Building Blocks of Our Platform: System Architecture and Component Interactions

The system comprises three core components:

Frontend:
Using **Next.js 14** for a responsive UI, with separate pages for male, female, and kids, and routes for products, cart, and checkout.

Backend:
Leveraging **Sanity** as a headless CMS for data management.

Third-Party APIs:
Integrating APIs for shipment tracking and payment gateways for secure transactions.

Interactions:

Users browse products through the UI.

The **frontend** communicates with APIs to retrieve data.

while the **backend** integrates with the headless CMS, **Sanity**, to manage and deliver content.

User Journey: Browsing, Ordering, and Payment Process:

User opens the platform and lands on the frontend where the products are displayed.

The frontend communicates with the API to fetch product data.

The data is then migrated into Sanity, the headless CMS.

Using GROQ, the product data is integrated into the frontend UI.

The user selects a product and places an order.

The order status is updated, and the shipment and payment processes begin.

User Journey:

Browsing, Ordering, and Payment Process

- **User → Frontend → Product:**

  The user opens the platform and interacts with the frontend, where they can browse and view products.

- **User → Product → Order:**

  After selecting a product, the user places an order through the frontend.

- **Order → Order Status Update → Shipment Tracking:**

  Once the order is placed, the order status is updated, and shipment tracking begins to keep the user informed.

Relationship:

- **One User → Many Orders**:

  A single user can place multiple orders on the platform.

- **One Order → Many Products**:

  Each order can contain multiple products, allowing users to purchase more than one item at a time.

API Specification Document

Endpoint: `/products`

- **Method**: `GET`
- **Description**: Fetch all available products from Sanity.

Response Example:

```
[
  {
    "id": 1,
    "name": "Product A",
    "price": 100,
    "stock": 50,
    "category": "Male",
    "size": "XL",
    "image": "https://example.com/images/productA.jpg"
  }
]
```

Endpoint: `/orders`

Method: `POST`

Description: Create a new order in Sanity.

Request Payload:

```
{
  "customer": {
    "order_Id": 244,
    "name": "John Doe",
    "email": "john@example.com",
    "address": "123 Main St",
    "timestamp": "4 days"
  },
  "products_Detail": [
    { "productId": 1, "quantity": 2 },
    { "productId": 2, "quantity": 1 }
  ],
  "paymentStatus": "pending"
}
```

Response Example:

```
{
  "orderId": 1234,
  "status": "pending",
```

```
  "totalPrice": 280,
  "timestamp": "4 days"
}
```

Endpoint: `/shipment`

- **Method**: `GET`
- **Description**: Track order status via third-party API.
- **Request Parameters**: `orderId`

Response Example:

```
{
  "shipmentId": "ABC123",
  "orderId": 1234,
  "status": "Shipped",
  "expectedDeliveryDate": "2025-01-20"
}
```

Endpoint: `/payment`

- **Method**: `POST`
- **Description**: Process payment for an order.

Request Payload:

```
{
  "orderId": 1234,
  "amount": 280,
  "paymentMethod": "Credit Card",
  "cardDetails": {
    "cardNumber": "4111111111111111",
    "expiryDate": "12/26",
    "cvv": "123"
  }
}
```

Response Example:

```
{
  "paymentId": "PAY12345",
  "status": "Success",
  "transactionDate": "2025-01-16",
  "amount": 280
```

```
}
```

Endpoint: `/customization`

- **Method**: `POST`
- **Description**: Add customization details to an order.

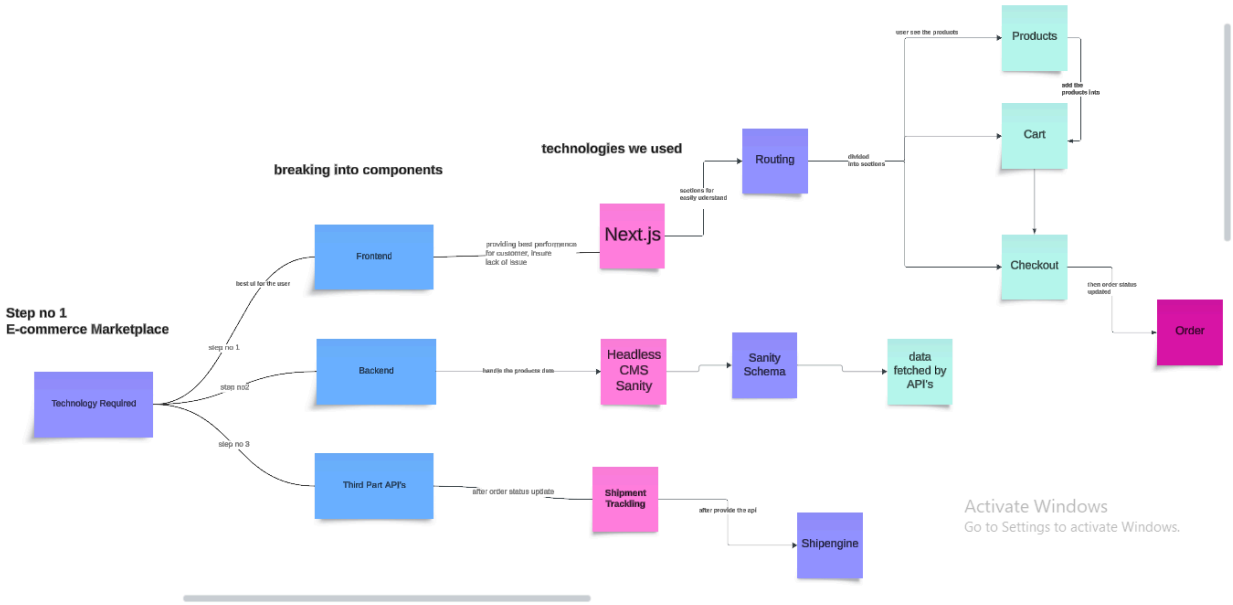
Request Payload:

```
{
  "orderId": 1234,
  "customization": {
    "engraving": "Happy Birthday",
    "color": "Black",
    "specialPackaging": true
  }
}
```

