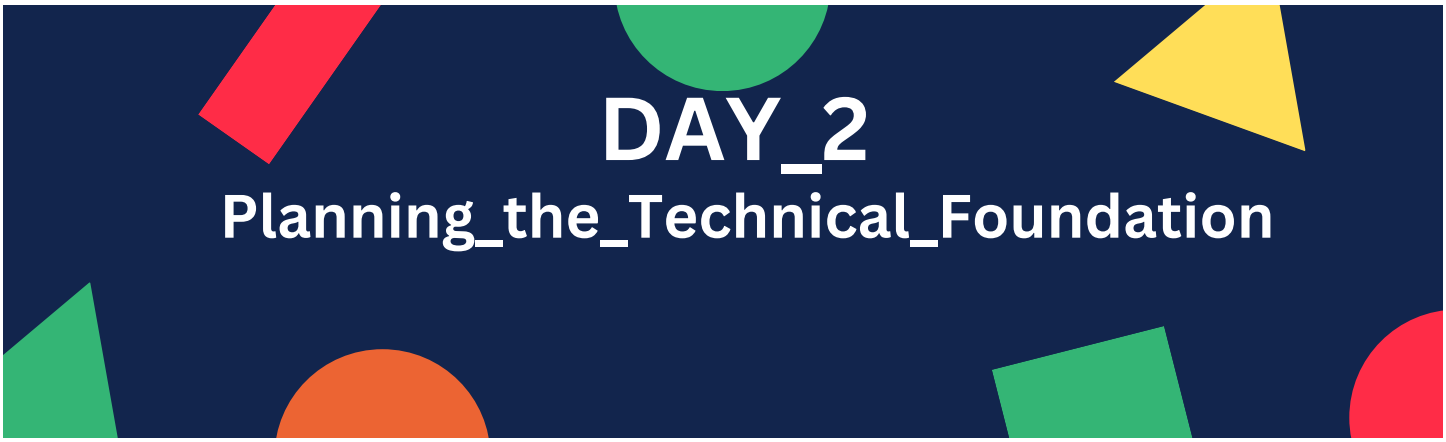


E-COMMERCE MARKET PLACE

Steps involved in building technical market place:

- Frontend: Next.js and Tailwind CSS for styling.
- CMS: Sanity (for dynamic content management).
- Order Tracking: ShipEngine (for real-time shipment updates).
- Payment Gateway: Stripe (for secure payment processing).

System overview:

- Frontend (Next.js):
- Client-side rendering to enhance speed and responsiveness.

○ Server-side rendering for SEO and product page preloading.

○ Tailwind CSS for styling.

Flowchart Overview:

Homepage: Displays product categories, featured items, and search bar.

Category Navigation: Allows users to explore products by categories.

Product Listing: Shows products with filtering and sorting options.

Search Bar: Enables users to search for specific products or sellers.

Product Details Page: Displays detailed product information, images, reviews, and add-to-cart options.

User Account: Manages profile, orders, wishlist, and saved addresses.

