Marketplace Technical Foundation

Overview of Day 2: Marketplace Builder Hackathon e-commerce website

This is Day 2 of the Marketplace Builder Hackathon, where I detail my strategy for building a real e-commerce website that I plan to launch in the market. In this document, I explain the technical foundation of my project. I will use **Next.js** for developing a dynamic and high-performing frontend and **Tailwind CSS** for creating a modern, responsive design. To manage and organize content efficiently, I will integrate **Sanity CMS**, and for extended functionality, I'll leverage **third-party APIs**. This combination ensures scalability, smooth user experiences, and robust backend functionality for a market-ready e-commerce platform.

Technical Requirements

Frontend Requirements

1. **User-Friendly and Easy to Use**: The interface should be intuitive and straightforward to ensure users can navigate and interact with it effortlessly.
2. **Responsive Design**: The website must adapt seamlessly to all screen sizes, including desktops, tablets, and smartphones. Elements should adjust or hide based on the screen type for an optimal viewing experience.
3. **Essential Pages**:
    - **Home Page**: An engaging and informative landing page.
    - **Product Listing Page**: Displays all available products with filters and sorting options.
    - **Product Detail Page**: Shows detailed information about individual products.
    - **Cart Page**: Allows users to view and manage selected items.
    - **Checkout Page**: Facilitates secure and simple payment and delivery processes.
    - **Order Confirmation Page**: Confirms order placement and provides a summary.

Backend Requirements: Sanity CMS

1. **Product Management**:
    - Store product details, including:

        - Name , Price ,Description ,Images ,Categories
    - Ensure easy retrieval and filtering of product information.
2. **Customer Management**:
    - Save customer details such as:
        - Name , Email ,Address ,Contact Information
3. **Order Management**:
    - Record user orders, including:
        - Products purchased , Order status (pending, shipped, delivered) ,Order date and time ,Total price
    - Link orders with user details to maintain a clear customer history.
4. **Schema Design in Sanity**:
    - Define schemas for products, customers, and orders to structure the data.
    - Use Sanity's schema easily fetch data for frontend needs with help of **GET Requests**.
5. **Data Security and Compliance**:
    - Ensure all customer and order data is stored securely.
    - Adhere to data protection standards to maintain privacy and security.

With Sanity CMS, the backend will be efficient, scalable, and well-organized to handle all content management needs while ensuring smooth communication with the frontend.

Third-Party APIs

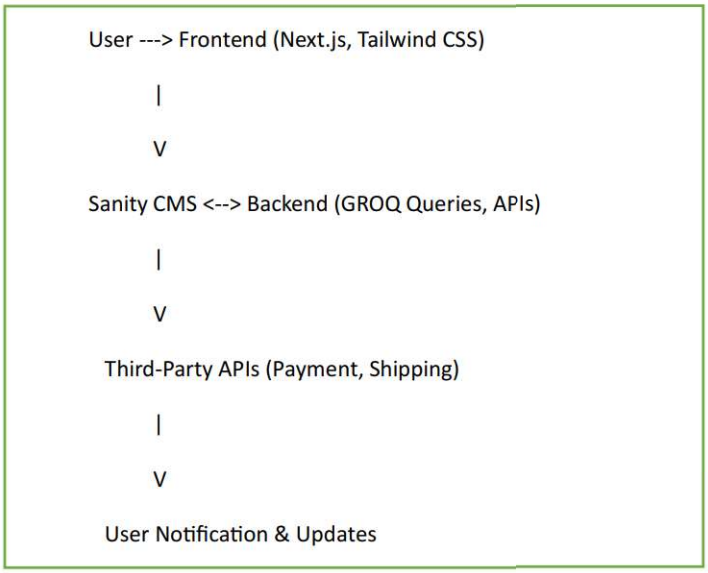
Third-party APIs are essential for adding advanced functionality and handling background services in the e-commerce website. Key APIs include:

1. **Shipment API**: For managing order shipping and tracking, providing real-time updates to customers.
2. **Payment Gateway API**: For secure payment processing, supporting multiple payment methods while ensuring compliance with security standards.
3. **Additional APIs**: For services like taxes, currency conversion, or recommendations, enhancing overall efficiency.

