DAY 4

Building Dynamic Frontend Components for Your Marketplace

Objective:
On Day 4, students will focus on designing and developing dynamic frontend components to display marketplace data fetched from Sanity CMS or APIs. This step emphasizes modular, reusable component design and real-world practices for building scalable and responsive web applications.

Key Learning Outcomes:

Build dynamic frontend components to display data from Sanity CMS or APIs.

Implement reusable and modular components.

Understand and apply state management techniques.

Learn the importance of responsive design and UX/UI best practices.

Prepare for real-world client projects by replicating professional workflows.

Key Components to Build:

1. Product Listing Component:

Render product data dynamically in a grid layout.

Include fields such as:

Product Name

Price

Image

Stock Status

Example layout: cards displaying product details.

Diagram:

| [Image] Title Price ⭐⭐⭐⭐☆ [Add to Cart] |
| [Image] Title Price ⭐⭐⭐⭐☆ [Add to Cart] |
+---+

2. Product Detail Component:

Create individual product detail pages using dynamic routing in Next.js.

Include detailed fields such as:

Product Description

Price

Available Sizes or Colors

Diagram:

3. Category Component:

Display categories dynamically fetched from the data source.

Enable filtering of products by selected categories.

4. Search Bar:

Implement search functionality to filter products by name or tags.

5. Cart Component:

Display added items, quantity, and total price.

Use state management for tracking cart items.

Diagram:

+---+

```
|                  Cart                    |
+------------------------------------------+
| [Image] Product Name   Qty: [ - 1 + ]  $XX.XX  |
| [Image] Product Name   Qty: [ - 2 + ]  $XX.XX  |
|------------------------------------------|
| Total: $XXX.XX                           |
| [Checkout]                               |
+------------------------------------------+
```

6. Wishlist Component:

Allow users to save products for future reference.

Use local storage or a global state management tool to persist data.

7. Checkout Flow Component:

Create a multi-step form for checkout, including fields for:

Billing and shipping address

Payment details (mock implementation)

8. User Profile Component:

Display user-specific details like:

Name, email, and saved addresses.

Order history with links to individual orders.

Account settings.

Diagram:

```
+------------------------------------------+
|              User Profile                |
+------------------------------------------+
| Name: John Doe                           |
| Email: john.doe@example.com              |
| [Order History]  [Saved Addresses] [Settings] |
+------------------------------------------+
```

9. Reviews and Ratings Component:

Allow users to view and submit reviews for products.

Display average ratings and individual reviews dynamically.

10. Pagination Component:

Break down large product lists into manageable pages.

Implement previous and next buttons or numbered pagination.

11. Filter Panel Component:

Provide advanced filtering options, such as:

Price range sliders

Brand selection

Availability toggles (e.g., "In Stock" only).

12. Related Products Component:

Suggest similar or complementary products on the product detail page.

Fetch data based on tags, categories, or customer behaviors.

13. Footer and Header Components:

Build consistent navigation and branding elements.

Include links to key pages (e.g., Home, About, Contact).

Ensure responsiveness and accessibility.

14. Notifications Component:

Show real-time alerts for actions like adding to cart, errors, or successful purchases.

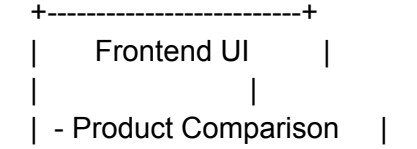
Use toast notifications or modal windows.