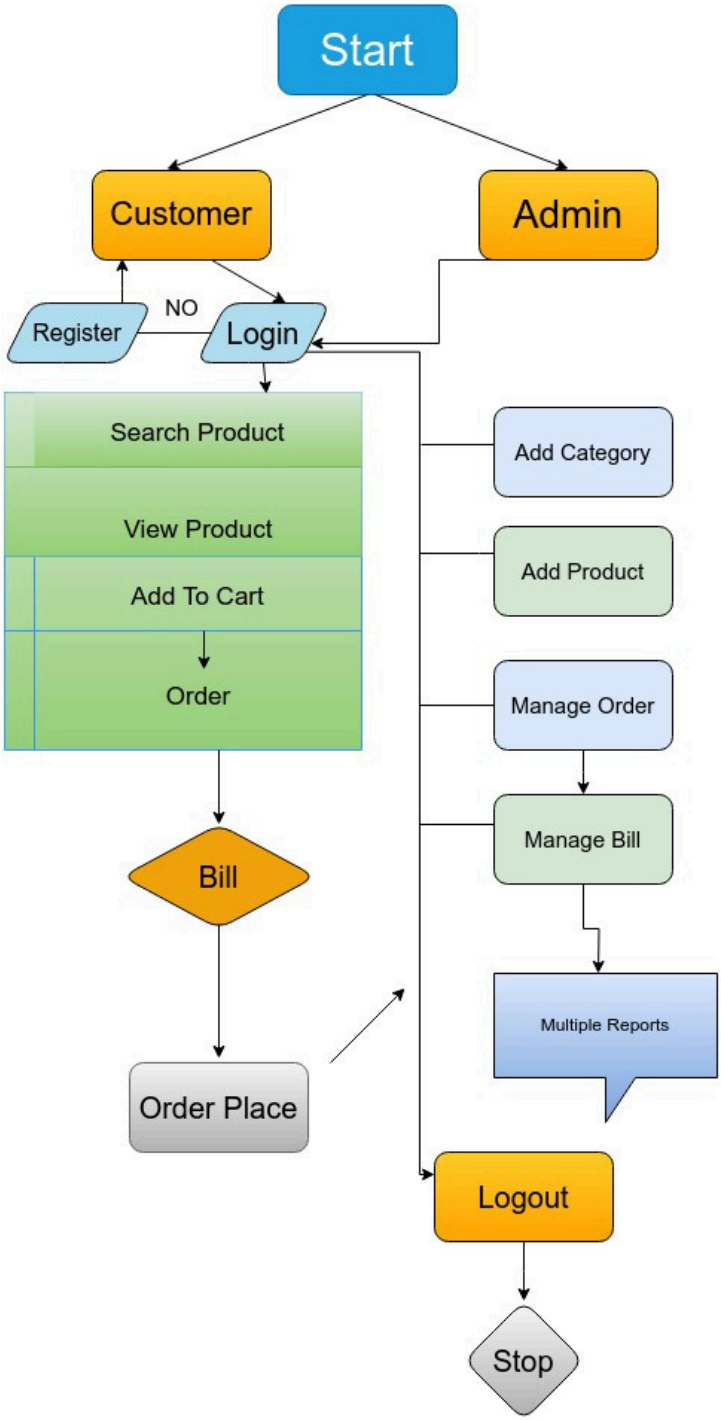
Shopping Cart: Summarizes selected items with options to edit and apply discounts.

Checkout Page: Captures billing, shipping, and payment information.

Order Confirmation: Confirms the purchase and provides order tracking.

Delivery: Product deliver to the address of the customer.

Flow Chart Diagram:



API Endpoints:

Authentication

1. POST /api/auth/register: User registration.
2. POST /api/auth/login: User login.

Product Management

1. GET /api/products: Get all products (with optional filters and pagination).
2. GET /api/products/{id}: Get details of a specific product.

Categories and Filters

1. GET /api/categories: Get all product categories.
2. GET /api/categories/{id}/products: Get products within a specific category.

Cart and Wishlist

1. POST /api/cart: Add an item to the cart.
2. GET /api/cart: View cart items.

Orders

1. POST /api/orders: Create a new order.
2. GET /api/orders: Get all orders for the user.

Here's a list of common API endpoints for a marketplace e-commerce website with a brief description of each:

Authentication

1. **POST /api/auth/register: User registration.**
2. **POST /api/auth/login: User login.**
3. **POST /api/auth/logout: User logout.**
4. **GET /api/auth/profile: Fetch user profile details.**
5. **PUT /api/auth/update: Update user profile or password.**

Product Management

1. **GET /api/products: Get all products (with optional filters and pagination).**
2. **GET /api/products/{id}: Get details of a specific product.**
3. **POST /api/products: Create a new product (seller only).**
4. **PUT /api/products/{id}: Update a product (seller only).**
5. **DELETE /api/products/{id}: Delete a product (seller only).**

Categories and Filters

1. **GET /api/categories: Get all product categories.**
2. **GET /api/categories/{id}/products: Get products within a specific category.**
3. **GET /api/filters: Fetch available filters (e.g., price, brand, rating).**

Cart and Wishlist

1. **POST /api/cart: Add an item to the cart.**
2. **GET /api/cart: View cart items.**
3. **PUT /api/cart/{id}: Update quantity of a cart item.**
4. **DELETE /api/cart/{id}: Remove an item from the cart.**
5. **POST /api/wishlist: Add an item to the wishlist.**
6. **GET /api/wishlist: Get wishlist items.**
7. **DELETE /api/wishlist/{id}: Remove an item from the wishlist.**

Orders

1. **POST /api/orders: Create a new order.**
2. **GET /api/orders: Get all orders for the user.**

Payment

1. **POST /api/payment: Process payment for an order.**
2. **GET /api/payment/status/{id}: Fetch payment status.**

Reviews and Ratings

1. POST /api/reviews: Add a review for a product.
2. GET /api/reviews/{product_id}: Get reviews for a specific product.

Admin/Seller Management

1. GET /api/admin/users: View all users (admin only).
2. GET /api/seller/orders: View all orders placed for a seller's products.

