This document describes the user journey flow for an eCommerce marketplace, focusing on key steps and their

technical implementation. It is tailored for building a robust and user-friendly platform.

Flowchart Overview

Below is the complete user journey flow for an eCommerce marketplace:

Home Page:

User lands on the homepage.

Displays featured categories, popular products, and a search bar.

Product Browsing:

User selects a category or uses the search bar.

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

Product Details:

User clicks on a product 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.

Essential pages: Home, Product Listing, Product Details, Cart, Checkout, and

Order Confirmation

Design System Architecture

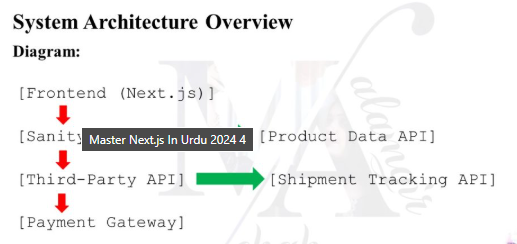
Create a high-level diagram showing how your system components interact Use

tools

like pen and paper or software like Lucidchart, Figma or Excalidraw For

example, a

more detailed architecture might include workflows such as:



**Components and Roles:**

Frontend (Next.js):

1. Displays 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 frontend to fetch and update data.

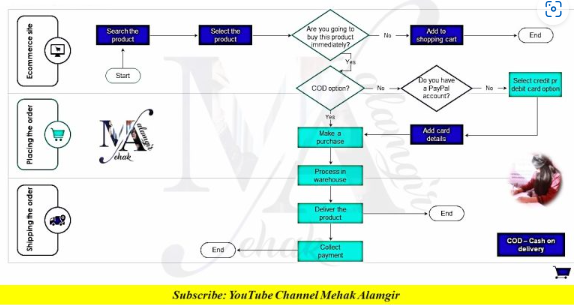
Product Data API:

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

Third-Party APIs:

Payment Gateway:

Processes user payments securely and provides transaction confirmation.



Also make this EDR diagram in your

project.

This diagram will define the relationships between entities in your database.

Example Entities:

1. Food Items:

o Fields

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

2. Categories:

o Fields

: id, name

3 Users:

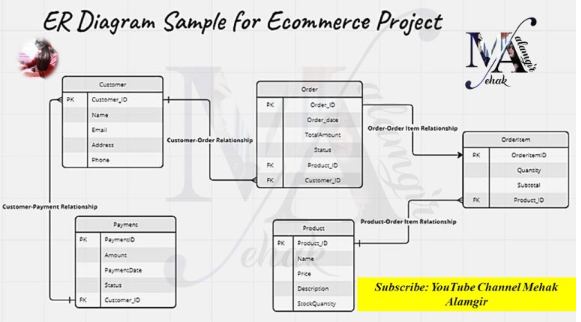
o Fields

: id, name, email, password.

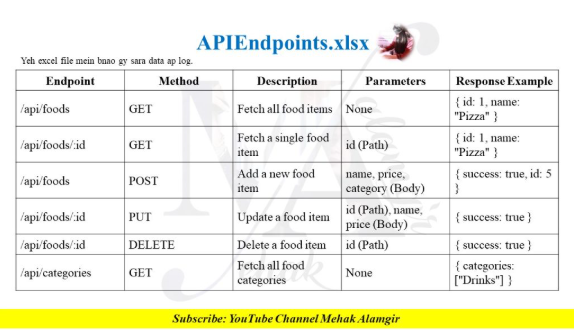
4. Orders:

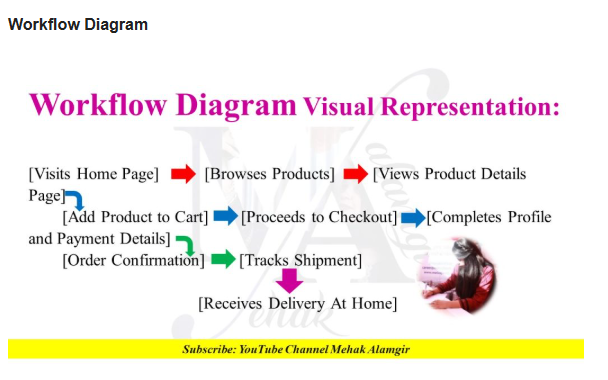
o Fields

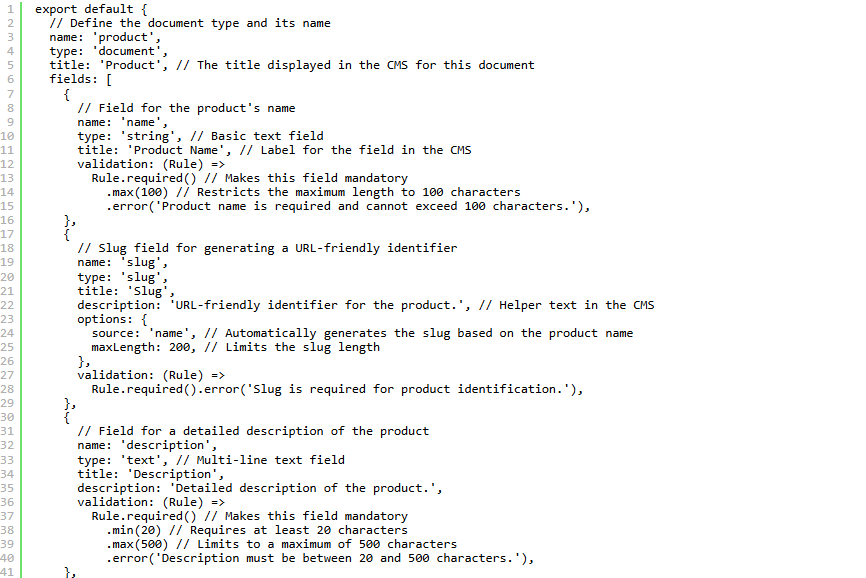
id, userld, orderDate, status

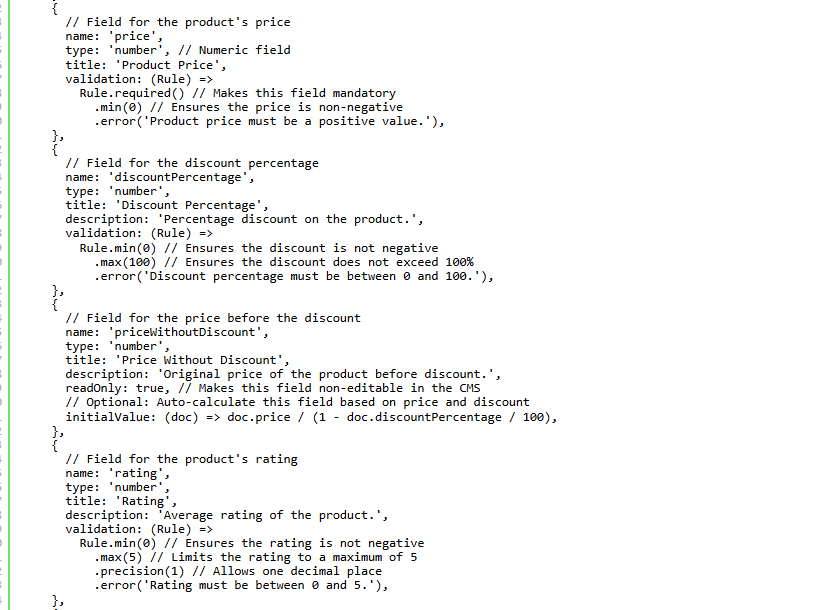


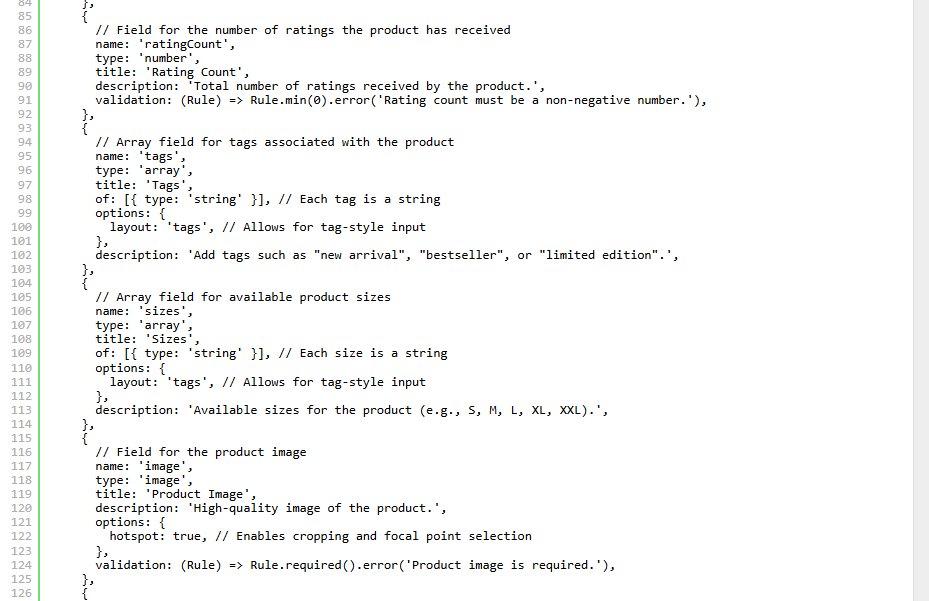
**APIEndpoints.xIsx**

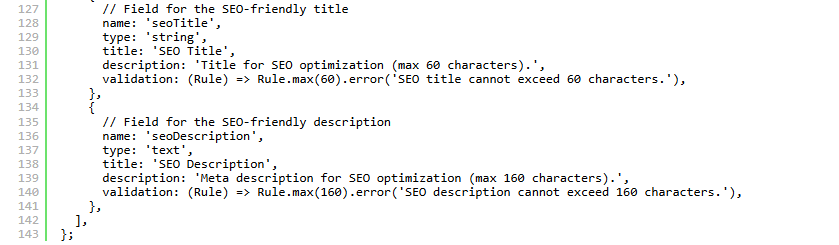


****

****





****