"Marketplace Technical Foundation (GENERAL E-COMMERCE) "

1. Define Technical Requirements

The first step is to translate your business goals into clear technical requirements. For each feature identified on Day 1, outline the following:

Frontend and Backend Requirements

Frontend Requirements:

The frontend of the application must be designed with user experience in mind. Key requirements include:

User-friendly interface: The interface should allow users to easily browse and navigate through products.

Responsive design: The design must be adaptable for both mobile and desktop users, ensuring a seamless experience across devices.

Essential pages: The application should include the following critical pages:

- Home
- About
- Contact
- Product Listing
- Product Details
- Cart
- Checkout
- Order Confirmation

Sanity CMS as Backend

Sanity CMS will serve as the backend for managing various data types essential for the marketplace. Key points include:

Data management: Sanity CMS will handle product data, customer details, and order records, functioning as the primary database.

Schema design: It is crucial to focus on designing schemas in Sanity that align with the business goals from the very beginning of the project.

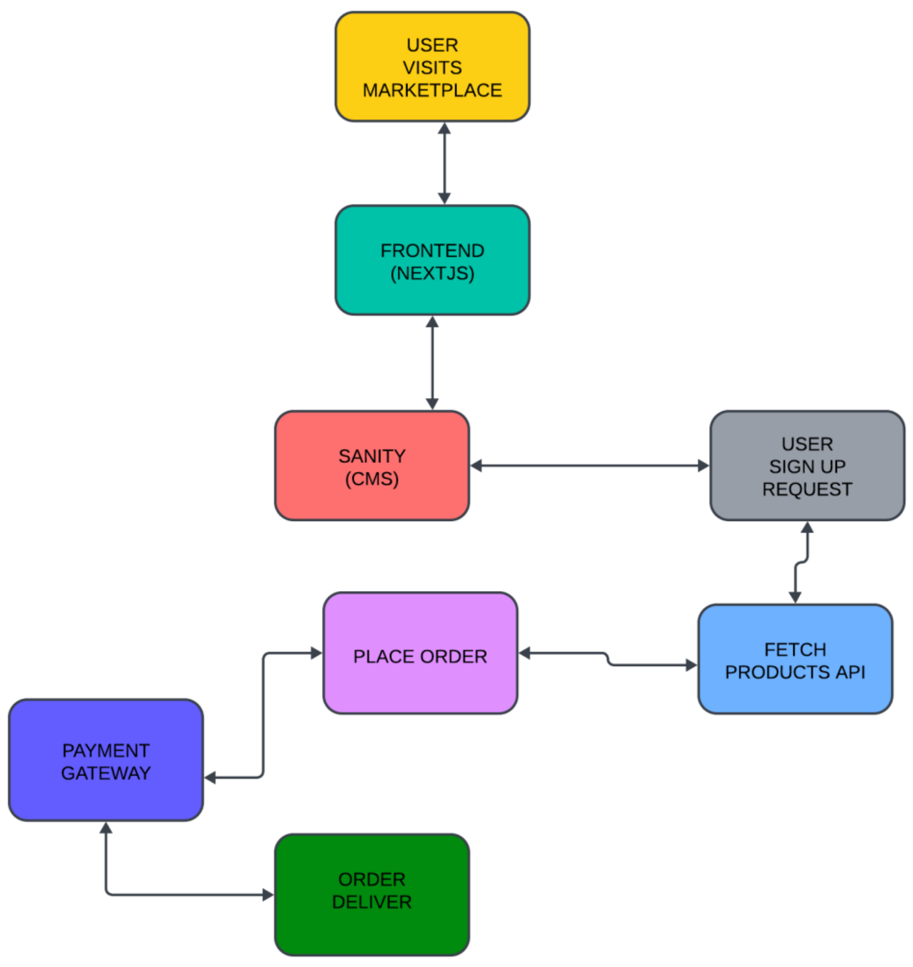
Third-Party APIs

Integrating third-party APIs is essential for enhancing the functionality of the application. Important considerations include:

API integration: Integrate APIs for various services such as shipment tracking and payment gateways to facilitate backend operations.