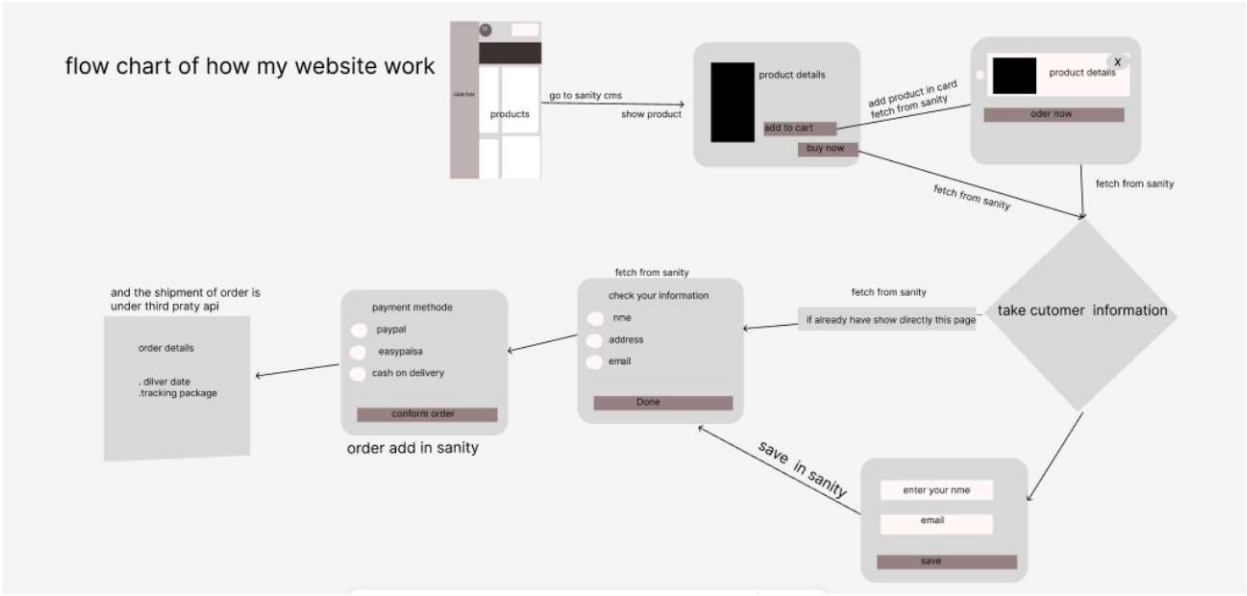
These APIs streamline critical operations, making the website more functional, reliable, and ready for the market.

