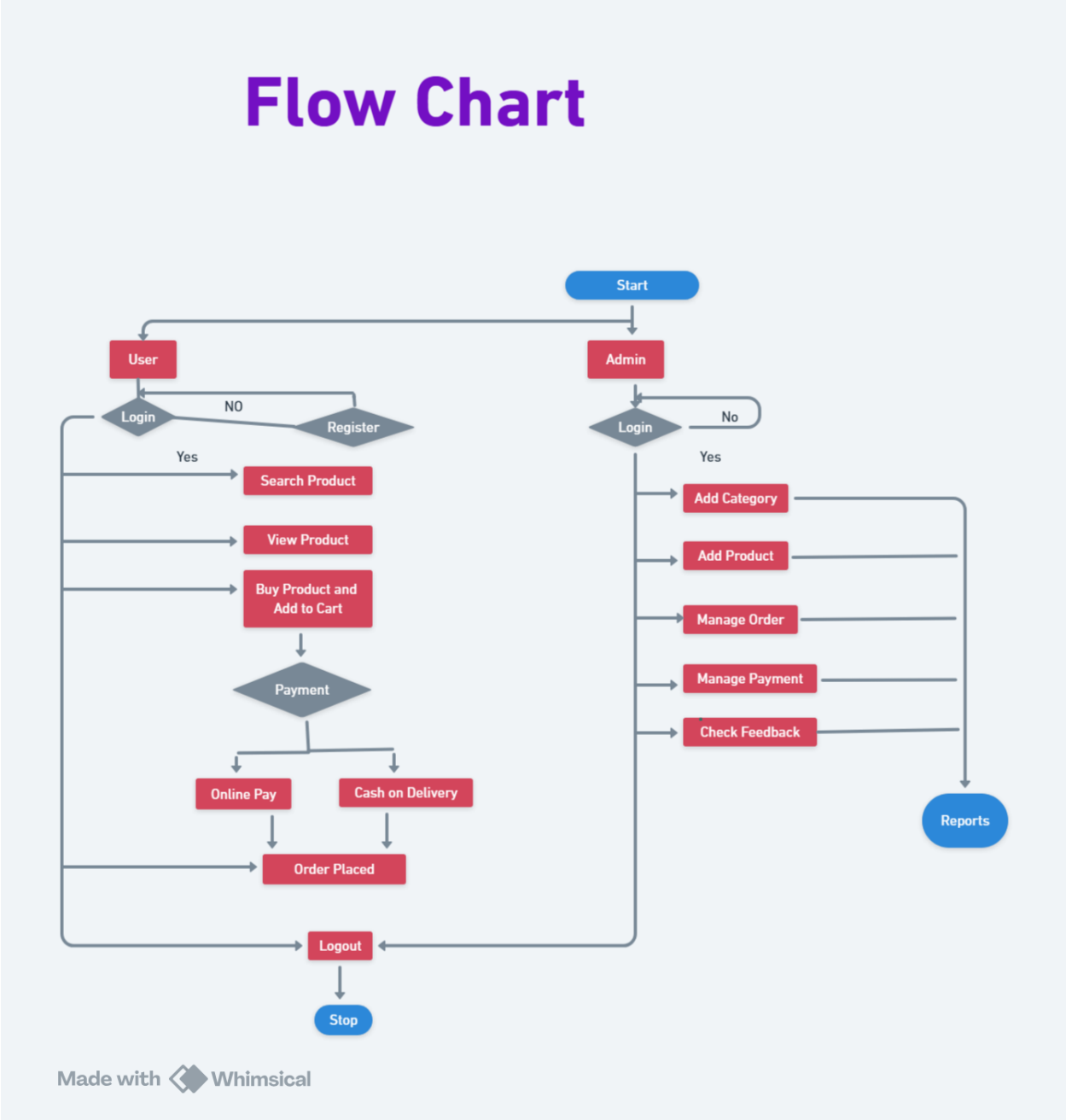
Shipment Tracking:

User visits the "Order History" section.

Real-time shipment tracking is enabled via API integration.

Delivery:

Product is delivered to the user's address.



Frontend Requirements:

User receives a notification and can leave a review.

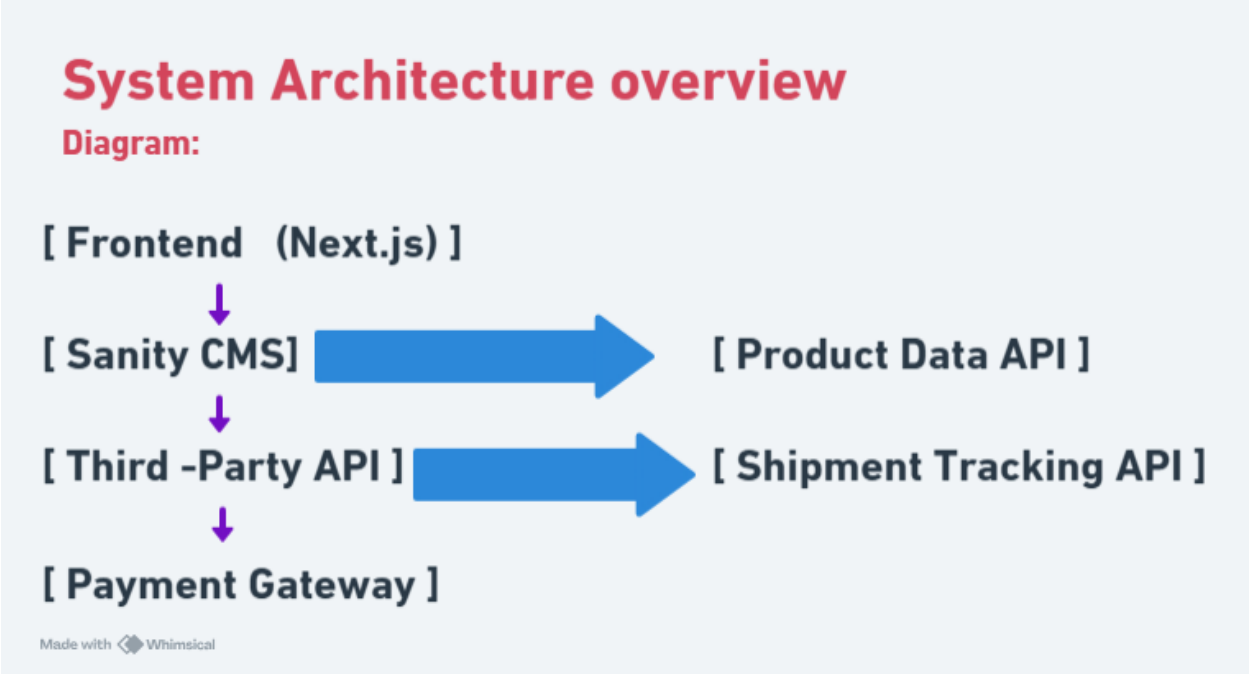
User – friendly interface for browsing products.

Responsive design for mobile and desktop users.

Essentials pages; Home, Product Listing, Product Details, Cart, Checkout, and Order Confirmation.

Design system Architecture:

Use tools ,a more detailed architecture might include workflows such as: Create a high –level diagram showing how your system components interact.



Components and Roles:

Frontend (Next.js):

1. ***Display the user interface for browsing products,managing the cart, and placing orders.***
2. ***Handles user interactions and communicates with backend services via APIs.***

Sanity CMS:

1. ***Acts as the primary backend to manage product data, customer details, and order records.***
2. ***Provides APIs for the fronted to fetch and update data.***

Product Data API:

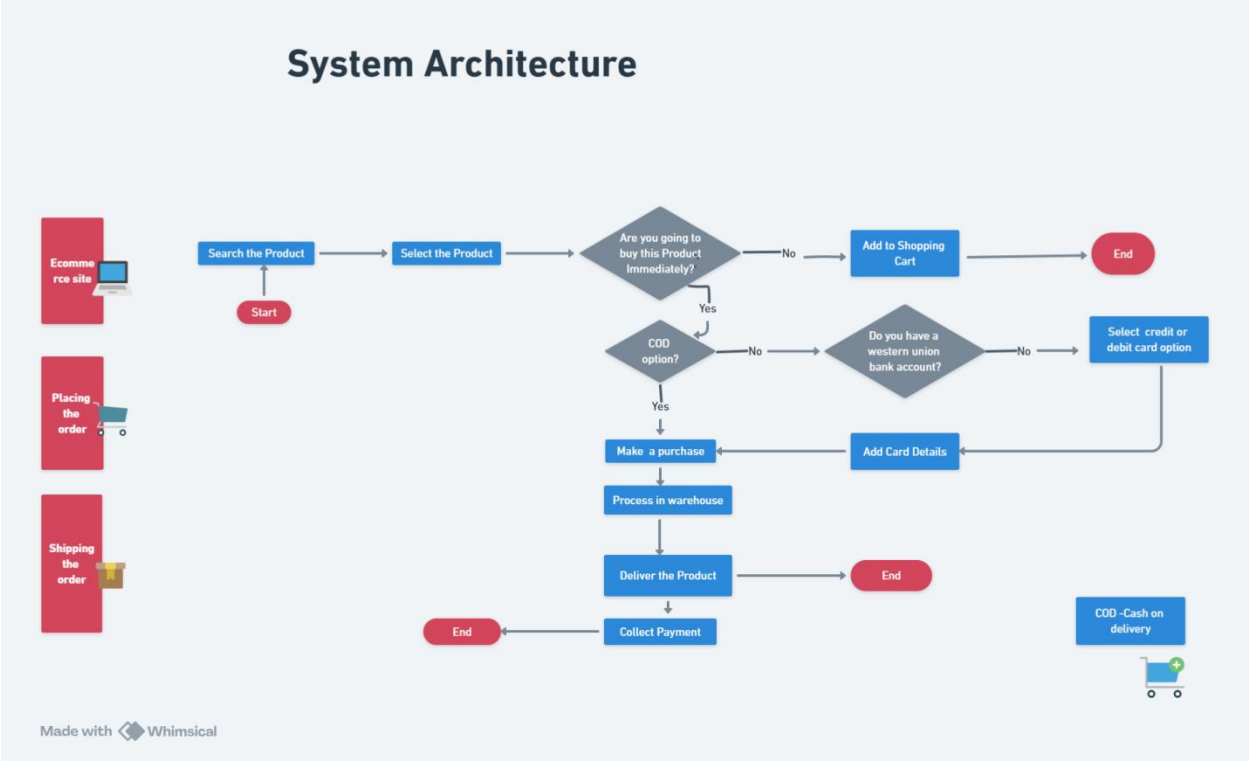
Provides endpoints to fetch product listings, details, and inventory status.

