Day 2 Plan: Building the Technical Foundation

1. Define Technical Requirements

Frontend Requirements

- **User-Friendly Interface:**
  - Quick product search and filtering.
  - Real-time stock visibility.
  - Delivery progress tracking.
- **Responsive Design:**
  - Optimized for desktop, mobile, and tablet.
- **Essential Pages:**

1. **Home:** Highlights deals and categories.
2. **Product Listing:** Displays products with availability status.
3. **Cart:** Summarizes selected items.
4. **Checkout:** Captures delivery address and payment details.
5. **Order Tracking:** Displays shipment status in real-time.

Backend Requirements

- **Sanity CMS for Product Data:**
  - **Schema:** Manage Products, Orders, Customers, Delivery Zones, and Shipments.
  - Real-time stock updates using Sanity's GROQ APIs.
- **Third-Party API Integrations:**
  - **Shipping:** Real-time traffic updates and shipment tracking.
  - **Payments:** Secure gateways like Stripe or PayPal.

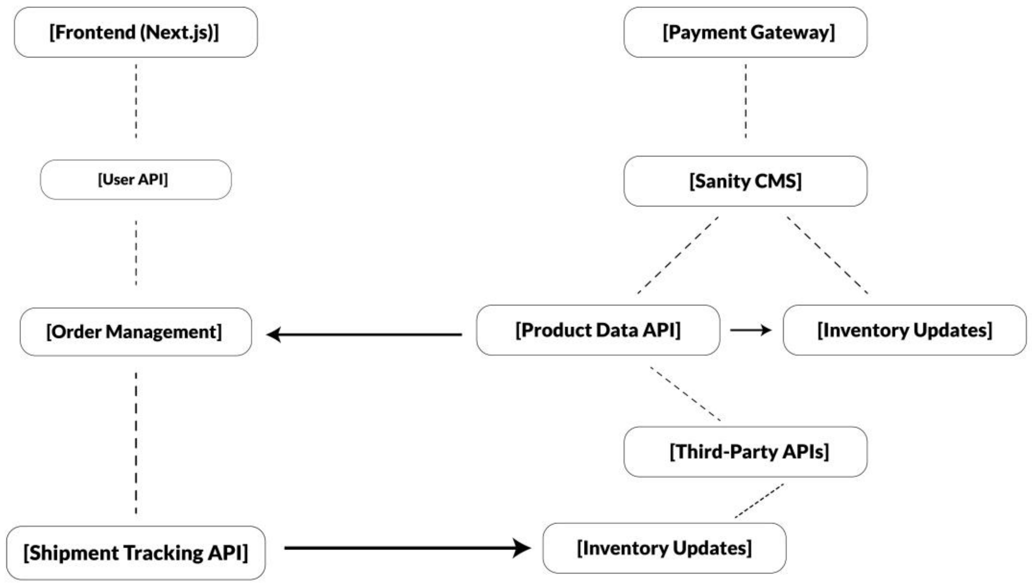
---

2. Design System Architecture

System Components

- **Frontend:**
  - Framework: Next.js for fast, SEO-friendly interfaces.
- **Backend:**
  - Sanity CMS: Data storage and retrieval.
  - External APIs: Integration for shipment and payment services.
- **APIs:**
  - RESTful endpoints for product listings, orders, and shipment tracking.

System Architecture Diagram



3. Plan API Requirements

Example Endpoints

Endpoint Name	Method	Purpose	Request Payload	Response Example
`/products`	GET	Fetch all product details	None	`{ "id": 1, "name": "Milk", "price": 50, "stock": 20 }`
`/products/:id`	GET	Fetch specific product details	None	`{ "id": 1, "name": "Milk", "price": 50, "stock": 20 }`
`/orders`	POST	Create a new order	`{ "customerId": 123, "items": [{"productId": 1, "quantity": 2}] }`	`{ "orderId": 456, "status": "Pending" }`
`/orders/:id/status`	GET	Get order status	None	`{ "orderId": 456, "status": "In`

Endpoint Name	Method	Purpose	Request Payload	Response Example
				`Transit", "ETA": "20 mins" }`
`/shipment/:id`	GET	Track shipment details	None	`{ "shipmentId": 789, "orderId": 456, "ETA": "10 mins" }`
`/inventory/update`	PATCH	Update product stock	`{ "productId": 1, "newStock": 15 }`	`{ "status": "Success", "updatedStock": 15 }`
`/users`	POST	Create a new user	`{ "name": "Ameen Alam", "email": "ameen@gmail.com" }`	`{ "userId": 789, "status": "Created" }`
`/users/:id`	GET	Fetch user details	None	`{ "id": 789, "name": "Ameen Alam", "orders": [] }`

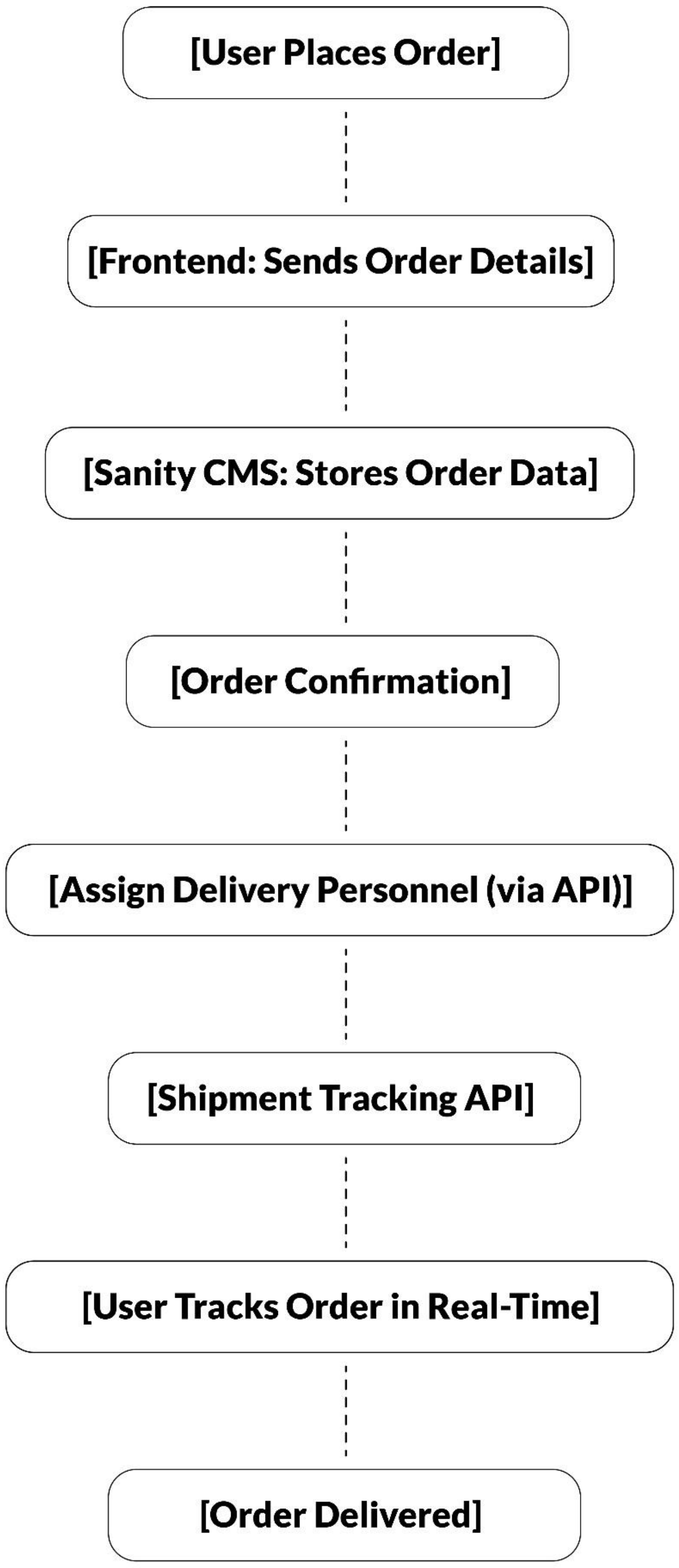
4. Technical Documentation

Sample Sanity Schema (Product)

```
javascript
CopyEdit
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 Quantity' },
    { name: 'category', type: 'string', title: 'Category' },
    {
      name: 'supplierInfo',
      type: 'object',
      title: 'Supplier Information',
      fields: [
        { name: 'name', type: 'string', title: 'Supplier Name' },
        { name: 'contact', type: 'string', title: 'Contact Information' }
      ]
    }
  ]
};
```

Workflow Example

1. **User selects products:** Product details fetched via `/products`.
2. **Order placed:** Details stored in Sanity using `/orders`.
3. **Shipment status updated:** Fetch updates from `/shipment`.


[User Places Order]
[Frontend: Sends Order Details]
[Sanity CMS: Stores Order Data]
[Order Confirmation]
[Assign Delivery Personnel (via API)]
[Shipment Tracking API]
[User Tracks Order in Real-Time]
[Order Delivered]

5. Collaborate and Refine

Tools and Practices

- **Version Control:** Use GitHub for tracking changes.
- **Peer Reviews:** Share and refine plans with team feedback.

Tools:

- **Lucidchart** for diagrams.
- **Postman** for testing APIs.

Key Outcomes

1. System Architecture: Visualized with all components.
2. API Specifications: Documented endpoints for seamless integration.
3. Sanity Schemas: Defined to manage product and order data.
4. Collaboration: Refinements incorporated from team feedback.