Marketplace Technical Foundation – Online Grocery Store

1. Define Technical Requirements

Frontend Requirements:

- **User-Friendly Interface**:
  - Intuitive navigation for browsing grocery items by categories (e.g., Vegetables, Fruits, Beverages, etc.).
  - Easy-to-use search bar with filtering options (price, popularity, availability, etc.).
  - Feature for users to mark favorite or commonly purchased items.
- **Responsive Design**:
  - Seamless experience on mobile, tablet, and desktop.
  - Adaptive layout for smaller screens, ensuring readability and usability.
- **Essential Pages**:
  - **Home Page**:
    - Highlight featured or discounted grocery items.
    - Display categories for easy access.
  - **Product Listing Page**:
    - Display products with images, names, prices, and stock availability.
    - Pagination or infinite scroll for large inventories.
  - **Product Details Page**:
    - Detailed view with product description, nutritional information, and related products.
    - Add-to-cart button with quantity selector.
  - **Cart Page**:
    - List of selected products with editable quantities.
    - Summary of the total cost including applicable taxes or delivery charges.
  - **Checkout Page**:
    - Input fields for user details (name, address, contact number).
    - Selection of delivery options and payment method.
  - **Order Confirmation Page**:
    - Confirmation details with order number and estimated delivery date.
    - Option to track order status.

Backend: Sanity CMS

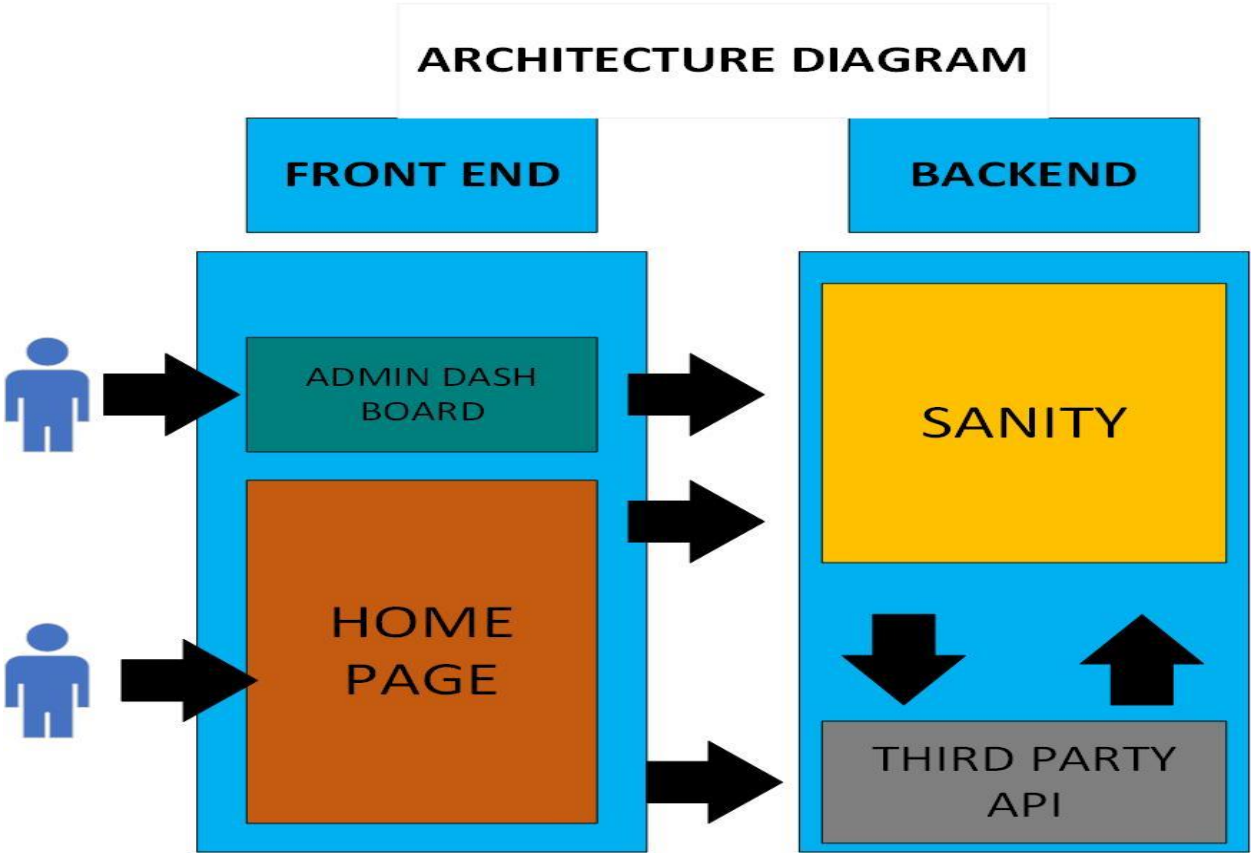
- **Database Management**:
  - Use Sanity CMS to manage and store data such as:
    - Product information: Name, description, price, stock, category, and images.
    - Customer details: Name, address, order history.
    - Order records: Order ID, items purchased, payment status, and delivery status.
  - Design schemas for:
    - **Products**:
      - Fields: Name, SKU, category, price, stock, discount, nutritional info, and image URLs.

        - **Customers**:
            - Fields: Name, email, phone, addresses, and order history.
        - **Orders**:
            - Fields: Order ID, customer reference, product list, total price, payment status, and delivery status.
- **Content Delivery**:
    - Use GROQ queries to fetch product and order data efficiently for the frontend.
    - Implement real-time updates for inventory changes or order status.

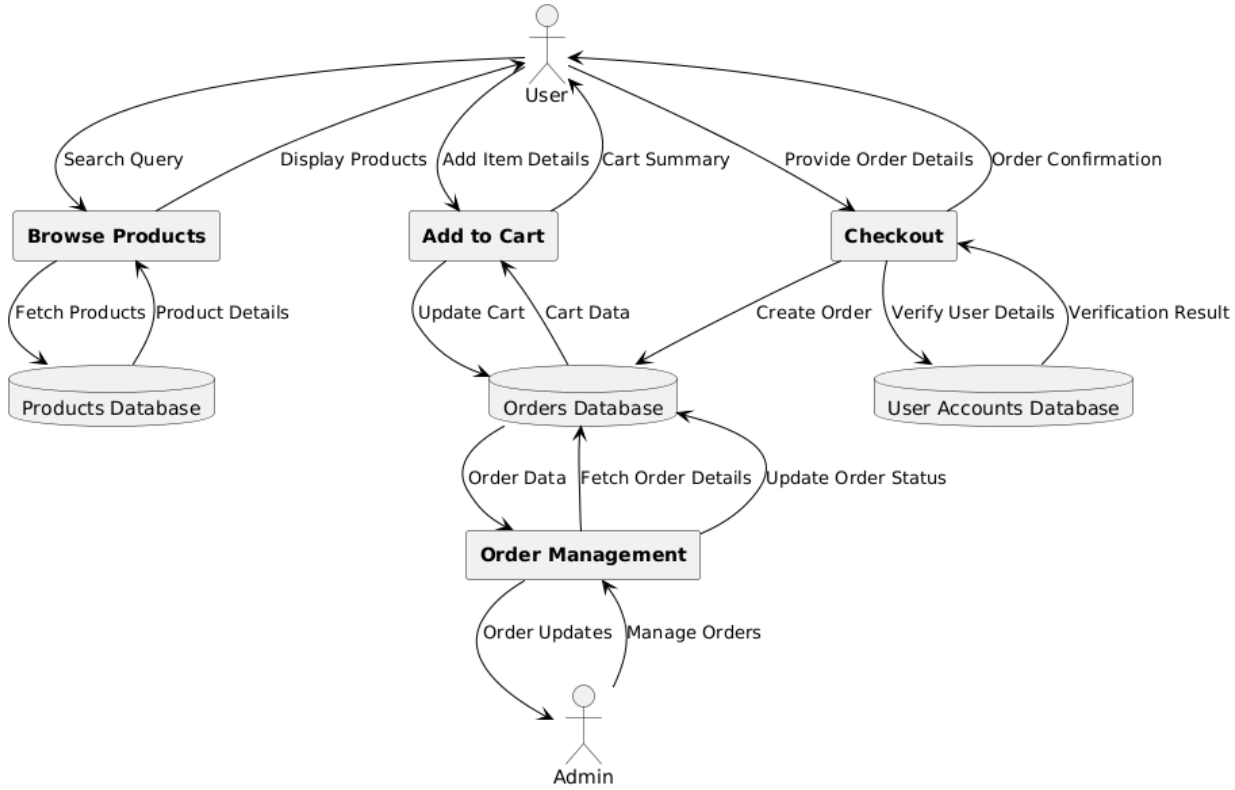
Third-Party APIs

- **Shipment Tracking**:
    - Integrate shipment APIs (e.g., FedEx, DHL, or local delivery services) to provide real-time order tracking.
    - Display tracking status on the Order Confirmation page.
- **Payment Gateway Integration**:
    - Integrate secure payment APIs (e.g., Easy paisa, Jazz Cash etc.) to handle transactions.
    - Support multiple payment methods: Credit/Debit Card, and Easypaisa.

System Architecture Overview



Key Workflows:



Detailed Component Roles:

1. Frontend (Next.js):

- Serves as the user interface for the grocery store.
- Handles user interactions (browsing, searching, ordering).
- Fetches dynamic product and order data from APIs.

2. Sanity CMS:

- Acts as the backend and database for managing:
  - Products, categories, and inventory.
  - Customer profiles and orders.
- Provides APIs to fetch, update, and manage data.

3. Third-Party APIs:

- **Shipment Tracking API**:
  - Retrieves real-time delivery updates for customer orders.
- **Payment Gateway API**:
  - Processes secure payments through credit/debit cards, or Easy paisa.

1. General Endpoints:

1. **Endpoint Name**: `/products`
    - **Method**: GET
    - **Description**: Fetch all available products from Sanity CMS.