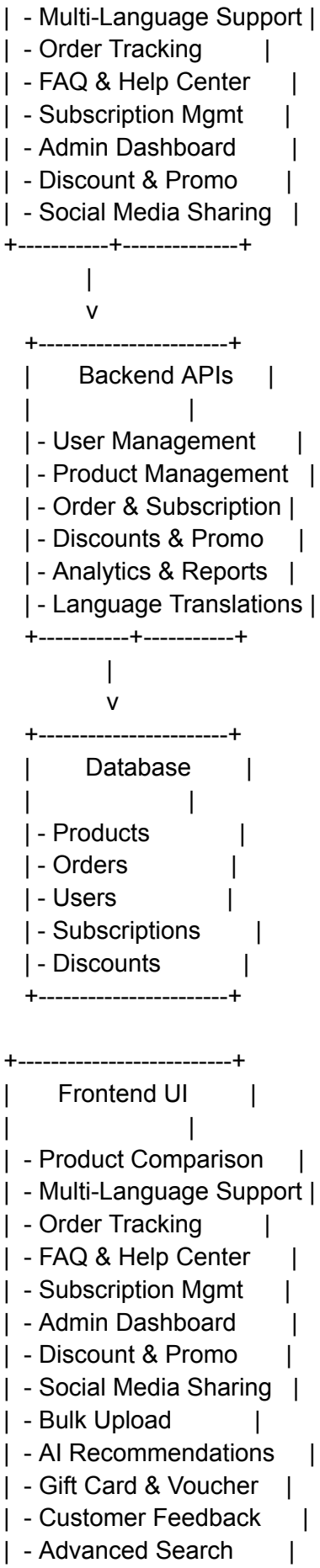
15. Analytics Dashboard Component:

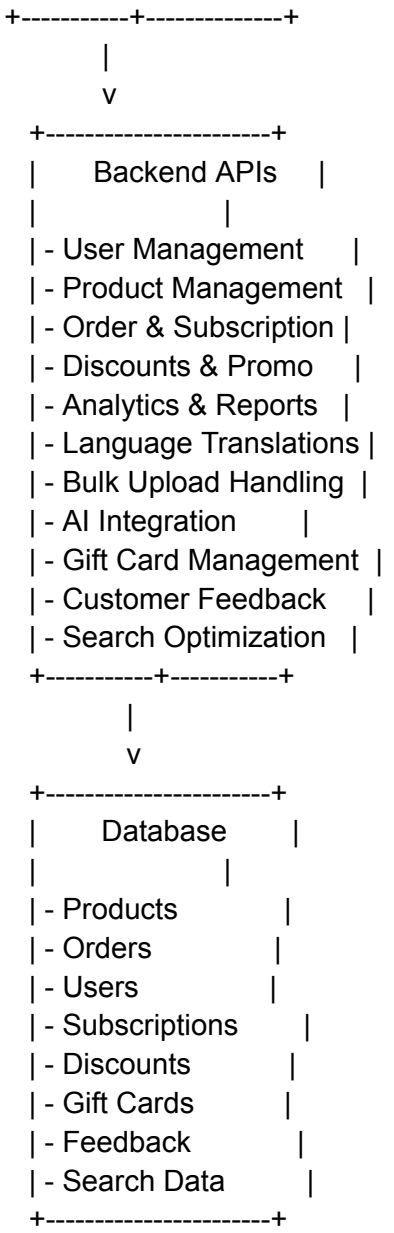
Display key performance indicators (KPIs) like sales, user traffic, and popular products.

Use charts and graphs for visual representation.

Implement UI initially, with the option to make it dynamic later.




- Multi-Language Support
- Order Tracking
- FAQ & Help Center
- Subscription Mgmt
- Admin Dashboard
- Discount & Promo
- Social Media Sharing
Backend APIs
- User Management
- Product Management
- Order & Subscription
- Discounts & Promo
- Analytics & Reports
- Language Translations
Database
- Products
- Orders
- Users
- Subscriptions
- Discounts
Frontend UI
- Product Comparison
- Multi-Language Support
- Order Tracking
- FAQ & Help Center
- Subscription Mgmt
- Admin Dashboard
- Discount & Promo
- Social Media Sharing
- Bulk Upload
- AI Recommendations
- Gift Card & Voucher
- Customer Feedback
- Advanced Search



Explanation of New Components:

Bulk Upload Component: Allows admin users to upload product data via CSV/Excel, with validation before insertion into the database.

AI Recommendations Component: Uses an AI model to recommend products based on user data (preferences or purchase history).

Gift Card and Voucher Component: Enables users to purchase and redeem gift cards and vouchers, with balance tracking.

