"Marketplace Technical Foundation- [General E-commerce]"

This document describes the user journey flow for an e-commerce 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 e-commerce 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:

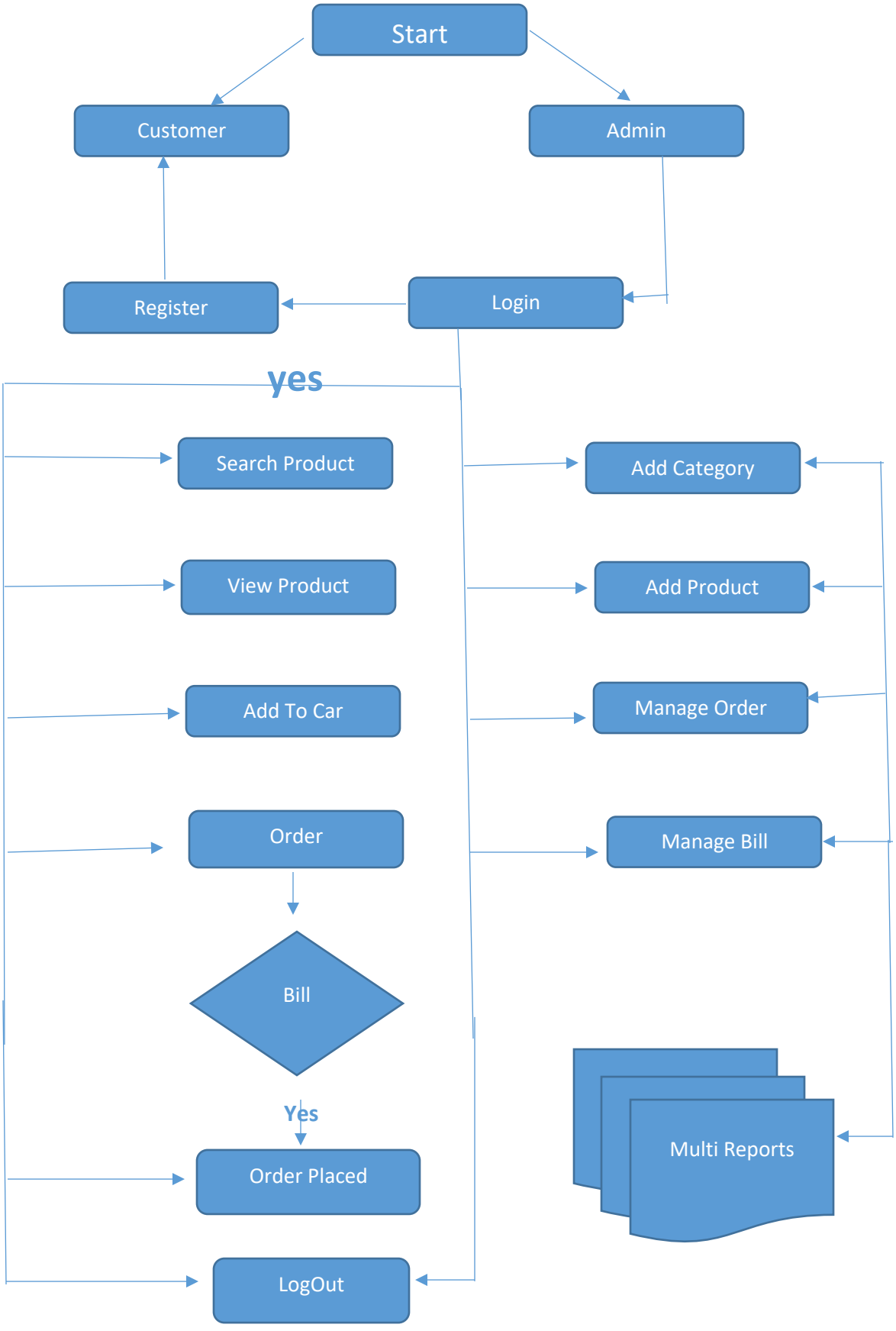
User visits the "Order History" section.

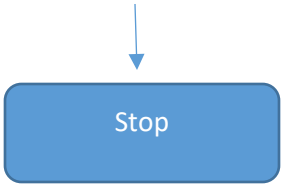
Real-time shipment tracking is enabled via APl integration.

Delivery:

Product is delivered to the user's address.