    - **Response Example**:

```
[
  { "id": 1, "name": "Apple", "price": 2.5, "stock": 50, "image":
"apple.jpg" },
  { "id": 2, "name": "Banana", "price": 1.2, "stock": 30,
"image": "banana.jpg" }
]
```

2. **Endpoint Name**: `/product/id`
    - **Method**: GET
    - **Description**: Fetch detailed information for a specific product.
    - **Response Example**:

```
{
  "id": 1,
  "name": "Apple",
  "price": 2.5,
  "stock": 50,
  "description": "Fresh red apples from organic farms.",
  "image": "apple.jpg"
}
```

```
export default {
  name: "product",
  title: "Product",
  type: "document",
  fields: [
    {
      name: "id",
      title: "ID",
      type: "number",
    },
    {
      name: "name",
      title: "Name",
      type: "string",
    },
    {
      name: "price",
      title: "Price",
      type: "number",
    },
    {
```

```
      name: "stock",
      title: "Stock",
      type: "number",
    },
    {
      name: "description",
      title: "Description",
      type: "text",
    },
    {
      name: "image",
      title: "Image",
      type: "image",
      options: {
        hotspot: true,
      },
    },
  ],
};
```

2. Order Management:

1. **Endpoint Name**: `/orders`
    - **Method**: POST
    - **Description**: Create a new order in Sanity CMS.
    - **Payload Example**:

```
{
  "customer": {
    "name": "John Doe",
    "email": "john@example.com",
    "address": "123 Main St, City, Country"
  },
  "orderDetails": [
    { "productId": 1, "quantity": 3 },
    { "productId": 2, "quantity": 1 }
  ],
  "paymentStatus": "Paid"
}
```

    - **Response Example**:

```
{ "orderId": 12345, "status": "Order Placed", "total": 9.7 }
```