Schema:

Product Schema:

```
import { defineType } from "sanity"

export const product = defineType({
  name: "product",
  title: "Product",
  type: "document",
  fields: [
    {
```

```
        name: "title",
        title: "Title",
        validation: (rule) => rule.required(),
        type: "string"
    },
    {
        name:"description",
        type:"text",
        validation: (rule) => rule.required(),
        title:"Description",
    },
    {
        name: "productImage",
        type: "image",
        validation: (rule) => rule.required(),
        title: "Product Image"
    },
    {
        name: "price",
        type: "number",
        validation: (rule) => rule.required(),
        title: "Price",
    },
    {
        name: "tags",
        type: "array",
        title: "Tags",
```

```
        of: [{ type: "string" }]
    },
    {
        name:"dicountPercentage",
        type:"number",
        title:"Discount Percentage",
    },
    {
        name:"isNew",
        type:"boolean",
        title:"New Badge",
    }
  ]
})
Order Schema
 
export const Order = {
name: 'order',
  title: 'Customer Order',   type: 'document',   fields: [
    { name: 'orderId', title: 'Order ID', type:
'string', validation: (Rule) => Rule.required() },
    { name: 'customer', type: 'reference', to: [{ type:
'customer' }], validation: (Rule) => Rule.required() },
    { name: 'items', type: 'array', of: [{ type:
'object', fields: [
      { name: 'product', type: 'reference', to: [{ type: 'product' }] },
      { name: 'quantity', type: 'number', validation:
```

```
(Rule) => Rule.min(1).required() }

  ]}] },

  { name: 'totalAmount', title: 'Total Amount', type:

'number', validation: (Rule) => Rule.min(0).required() },

  { name: 'shippingAddress', type: 'object', fields:

[

{ name: 'street', type: 'string', validation:

(Rule) => Rule.required() },

   { name: 'city', type: 'string', validation:

(Rule) => Rule.required() },

   { name: 'zipCode', type: 'string', validation:

(Rule) => Rule.required() }

  ] },

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

'datetime', validation: (Rule) => Rule.required() },

  { name: 'status', title: 'Status', type: 'string', options: { list: ['Pending', 'Shipped', 'Delivered'], layout:
'dropdown' }, validation: (Rule) => Rule.required() },

 ],

};

Customer Schema  export const Customer = {   name: 'customer',   title: 'Customer', type: 'document',

 fields: [

  { name: 'id', type: 'string', title: 'Customer ID', validation: (Rule) => Rule.required() },

  { name: 'name', type: 'string', title: 'Full Name', validation: (Rule) => Rule.required() },

  { name: 'email', type: 'string', title: 'Email

Address', validation: (Rule) => Rule.required().email() },
```

```
    { name: 'phone', type: 'string', title: 'Phone
Number', validation: (Rule) =>
Rule.regex(/^\+?[0-9]{10,15}$/).required() },
    { name: 'address', type: 'object', fields: [
      { name: 'street', type: 'string', validation:
(Rule) => Rule.required() },
      { name: 'city', type: 'string', validation:
(Rule) => Rule.required() },
      { name: 'zipCode', type: 'string', validation:
(Rule) => Rule.required() }
    ] },
{ name: 'orders', type: 'array', of: [{ type:
'reference', to: [{ type: 'order' }] }] },
  ],
};
```

```
Shipment Schema  export const Shipment = {   name: 'shipment',   title: 'Shipment',   type: 'document',
fields: [
    { name: 'shipmentId', title: 'Shipment ID', type:
'string', validation: (Rule) => Rule.required() },
    { name: 'orderId', type: 'reference', to: [{ type:
'order' }], validation: (Rule) => Rule.required() },
    { name: 'shippingCarrier', type: 'string', title: 'Shipping Carrier', validation: (Rule) =>
Rule.required() },
    { name: 'trackingNumber', type: 'string', title:
'Tracking Number' },
```

{ name: 'shipmentDate', type: 'datetime', title: 'Shipment Date', validation: (Rule) =>

Rule.required() },

{ name: 'shipmentStatus', type: 'string', options:

{ list: ['Shipped', 'In Transit', 'Delivered'], layout:

'dropdown' }, validation: (Rule) => Rule.required() },

],

};

Conclusion:

This plan ensures a scalable and robust e-commerce platform using modern technologies like Next.js, Tailwind CSS, Sanity CMS, ShipEngine, and Stripe for payment processing. It allows small businesses to manage products and orders efficiently while providing a seamless user experience.

COMPLETE VIEW OF MARKETPLACE ARCHITECTURE: