DAY 2 PLANNING THE TECHNICAL FOUNDATION

1) Define Technical Requirements

Frontend Requirements:

1) **Home Page**: Serves as the entry point, showcasing featured furniture, promotions, and navigation links to other key pages. Users can access the product listing, about, FAQs, or contact pages directly from here.
2) **Product Listing Page**: Displays a catalog of all available furniture products with filters and search functionality. Clicking on a product redirects users to the **Product Details Page**.
3) **Product Details Page**: Provides detailed information about a selected product, including images, descriptions, price, and customization options. From here, users can add the product to their cart or explore related products.
4) **Cart Page**: Displays all selected items with options to adjust quantities or remove items. Provides a clear call-to-action to proceed to the **Checkout Page**.
5) **Checkout Page**: Collects user information, shipping address, and payment details to complete the purchase. On successful payment, it redirects users to the **Order Confirmation Page**.
6) **Order Confirmation Page**: Confirms that the order has been successfully placed, with details like the order number and estimated delivery date. Includes a button to return to the **Home Page**.
7) **About Page**: Shares the company's story, mission, and values to build trust with customers. Includes links to the **Home Page** and **Contact Page** for easy navigation.
8) **FAQs Page**: Provides answers to common questions about products, delivery, and returns. Links to the **Contact Page** for further inquiries.
9) **Contact Page**: Offers a form for users to get in touch with customer support. Also includes links to the **FAQs Page** and social media profiles for additional assistance.

Sanity CMS as Backend:

1) Sanity CMS will store and manage all product data, customer details, and order records.
2) It works as a database for the marketplace, making it easy to add, update, or retrieve information.
3) Custom schemas will be created to match the business needs, ensuring data is organized and easy to manage.

Third-Party APIs:

1) Use APIs for shipment tracking to keep customers updated on their orders.
2) Integrate secure payment gateway APIs to handle online transactions smoothly.
3) Include other necessary APIs to enhance backend services, such as tax calculations or email notifications.
4) Ensure these APIs provide accurate data to support frontend features like order updates and payment confirmations.

2) Plan API Requirements

Endpoint Name	Method	Description	Payload (if applicable)	Sample Response
/products	GET	Fetch all available products from Sanity	N/A	{ "id": 1, "name": "Sofa", "price": 300, "stock": 10, "image": "url" }
/orders	POST	Track order shipment status via a third-party API.	{ "customerId": 101, "products": [{"id": 1, "quantity": 2}], "paymentStatus": "Paid" }	{ "orderId": 123, "status": "Pending", "total": 600 }
/shipment	GET	Track order shipment status via a third-party API.	{ "orderId": 123 }	{ "shipmentId": 789, "status": "In Transit", "ETA": "2025-01-18" }
/product-details	GET	Fetch detailed information for a specific product	{ "productId": 1 }	{ "id": 1, "name": "Sofa", "price": 300, "description": "Modern sofa.", "dimensions": "6ft x 3ft" }
/update-stock	PATCH	Update stock levels for a product in Sanity.	{ "productId": 1, "newStock": 8 }	{ "productId": 1, "updatedStock": 8, "status": "Success" }

3)Data Schema Design for Furniture Marketplace

Purpose: Define entities and relationships to organize and manage data in the database or CMS.

Key Entities and Their Fields:

1. **Products**
    - id (string): Unique identifier.
    - name (string): Product name.
    - description (string): Product details.
    - price (number): Cost of the product.
    - stock (number): Available quantity.
    - image (string): URL for the product image.
    - categoryId (string): Links to **Categories**.
2. **Categories**
    - id (string): Unique identifier.
    - name (string): Category name (e.g., Sofa, Bed).
3. **Customers**
    - id (string): Unique identifier.
    - name (string): Full name of the customer.
    - email (string): Contact email.
    - address (string): Shipping address.
4. **Orders**
    - id (string): Unique identifier.
    - customerId (string): Links to **Customers**.
    - products (array): List of ordered products with quantity.
    - total (number): Total price of the order.
    - status (string): Order status (e.g., Pending, Shipped).

    - createdAt (date): Order creation date.

5. **Shipments**
    - id (string): Unique identifier.
    - orderId (string): Links to **Orders**.
    - status (string): Current shipment status.
    - ETA (date): Estimated delivery date.

Entity Relationships:

- **Products → Categories**: One-to-Many (A category has many products, a product belongs to one category).
- **Customers → Orders**: One-to-Many (A customer can place multiple orders, an order belongs to one customer).
- **Orders → Products**: Many-to-Many (An order can have multiple products, and a product can belong to multiple orders).
- **Orders → Shipments**: One-to-One (Each order has one shipment, and each shipment belongs to one order).

Implementation in Sanity CMS:

- **Products**: Create a schema with fields for product data and a reference to **Categories**.
- **Categories**: Simple schema to define categories.
- **Orders**: Schema with references to **Customers** and an array of ordered products.
- **Shipments**: Schema with references to **Orders**.