Data provision: Ensure that the APIs provide all necessary data to support frontend functionality, allowing for a smooth

2. Design System Architecture



Typical Data Flow

The data flow in this architecture can be outlined in several steps:

1. A user visits the marketplace frontend to browse products.
2. The frontend makes a request to the Product Data API (powered by Sanity CMS) to fetch product listings and details, which are displayed dynamically on the site.
3. When the user places an order, the order details are sent to Sanity CMS via an API request, where the order is recorded.
4. Shipment tracking information is fetched through a Third-Party API and displayed to the user in real-time.
5. Payment details are securely processed through the Payment Gateway, and a confirmation is sent back to the user and recorded in Sanity CMS.

Understanding Component Interactions

This detailed workflow illustrates how components interact in a real-world scenario, emphasizing the seamless flow of data between them. It highlights the role of:

- Frontend interactions supported by Sanity CMS for content management.
- Third-party APIs for logistics and shipment tracking.
- Payment gateways for secure transaction processing.

Including these details in your architecture will aid in visualizing dependencies and planning integrations effectively.

Key Workflows to Include

When designing the system architecture, consider incorporating the following key workflows:

1. **User Registration:**
   - User signs up.
   - Data is stored in Sanity.
   - Confirmation sent to the user.
2. **Product Browsing:**
   - User views product categories.

    - Sanity API fetches data.
    - Products displayed on frontend.
3. **Order Placement:**
    - User adds items to the cart.
    - User proceeds to checkout.
    - Order details saved in Sanity.
4. **Shipment Tracking:**
    - Order status updates fetched via 3rd-party API.
    - Information displayed to the user.

3. DATA SCHEMA SANITY

Product Schema

```
export default {
  name: 'product',
  title: 'Product',
  type: 'document',
  fields: [
    { name: 'name', title: 'Product Name',  type: 'string',  },
    { name: 'description', title: 'Description,  type: 'text', },
    { name: 'price', title: 'price,  type: 'number', },
    { name: 'stock', title: 'stock level,  type: 'number', },
    { name: 'image', title: ‘Product image’,  type: 'image', },
    {
      name: 'category',
      title: 'Category',
      type: 'reference',
      to: [{ type: 'category' }],
    },
  ],
};
```

Order Schema

```
export default {
  name: 'order',
  type: 'document',
  title: 'Order',
  fields: [
    {
      name: 'orderId',
      type: 'string',
      title: 'Order ID',
    },
    {
      name: 'customer',
      type: 'reference',
      to: [{ type: 'customer' }],
      title: 'Customer',
    },
    {
      name: 'products',
      type: 'array',
      title: 'Products',
      of: [
        {
          type: 'object',
          fields: [
            {
              name: 'product',
              type: 'reference',
              to: [{ type: 'product' }],
              title: 'Product',
            },
            {
              name: 'quantity',
              type: 'number',
              title: 'Quantity',
            },
          ],
        },
      ],
    },
    {
      name: 'totalAmount',
      type: 'number',
      title: 'Total Amount',
    },
    {
      name: 'status',
      type: 'string',
      title: 'Order Status',
      options: {
        list: [
          { title: 'Pending', value: 'Pending' },
          { title: 'Shipped', value: 'Shipped' },
          { title: 'Delivered', value: 'Delivered' },
        ],
      },
    },
    {
      name: 'createdAt',
      type: 'datetime',
      title: 'Order Date',
    },
  ],
    };
```

Customer Schema

```
export default {
  name: 'customer',
  type: 'document',
  title: 'Customer',
  fields: [
    {
      name: 'name',
      type: 'string',
      title: 'Customer Name',
    },
    {
      name: 'email',
      type: 'string',
      title: 'Email',
    },
    {
      name: 'contactNumber',
      type: 'string',
      title: 'Contact Number',
    },
    {
      name: 'address',
      type: 'text',
      title: 'Address',
    },
  ],
};
```

Category Schema

```
export default {
  name: 'category',
  type: 'document',
  title: 'Category',
  fields: [
    {
      name: 'title',
      type: 'string',
      title: 'Category Title',
    },
    {
      name: 'description',
      type: 'text',
      title: 'Description',
    },
  ],
};
```