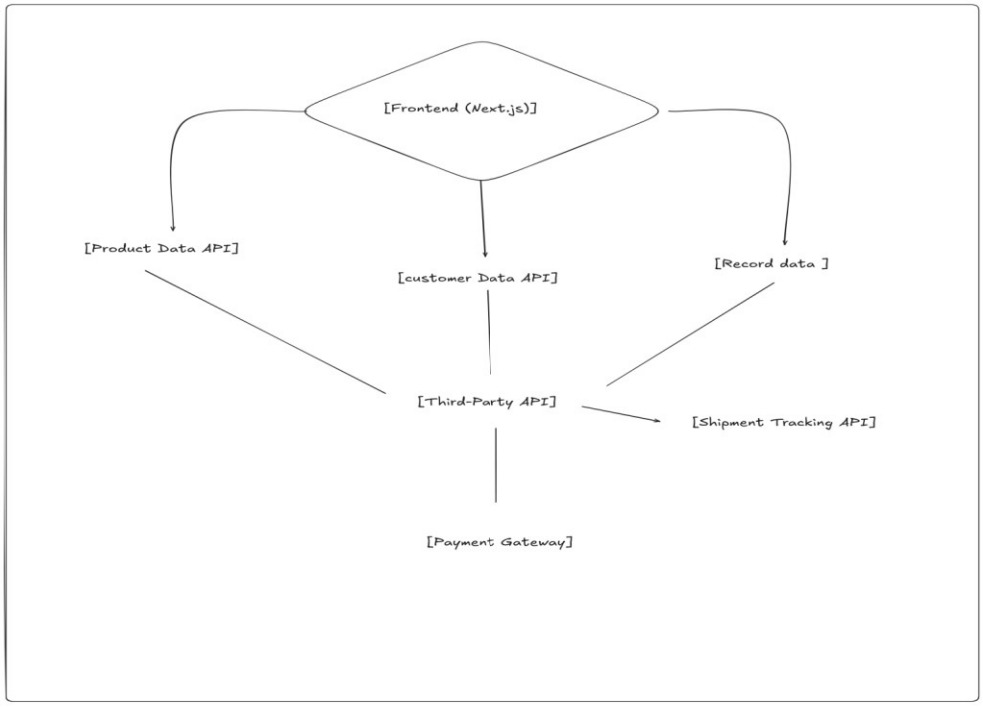
E-Commerce Website Workflow

1. **User Interaction (Frontend):**
   - Users browse the homepage, view product listings, add items to the cart or wishlist, and proceed to checkout with shipping and payment details.
2. **Backend Processing:**
   - **Sanity CMS**: Manages product, customer, and order data with GROQ queries for fetching and updating.
   - **Order Handling**: Validates and saves orders, initializing their status (e.g., pending, processing).
   - **Third-Party APIs**: Integrates payment gateways for secure transactions and shipment APIs for tracking.
3. **Database Interaction:**
   - Stores and updates product, user, and order information in structured schemas within Sanity CMS.
4. **Notifications & Updates:**
   - Sends order confirmation and shipment details to users via email/notifications.
   - Tracks updates and reflects them on the frontend.
5. **Frontend Updates:**
   - Displays order confirmation, tracking info, and dynamically updates cart totals, stock, and recommendations.
   - ailability, and recommendations.



Design System Architecture





Plan API Requirements

GeneraleCommerce

1. products

- **Endpoint Name**: /product
  - **Method**: GET
  - **Description**: Fetch all product
  - **Response Example**:

```
[ { "product_id": 1, "product_name": "T-Shirt", "price": 20.99, "imageUrl": "https://example.com/images/tshirt.jpg", "slug": "t-shirt", "categories": ["Apparel", "Men"], "description": "A comfortable cotton t-shirt.", "size": ["S", "M", "L", "XL"], "variety": ["Red", "Blue", "Black"], "stock": 50, "detail": "100% Cotton. Machine washable.", "discount": 10 }]
```

2. Orders

- **Endpoint Name**: /orders
  - **Method**: GET
  - **Description**: Fetch all orders.
  - **Response Example**:

```
[
  {
    "order_id": 1, "product_id": 101,  "customer_id": 12345,  "quantity": 2, "status": "Pending "order_time_date": "2025-01-17T10:00:00Z",
    "payment_method": "Credit Card"
  },
]
```

3. Shipments

- **Endpoint Name**: /shipments
  - **Method**: GET
  - **Description**: Fetch all shipments.
  - **Response Example**:

```
{
  "shipment_id": 123,  "order_id": 456, "status": "Out for Delivery",  "cost": 15.99, "delivery_date": "2025-01-20T10:00:00Z"
}
```

4. Delivery Boy

- **Endpoint Name**: /delivery-boys
  - **Method**: GET
  - **Description**: Fetch all delivery boys.
  - **Response Example**:

```
[ { "deliveryboy_id": 1, "username": "John Doe", "profile_image": "https://example.com/images/johndoe.jpg", "coverage_area": ["North Zone", "South Zone"], "rating": 4.8, "vehicle_details": { "type": "Bike", "plate_number": "XYZ-1234" },]
```

5. Delivery Zone

- **Endpoint Name**: /delivery-zones
  - **Method**: GET
  - **Description**: Fetch all delivery zones.
  - **Response Example**