Third –Party APIs:

Integrates services like shipment tracking and payment processing.

Payment Gateway:

Processes user payments securely and provides transaction confirmation.



Also make this EDR diagram in your project.

This diagram will define the relations between entities in my database.

1. Cloth Items:

**Fields:* id, name, price , description, image, categoryId.

2. Categories:

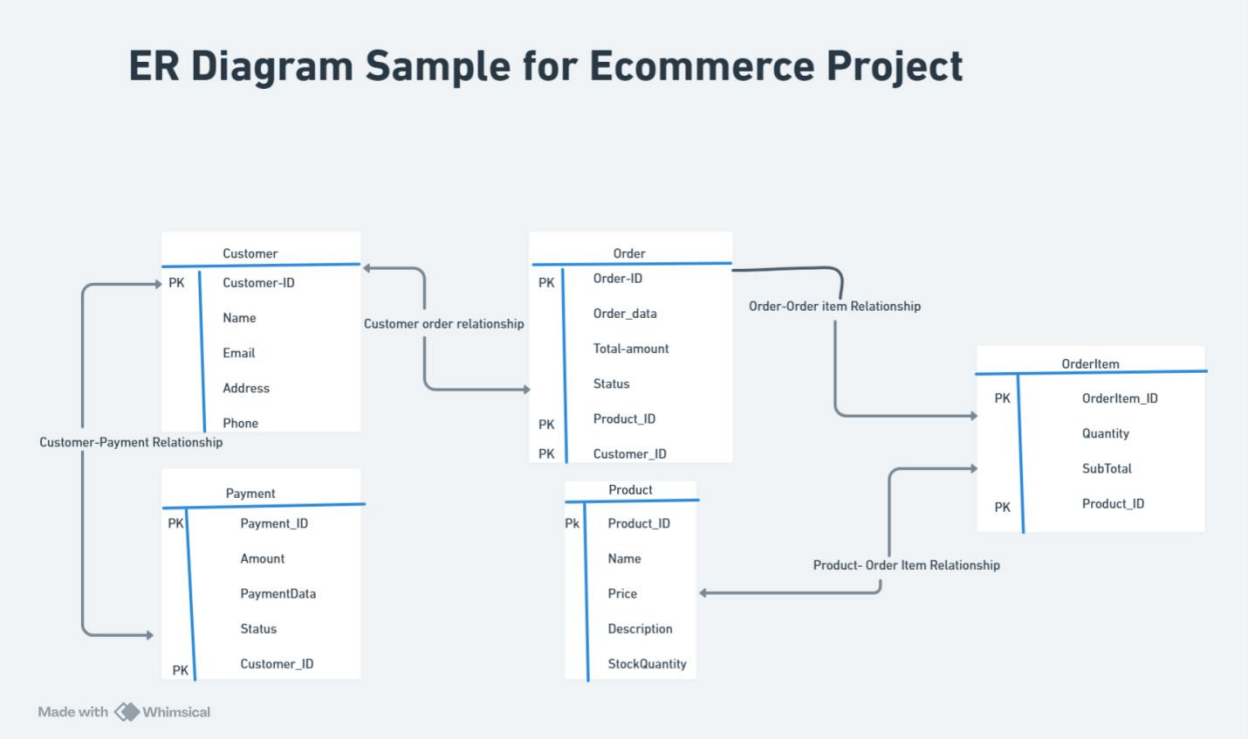
**Fields:* id, name.