Frontend Requirements: Flow Chart By Nida Iqbal





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, as

System Architecture Overview

Diagram:

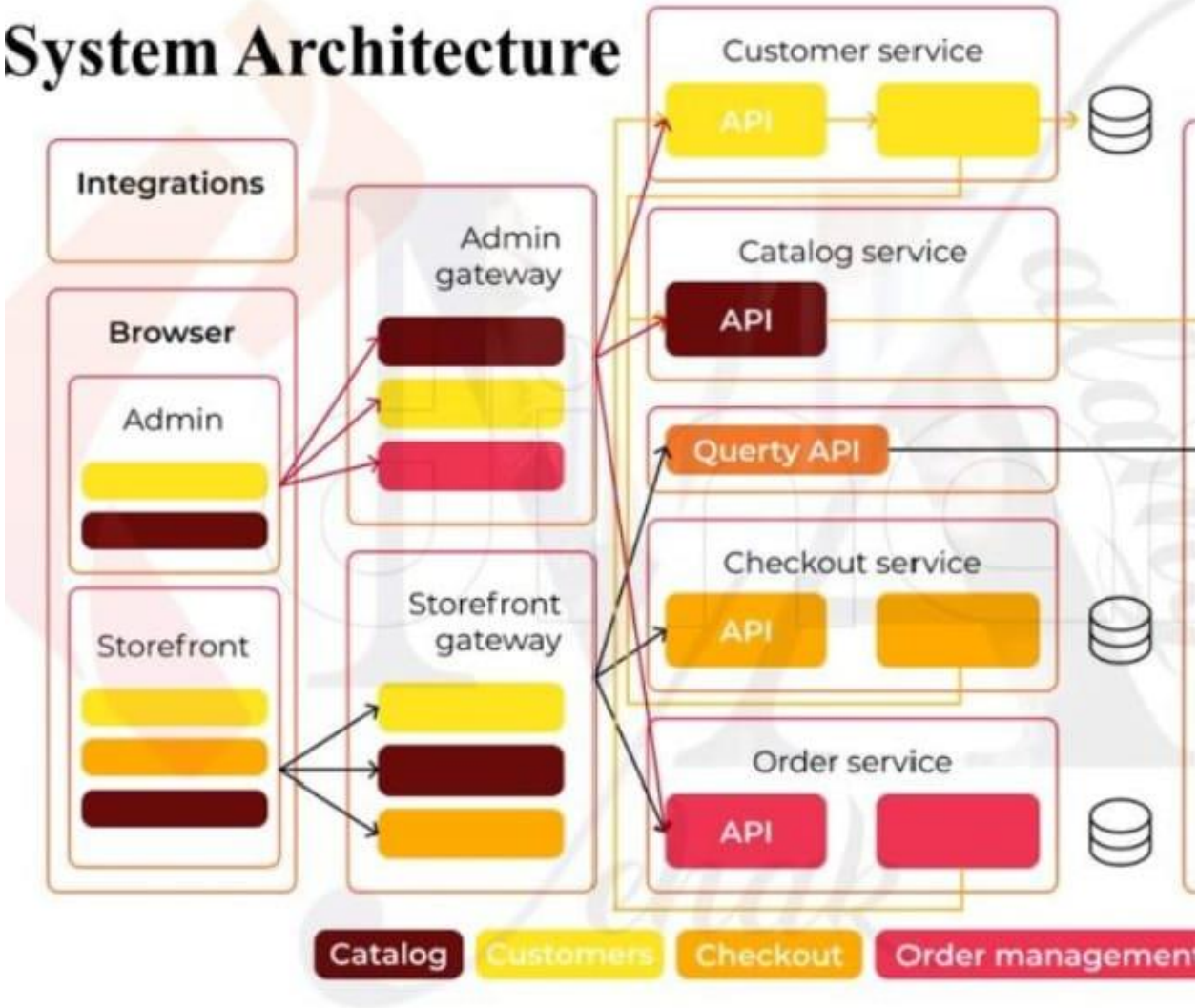
[Frontend (Next.js)]

[Sanity CMS] ⟶ [Product Data API]

[Third-Party API] ⟶ [Shipment Tracking API]

[Payment Gateway]

System Architecture



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:

Integrates services like shipment tracking and payment processing.

Payment Gateway:

Processes user payments securely and provides transaction confirmation.

Example Entities:

1. **Food 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

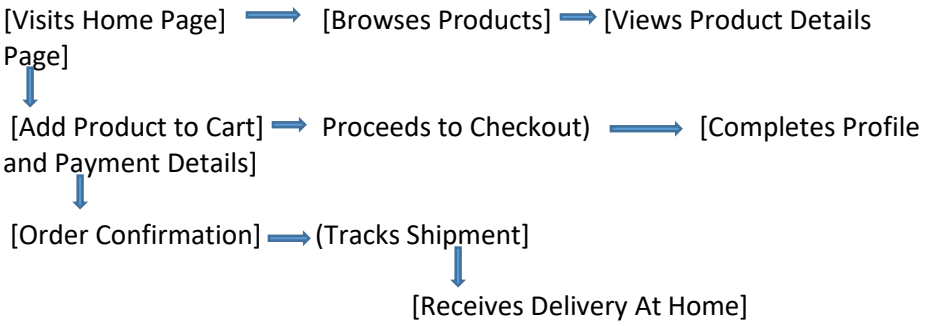
APIEndpoints.xlsx

APIEndpoints.xlsx

Endpoint	Method	Description	Parameters	Response Example
/api/foods	GET	Fetch all food	None	(id: 1, name: "Pizza" }
/api/foods/:id	GET	Fetch a single food item	id (Path)	(id: 1, name: "Pizza" }
/api/foods	POST	Add a new food item	name, price, category (Body)	{success: true. id: 5}
/api/foods/:id	PUT	Update a food item	id (Path), name, price (Body)	{success: true}
/api/foods/:id	DELETE	Delete a food item	id (Path),	{success: true}
/api/categories	GET	Fetch all food categories	None	{categories [“Drinks"]}