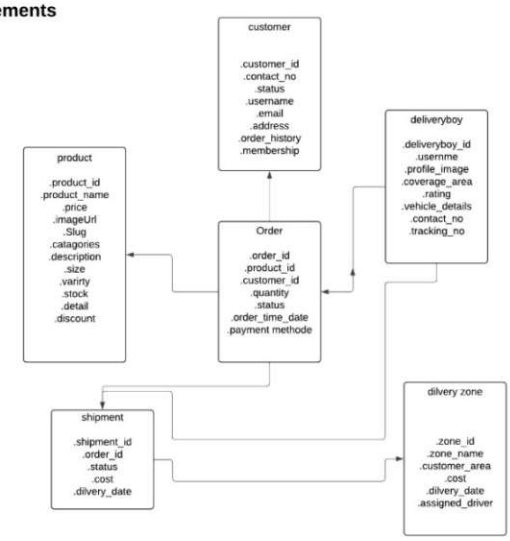
```
{"zone_name": "East Zone","customer_area": ["Area 5", "Area 6"],"cost": 20.00,"delivery_date": "2025-01-20T12:00:00Z",
  "assigned_driver": "Sam Wilson"
}
```

6 Customer API

- **Endpoint Name**: /customers
- **Method**: GET
- **Description**: Fetch all customer details.
  - **Response**

```
[ { "customer_id": 1, "contact_no": "1234567890", "status": "Active", "username": "john_doe", "email": "john.doe@example.com", "address": "123 Main Street, Cityville", "order_history": [ { "order_id": 101, "total": 50.00, "date": "2025-01-10" }, { "order_id": 102, "total": 30.00, "date": "2025-01-15" } ], "membership": "Gold" },]
```

3. Plan API Requirements



Sanity Schema

- **product**

```
{
name: 'product',
fields: [
  { name: 'product_id', type: 'number' },
  { name: 'product_name', type: 'string' },
```

```
  { name: 'price', type: 'number' },
  { name: 'imageUrl', type: 'image' },
  { name: 'slug', type: 'string' },
  { name: 'categories', type: 'array', of: [{ type: 'string' }] },
  { name: 'description', type: 'string' },
  { name: 'size', type: 'array', of: [{ type: 'string' }] },
  { name: 'variety', type: 'array', of: [{ type: 'string' }] },
  { name: 'stock', type: 'number' },
  { name: 'detail', type: 'string' },
 { name: 'discount', type: 'number' }
 ]
}
```

- **Order**

```
{
 name: 'order',
 fields: [
  { name: 'order_id', type: 'number' },
  { name: 'product_id', type: 'number' },
  { name: 'customer_id', type: 'number' },
  { name: 'quantity', type: 'number' },
  { name: 'status', type: 'string' },
  { name: 'order_time_date', type: 'datetime' },
  { name: 'payment_method', type: 'string' }
 ]
}
```

- **Shipment**

```
{
 name: 'shipment',
 fields: [
  { name: 'shipment_id', type: 'number' },
  { name: 'order_id', type: 'number' },
  { name: 'status', type: 'string' },
```

```
      { name: 'cost', type: 'number' },
      { name: 'delivery_date', type: 'datetime' }
     ]
    }
```

- **Customer**

```
{
 name: 'customer',
 fields: [
  { name: 'customer_id', type: 'number' },
  { name: 'contact_no', type: 'string' },
  { name: 'status', type: 'string' },
  { name: 'username', type: 'string' },
  { name: 'email', type: 'string' },
  { name: 'address', type: 'string' },
  {
   name: 'order_history',
   type: 'array',
   of: [
    {
     type: 'object',
     fields: [
      { name: 'order_id', type: 'number' },
      { name: 'total', type: 'number' },
      { name: 'date', type: 'datetime' }
     ]
    }
   ]
  },
  { name: 'membership', type: 'string' }
 ]
```

This is the completion of my Day 2 Hackathon work by **Mehwish Sajjad** I will strive to bring all these ideas to life in my e-commerce website