|  |
| --- |
| HACKATON – 3  Day -2 Planning The Technical Foundation  Marketplace Technical Foundation [E-Commerce] |
|  |

1. Technical Requirements:

- Responsive frontend using Next.js

- Sanity CMS for product catalog management

- Payment gateway integration

- Order management system

- Search and filtering functionality

- Product recommendations

1. System Architecture:

[Frontend (Next.js)] <-> [Sanity CMS] <-> [Payment API, Shipping API]

1. Key Workflows:

* Product browsing and search
* Add to cart and checkout
* Order placement and tracking
* User account management

1. API Endpoints:

* GET /products - Fetch product catalog
* POST /orders - Place new order
* GET /orders/id - Get order details

1. Sanity Schema:

* Product: name, price, description, images, category
* Order: customer, items, total, status

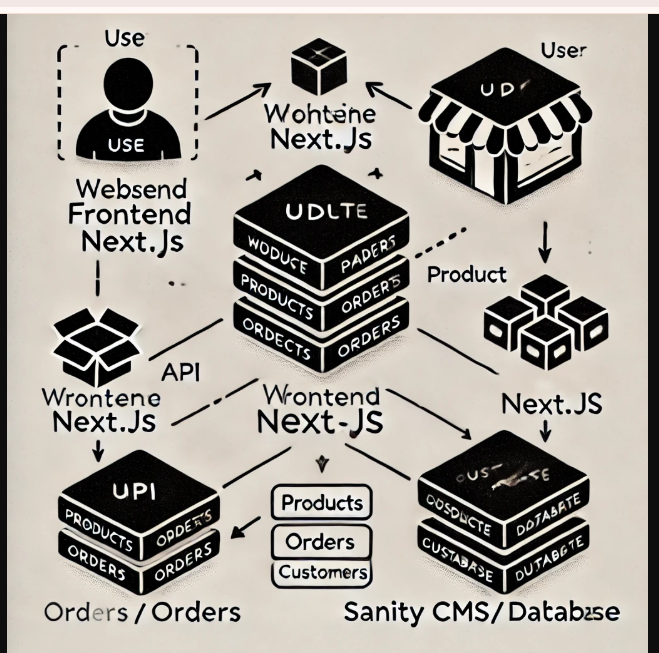
1. Best Practices:

* Optimize images for fast loading
* Implement SEO best practices
* Ensure mobile-friendly design
* Use caching for improved performance

1. Unique Features:

* Virtual room planner tool
* Augmented reality product viewer
* Custom furniture designer

Notes:

* Focus on high-quality product imagery
* Implement robust search and filtering
* Provide detailed product information and dimensions
* Offer style guides and room inspiration galleries
* Enable customer reviews and ratings
* Implement Wishlist functionality
* 

Explanation of the Next.js project structure:

1. `pages/`: Contains route components

1. `index.js`: Home page

2. `products/`: Directory for product pages

1. `index.js`: All products page

2. `[...slug].js`: Dynamic product detail pages

3. `cart.js`: Shopping cart page

4. `checkout.js`: Checkout process page

5. `api/`: API routes for server-side operations

1. `products.js`: Product-related API endpoints

2. `orders.js`: Order-related API endpoints

3. `shipping.js`: Shipping-related API endpoints

6. `tracking.js`: Order tracking page

7. `account/`: User account pages

1. `index.js`: Account dashboard

2. `orders.js`: Order history

3. `wishlist.js`: User's wishlist

8. `\_app.js`: Custom App component

9. `\_document.js`: Custom Document component

2. `components/`: Reusable React components

1. `Layout/`: Layout components

1. `Header.js`: Site header with navigation

2. `Footer.js`: Site footer

3. `Navbar.js`: Navigation component

2. `Product/`: Product-related components

1. `ProductCard.js`: Individual product display

2. `ProductDetails.js`: Detailed product view

3. `ProductList.js`: List of products

3. `Cart/`: Cart-related components

1. `CartItem.js`: Cart item component

2. `CartSummary.js`: Cart total and checkout button

4. `Checkout/`: Checkout-related components

1. `CheckoutForm.js`: Main checkout form

2. `PaymentForm.js`: Payment information form

5. `Shipping/`: Shipping-related components

1. `ShippingForm.js`: Shipping information form

2. `TrackingInfo.js`: Order tracking component

6. `Account/`: Account-related components

1. `AccountDetails.js`: User account information

2. `OrderHistory.js`: User's order history

7. `UI/`: Reusable UI components

1. `Button.js`: Custom button component

2. `Input.js`: Custom input component

3. `Modal.js`: Modal/dialog component

3. `styles/`: CSS files

1. `globals.css`: Global styles

2. `Home.module.css`: Styles for home page

3. `Product.module.css`: Styles for product pages

4. `public/`: Static assets

1. `images/`: Product and site images

2. `fonts/`: Custom font files

3. `favicon.ico`: Site favicon

5. `lib/`: Utility functions and API clients

1. `sanityClient.js`: Sanity CMS client configuration

2. `api.js`: API utility functions

3. `auth.js`: Authentication utilities

4. `cart.js`: Cart management functions

6. `sanity/`: Sanity CMS configuration

1. `schemas/`: Content type definitions for Sanity

1. `product.js`: Product schema

2. `category.js`: Category schema

3. `order.js`: Order schema

4. `customer.js`: Customer schema

5. `shipping.js`: Shipping schema

2. `sanity.config.js`: Sanity configuration file

3. `sanity.cli.js`: Sanity CLI configuration

7. `utils/`: Helper functions

1. `formatCurrency.js`: Currency formatting utility

2. `validateForm.js`: Form validation functions

3. `seo.js`: SEO-related utilities

This structure organizes our FURNIRO e-commerce website efficiently, separating concerns and making it easy to manage different aspects of your application. The integration with Sanity CMS allows for flexible content management of our product catalog, while the addition of shipping and tracking components ensures a complete e-commerce experience.

Key features of this structure:

- Clear separation of pages, components, and API routes

- Modular component structure for easy maintenance and scalability

- Integration of Sanity CMS for content management

- Inclusion of shipping and tracking functionality

- Separation of styles for better organization

- Utility functions for common operations

- SEO considerations with custom `\_document.js` and SEO utilities

This structure provides a solid foundation for building a robust, scalable, and maintainable our e-commerce website products.