Response Example:

```
{
  "orderId": 1234,
  "status": "Customization Added",
  "customization": {
    "engraving": "Happy Birthday",
    "color": "Black",
    "specialPackaging": true
  }}
```

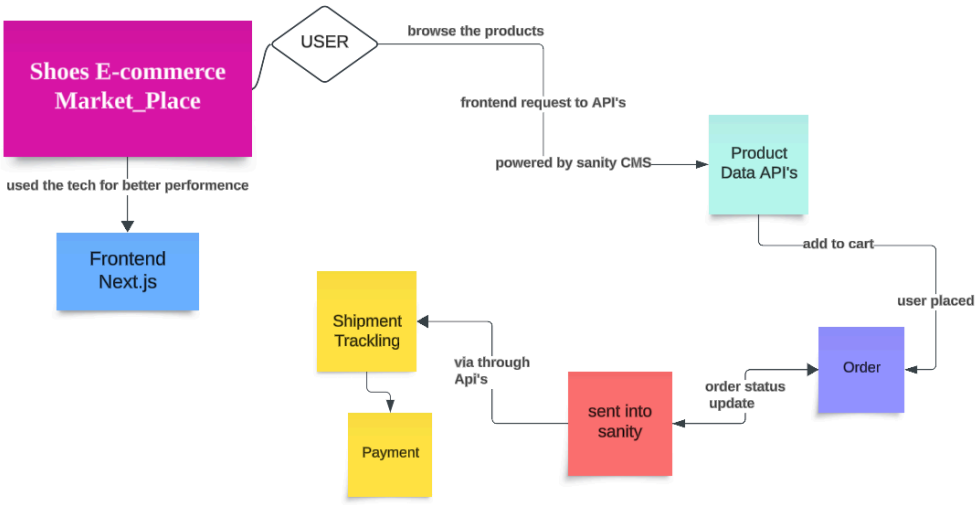
Workflow Diagram: Visualizes user interactions and data flows.



Data Schema Design:

Entities

- **Product,**
- **Order,**
- **Shipment ,**
- **Payment**



Technical Roadmap

1. Initiation
    - Define objectives, finalize requirements, and choose technology stack.
2. Frontend Development
    - Set up Next.js and responsive UI for products, cart, and checkout.
    - Integrate product data using APIs.
3. Backend Development
    - Configure Sanity CMS for products and orders.
    - Integrate payment and shipment tracking APIs.
4. API Development
    - Create endpoints for products, orders, payments, and shipments.
    - Add error handling and security measures.
5. Customization Features

    - Enable product customization (e.g., engraving, packaging).
6. Testing
    - Conduct unit and end-to-end testing for system reliability.
7. Deployment
    - Host the platform on a production server and go live.
8. Maintenance
    - Monitor performance and address updates based on user feedback.

Deliverables: Fully functional eCommerce platform with dynamic product management, order processing, shipment tracking, and customization options.

Quick Summary

- Core Components:
    - Frontend: Next.js 14 for responsive UI.
    - Backend: Sanity CMS for product and order management.
    - Third-Party APIs: Shipment tracking and payment processing.
- User Journey:
    - Browse products → Place order → Track shipment and payment.
- Key APIs:
    - `/products`, `/orders`, `/shipment`, `/payment`, `/customization`.
- Roadmap:
    - Define requirements → Build frontend/backend → Integrate APIs → Test and deploy.
- Deliverables: Fully functional eCommerce platform with customization and order tracking.

  Great platforms are built on simple foundations but deliver exceptional experiences.