2. **Endpoint Name**: `/orders/Id`
    - **Method**: GET

- **Description**: Fetch order details for a specific order.
- **Response Example**:

```
{
  "orderId": 12345,
  "customer": {
    "name": "John Doe",
    "email": "john@example.com",
    "address": "123 Main St, City, Country"
  },
  "orderDetails": [
    { "productId": 1, "name": "Apple", "quantity": 3, "price":
2.5 },
    { "productId": 2, "name": "Banana", "quantity": 1, "price":
1.2 }
  ],
  "paymentStatus": "Paid",
  "total": 9.7,
  "status": "Processing"
}
```

```
export default {
  name: "order",
  title: "Order",
  type: "document",
  fields: [
    {
      name: "orderId",
      title: "Order ID",
      type: "number",
    },
    {
      name: "customer",
      title: "Customer",
      type: "object",
      fields: [
        { name: "name", title: "Name", type: "string" },
        { name: "email", title: "Email", type: "string" },
        { name: "address", title: "Address", type: "string" },
      ],
    },
    {
      name: "orderDetails",
      title: "Order Details",
      type: "array",
      of: [
        {
          type: "object",
          fields: [
            { name: "productId", title: "Product ID", type: "number" },
```

```
            { name: "name", title: "Product Name", type: "string" },
            { name: "quantity", title: "Quantity", type: "number" },
            { name: "price", title: "Price", type: "number" },
          ],
        },
      ],
    },
    {
      name: "paymentStatus",
      title: "Payment Status",
      type: "string",
    },
    {
      name: "total",
      title: "Total",
      type: "number",
    },
    {
      name: "status",
      title: "Status",
      type: "string",
    },
  ],
};
```

3. Shipment Tracking:

1. **Endpoint Name**: `/shipment`
    - **Method**: GET
    - **Description**: Track order status via a third-party API.
    - **Response Example**:

```
{
  "shipmentId": "SHIP123",
  "orderId": 12345,
  "status": "In Transit",
  "expectedDelivery": "2025-01-20T15:00:00Z"
}
```

2. **Endpoint Name**: `/shipment/Id`
    - **Method**: GET
    - **Description**: Fetch detailed shipment tracking information.
    - **Response Example**:

```
{
```

```
  "shipmentId": "SHIP123",
  "orderId": 12345,
  "status": "Out for Delivery",
  "location": "City Warehouse",
  "expectedDelivery": "2025-01-20T15:00:00Z"
}
```


SanitySchematable

Endpoints	Method	Purpose	Response Example
/products	GET	Fetch all available products	[{ "id": 1, "name": "Apple", "price": 2.5, "stock": 50, "image": "apple.jpg" }, { "id": 2, "name": "Banana", "price": 1.2, "stock": 30, "image": "banana.jpg" }]
/product/id	GET	Fetch detailed information specific product.	{"id": 1,"name": "Apple", "price": 2.5, "stock": 50, "description": "Fresh red apples from organic farms.","image": "apple.jpg"}
/orders	POST	Create a new order	{"customer": {"name": "John Doe","email": "john@example.com", "address": "123 Main St,City, Country"}, "orderDetails": [{ "productId": 1, "quantity": 3 },{ "productId": 2, "quantity": 1 }], "paymentStatus": "Paid"}
/orders/Id	GET	Fetch order details for a specific order.	{"orderId": 12345,"customer": {"name": "John Doe","email": "john@example.com","address": "123 Main St, City, Country"}, "orderDetails": [{ "productId": 1, "name": "Apple","quantity": 3, "price": 2.5 },{ "productId": 2, "name": "Banana", "quantity": 1, "price": 1.2 }], "paymentStatus": "Paid","total": 9.7,"status": "Processing"}
/shipment	GET	Track order status	{ "shipmentId": "SHIP123", "orderId": 12345, "status": "In Transit", "expectedDelivery": "2025-01-20T15:00:00Z" }
/shipment/Id	GET	Fetch detailed shipment tracking information.	{"shipmentId": "SHIP123","orderId": 12345,"status": "Out for Delivery", "location": "City Warehouse","expectedDelivery":"2025-01-20T15:00:00Z"}