Workflow Diagram

Workflow Diagram Visual Representation:



SanitySchema.js

```
export default {
    // Define the document type and its name
    name: 'product',
    type: 'document',
    title: 'Product', // The title displayed in the CMS for this document
    fields: [
        {
            // Filed for the product's name
            name: 'name',
            type: 'string', // Basic text field
            title: 'Product Name', // Label for the field in the CMS
            validation: (Rule) =>
                Rule.required() // Make this field mandatory
                .max(100) // Restricts the maximum length to 100 characters
                .error('Product name is required and cannot exceed 100
characters.')
        },
        {
            // Slug field for generating a URL-friendly identifier
            name: 'slug',
            type: 'slug',
            title: 'Slug',
            description: 'A URL-friendly name for this product.', // Help text in
            options: {
                source: 'name', // Slug is generated from the product name
                maxLength: 200, // Limit the slug length
            },
            validation: (Rule) =>
                Rule.required().error('Slug is required for product
identification.'),
        },
        {
            // Field for the product's description
            name: 'description',
            type: 'text', // Multi-Line text field
            title: 'Product Description',
            description: 'A detailed description of the product.',
            validation: (Rule) =>
                Rule.required() // Make this field mandatory
                .min(20) // Minimum length of 20 characters
                .max(500) // Maximum length of 500 characters
```

```
                .error('Product description is required and must be between 20
and 500 characters.'),
        },
        {
            // Field for the product's price
            name: 'price',
            type: 'number', // Field for a numerical value
            title: 'Product Price',
            validation: (Rule) =>
                Rule.required() // Make this field mandatory
                .min(0) // Ensure the price is non-negative
                .error('Product price must be a positive number.'),
        },
        {
            // Field for teh discount percentage
            name: 'discountPercentage',
            type: 'number',
            title: 'Discount Percentage',
            description: 'Percentage discount on the product.',
            validation: (Rule) =>
                Rule.min(0) // Ensure the discount is not negative
                .max(100) // Ensure teh discount is not negative
                .error('Discount percentage must be between 0 and 100.'),
        },
        {
            // Field for the price before the discount
            name: 'priceWithoutDiscount',
            type: 'number',
            title: 'Price Without Discount',
            description: 'Original price of the product before the discount.',
            readOnly: true, // Make this field non-editable in the CMS
                // Optional: Auto-calculate this field based on price and
discount
                initialValue: (doc) => doc.price / (1 - doc.discountPercentage /
100),
        },
        {
            // Field for the product's rating
            name: 'rating',
            type: 'number',
            title: 'Rating',
            description: 'Average rating of the product.',
            validation: (Rule) =>
                Rule.min(0) // Ensure the rating is not negative
                .max(5) // Ensure the rating is not more than 5
```

```
                .precision(1) // Allow one decimal place
                .error('Rating must be between 0 and 5.'),
        },
        {
            // Field for the number of ratings the product has received
            name: 'ratingCount',
            type: 'number',
            title: 'Rating Count',
            description: 'Total Number of ratings the product has received.',
            validation: (Rule) => Rule.min(0).error('Rating count must be a non-
negative'),
        },
        {
            // Array field for tags associated with the product
            name: 'tags',
            type: 'array',
            title: 'Tags',
            of: [{ type: 'string' }], // Each tag is a string
                option: {
                    layout: 'tags', // Allow for tag-Style input
                },
            description: 'Add tags such as "new arrival", "bestseller", or
"limited edition" '
        },
        {
            // Array field for available product sizes
            name: 'sizes',
            type: 'array',
            title: 'Sizes',
            of: [{ type: 'string' }], // Each size is a string
                option: {
                    layout: 'tags', // Allows for tag-style input
                },
            description: 'Available sizes for the product (e.g., S, M, L, XL,
XXL).',
        },
        {
            // Field for the product image
            name: 'image',
            type: 'image',
            title: 'Product Image',
            description: 'High-quality image of the product.',
            option: {
                hotspot: true, // Enables cropping and focal point selection
            },
```

```
            validation: (Rule) => Rule.required().error('Product image is
required'),
        },
        {
            // Field for the SEO-friendly title
            name: 'seoTitle',
            type: 'string',
            title: 'SEO Title',
            description: 'Title for SEO optimization (max 60 characters).',
            validation: (Rule) => Rule.max(60).error('SEO title must be 60
characters'),
        },
        {
            // Field for the SEO-friendly description
            name: 'seoDescription',
            type: 'text',
            title: 'SEO Description',
            description: 'Meta description for SEO optimization (max 160
characters).',
            validation: (Rule) => Rule.max(160).error('SEO description can not be
exceed 160 characters'),
        },
    ],
};
```