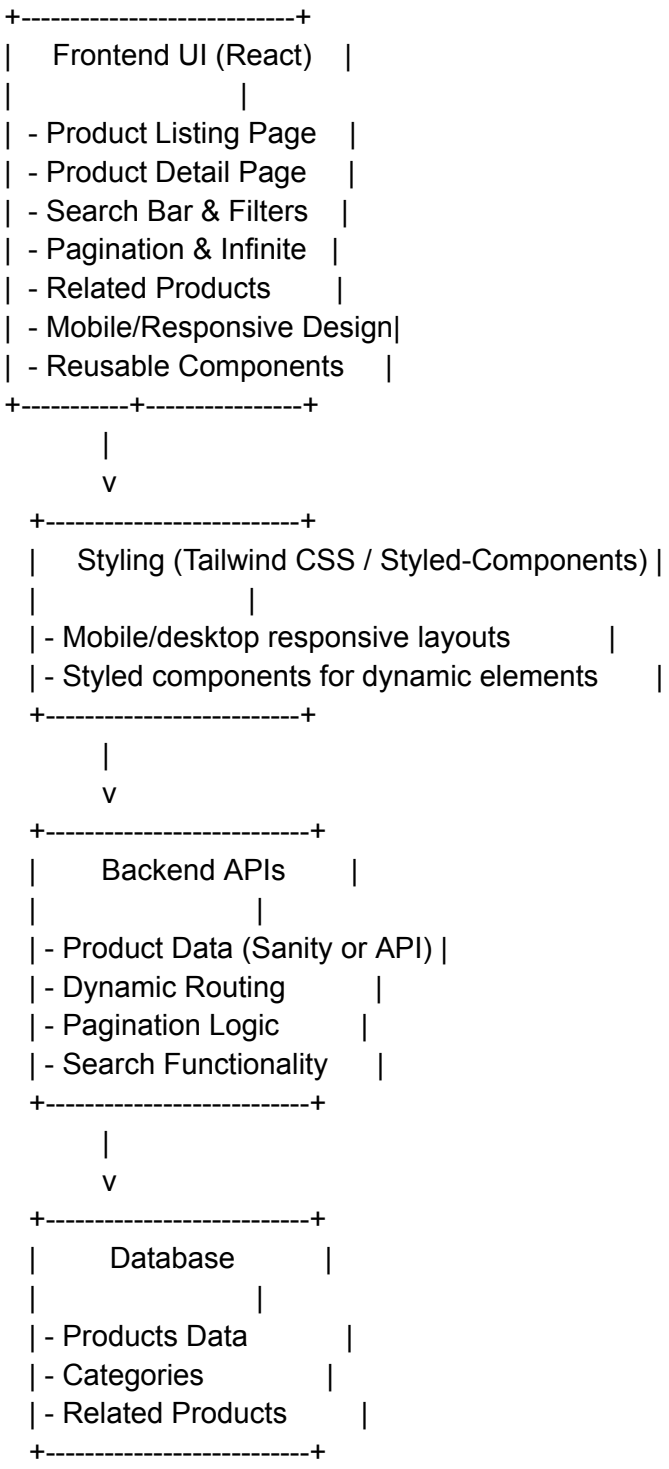
Customer Feedback Component: A form for feedback submission, with an admin interface to review aggregated feedback.

Advanced Search Component: Implements faceted search with multiple filters (e.g., price, ratings, brands).

Frontend Best Practices:

Reusable Components: Build modular components (e.g., ProductCard, CategoryFilter) for reusability across pages.

State Management: Use React's useState for local state and useContext for global state management.



Key Components:

Product Listing Page: Displays dynamic product data fetched from an API (Sanity or other).

Product Detail Page: Implements dynamic routing for individual product details.

Category Filters & Search Bar: Filters products by categories, tags, and names.

Pagination/Infinite Scrolling: Ensures large datasets are handled efficiently.

Reusable Components: Includes modular components (e.g., ProductCard, ProductList) for scalability and flexibility.

Mobile Responsiveness: Tailwind CSS ensures responsiveness across devices.

Performance Optimization: Lazy loading for images, pagination for large datasets.

Steps for Implementation:

Setup: Integrate the project with Sanity CMS or an API, test data fetching.
Build Components: Develop reusable UI components like ProductCard, ProductList, and SearchBar.
Performance Optimizations: Implement lazy loading and infinite scrolling where appropriate.
Expected Output:
Functional Deliverables:
Screenshots of product listing, detail pages, filters, and pagination.
Code Deliverables:
Key components code snippets for ProductCard, ProductList, SearchBar, API integration, and dynamic routing.