System Architecture for Next.js Project

By. Saima Bukhsh GIAIC Q2

This system architecture illustrates how the components in your Next.js project interact with each other and external services.

High-Level Overview

```
[Frontend (Next.js)]
    |        |         |
[Sanity CMS] [Third-Party APIs] [Other Services]
```

Frontend (Next.js)

The primary interface for users:

Features:

- **User account management**: Login, registration, and password recovery.
- **E-commerce**: Product browsing, detailed views, cart operations, and checkout.
- **Blog Section**: Articles and related content.
- **Contact Form**: For user inquiries.

Tools/Technologies:

- React components, Next.js pages, and TailwindCSS for styling.

Backend: Sanity CMS

The backend for managing and retrieving content:

Features:

- Stores product details, user data, and blog content.
- Provides APIs for fetching and manipulating data.

Schema Example:

```
export default {
  name: 'product',
  type: 'document',
  fields: [
    { name: 'name', type: 'string', title: 'Product Name' },
    { name: 'price', type: 'number', title: 'Price' },
    { name: 'stock', type: 'number', title: 'Stock Level' }
  ]
};
```

Third-Party APIs

Provides external functionalities:

Features:

- **Shipment Tracking**: Track order delivery updates.
- **Payment Processing**: Secure integration for transactions.
- **Analytics**: User behavior tracking.

Sample Endpoint:

```
{
  "endpoint": "/shipment",
  "method": "GET",
  "description": "Track order status via third-party API",
  "response": {
    "orderId": 123,
    "status": "In Transit",
    "expectedDelivery": "2025-01-20"
  }
}
```

Key Workflows

1. User Registration

```
User -> Next.js Form -> Sanity CMS API -> User Data Saved
```

2. Product Browsing

```
User -> Next.js Product Page -> Sanity CMS API -> Products Fetched ->
Displayed on UI
```

3. Order Placement

```
User -> Cart Checkout -> Sanity CMS API -> Order Created
```

4. Shipment Tracking

```
User -> Shipment Status Page -> Third-Party API -> Status Displayed
```

Example Data Flow

1. A user accesses the **Product Listing Page**.
2. Next.js fetches product data from the **Sanity CMS** API.
3. The data is dynamically rendered as responsive UI elements.
4. When a product is added to the cart, details are sent to **Sanity CMS**.
5. During checkout, the payment gateway processes payment.
6. Once the order is confirmed, shipment tracking is provided via a **Third-Party API**.

This architecture ensures modular, scalable, and maintainable development while adhering to best practices for modern web applications.

Detailed Workflow for Next.js Project

1. User Registration

- User signs up via the registration form on the account page.
- User data (username, email, password) is sent to Sanity CMS.
- Sanity CMS stores the data securely.
- A confirmation email is sent to the user.

2. Product Browsing

- User navigates to the shop page.
- Categories and products are dynamically fetched from the Sanity CMS API.
- Data is displayed on the frontend, including product images, prices, and availability.

3. Product Details View

- User selects a product to view its details.
- The product details page fetches specific product data from Sanity CMS.
- Data such as product description, reviews, and stock level is displayed.

4. Cart Management

- User adds a product to the cart from the product details page.
- Cart data is managed on the frontend and synced with Sanity CMS for persistence.
- Users can update quantities or remove items from the cart.

5. Order Placement

- User proceeds to the checkout page.
- Billing and shipping details are collected via a form.
- Order details, including cart items, billing info, and payment status, are sent to Sanity CMS.
- Sanity CMS records the order.

6. Payment Processing

- User selects a payment method and submits the payment.
- Payment gateway processes the transaction securely.
- Confirmation is sent to both the user and Sanity CMS.

7. Shipment Tracking

- After order placement, shipment details are generated.
- Third-party APIs fetch real-time shipment tracking data.
- Users can view the order status and expected delivery time on the shipment tracking page.

8. Contact Form Submission

- User submits inquiries via the contact page.
- Data is sent to Sanity CMS or an email service for further action.
- Acknowledgment is displayed on the frontend.

By organizing the workflows clearly, this ensures efficient implementation and a smooth user experience across the platform.

Planned API Requirements

General API Endpoints

Endpoint Name: `/products`

- **Method**: GET
- **Description**: Fetch all available products from Sanity CMS.
- **Response**:
- `[`
- `  {`
- `    "id": "1",`
- `    "name": "Asgaard Sofa",`

- `    "price": 250000,`
- `    "stock": 5,`
- `    "image": "asgaard-sofa.png"`
- `  },`
- `  {`
- `    "id": "2",`
- `    "name": "Trenton Modular Sofa",`
- `    "price": 150000,`
- `    "stock": 8,`
- `    "image": "trenton-sofa.png"`
- `  }`
- `]`

Endpoint Name: `/orders`

- **Method**: POST
- **Description**: Create a new order in Sanity CMS.
- **Payload**:
- `{`
- `  "customer": {`
- `    "name": "John Doe",`
- `    "email": "johndoe@example.com"`
- `  },`
- `  "products": [`
- `    { "id": "1", "quantity": 2 },`
- `    { "id": "2", "quantity": 1 }`
- `  ],`
- `  "paymentStatus": "Paid"`
- `}`
- **Response**:
- `{`
- `  "orderId": "12345",`
- `  "status": "Success",`
- `  "message": "Order created successfully."`
- `}`

Endpoint Name: `/shipment`

- **Method**: GET
- **Description**: Track order status via third-party API.
- **Response**:
- `{`
- `  "shipmentId": "67890",`
- `  "orderId": "12345",`
- `  "status": "In Transit",`
- `  "expectedDelivery": "2025-01-20"`
- `}`

This API design aligns with the workflows, ensuring clarity and ease of implementation for developers.