3. *Users:*

**Fields*: id, name, email, password.

4. *Orders:*

**Fields*: id, userId, orderDate,status.

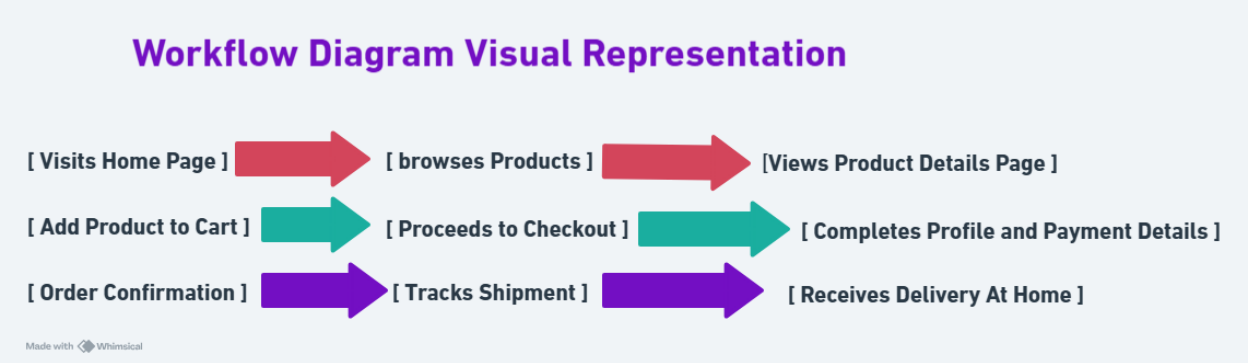


APIEndpoints.xlsx

End points	**Method**	**Description**	**Parameters**	**Response Example**

/api/clothes	GET	Fetch all clothes items	None	{id: 1,name: "Cloth"}
/api/clothes/:id	GET	Fetch a single clothes item	id (Path)	{id: 1,name: "Cloth"}
/api/clothes	POST	Add a new clothes item	name,price,category (Body)	{ success: true, id; 5}
/api/clothes/:id	PUT	Update a cloth item	id (Path),name,price (Body)	{ success: true }
/api/clothes/:id	DELETE	Delete a cloth item	id (Path)	{ success: true }
/api/caregories	GET	Fetch all cloth categories	None	{ categories: [" "] }

Work flow diagram:



Clothes Sanity Schema:

```
    export default {
name: 'clothingItem',
title: 'Clothing Item',
type: 'document',
fields: [
 {
  name: 'title',
  title: 'Title',
  type: 'string',
  description: 'Name of the clothing item',
  validation: (Rule) => Rule.required().min(3).max(50),
 },
 {
  name: 'description',
  title: 'Description',
  type: 'text',
  description: 'Detailed description of the item',
  validation: (Rule) => Rule.required().min(10).max(500),
 },
 {
  name: 'price',
  title: 'Price',
  type: 'number',
  description: 'Price of the item in your currency',
  validation: (Rule) => Rule.required().min(0),
 },
 {
```

```
  name: 'image',
  title: 'Image',
  type: 'image',
  options: {
    hotspot: true,
  },
  description: 'Upload an image of the clothing item',
  validation: (Rule) => Rule.required(),
},
{
  name: 'size',
  title: 'Size',
  type: 'array',
  of: [{ type: 'string' }],
  options: {
    list: [
      { title: 'Small', value: 'S' },
      { title: 'Medium', value: 'M' },
      { title: 'Large', value: 'L' },
      { title: 'Extra Large', value: 'XL' },
    ],
  },
  description: 'Available sizes',
},
{
  name: 'gender',
  title: 'Gender',
    list: [
```

```
      { title: 'Male', value: 'male' },
      { title: 'Female', value: 'female' },
      { title: 'Unisex', value: 'unisex' },
    ],
  },
  description: 'Target gender for the clothing item',
  validation: (Rule) => Rule.required(),
},
{
  name: 'category',
  title: 'Category',
  type: 'string',
  options: {
    list: [
      { title: 'Shirts', value: 'shirts' },
      { title: 'Pants', value: 'pants' },
      { title: 'Dresses', value: 'dresses' },
      { title: 'Outerwear', value: 'outerwear' },
      { title: 'Accessories', value: 'accessories' },
    ],
  },
  description: 'Category of the clothing item',
  validation: (Rule) => Rule.required(),
},
{
  name: 'stock',
  title: 'Stock',
  type: 'number',
```

```
    description: 'Number of items available in stock',
    validation: (Rule) => Rule.required().min(0),
   },
   {
    name: 'sku',
    title: 'SKU',
    type: 'string',
    description: 'Stock Keeping Unit (unique identifier for the
item)',
    validation: (Rule) => Rule.required().min(3).max(20),
   },
  ],
};
```