DAY 4 - BUILDING DYNAMIC FRONTEND COMPONENTS FOR YOUR MARKETPLACE

Prepared by Ameen Alam

First published: PKT 11:08 AM, January 15, 2025

Document Revision Information

Version	Date	Amendment	Author
1.0	11:08 AM, January 15, 2025	Initial release of Day1	Ameen Alam
1.1	03:00 PM, January 15, 2025	Added Examples in Day1	Ameen Alam
2.0	01:00 PM, January 16, 2025	Day 2 Release	Ameen Alam
2.1	02:00 PM, January 17, 2025	Self-Validation Checklist	Ameen Alam
3.0	04:00 PM, January 17, 2025	Day 3 Release	Ameen Alam
3.1	08:30 PM, January 17, 2025	Minor correction in link	Ameen Alam
4.0	04:00 AM, January 19, 2025	Day 4 Release	Ameen Alam

Table of Contents

Day 3 Recap:

On Day 3, you focused on API integration and data migration to prepare your projects for dynamic data handling. you explored the following key areas:

1. Understanding and testing provided API endpoints.

2. Designing and implementing data schemas in Sanity CMS.

3. Migrating data from external sources, including APIs, to Sanity CMS.

4. Learning how to handle API responses and errors effectively.

5. Ensuring that the data structure aligns with marketplace requirements.

Day 3 emphasized the importance of integrating and managing backend data sources, replicating real-world scenarios where developers work with APIs and headless CMS platforms.

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:

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

2. Implement reusable and modular components.

3. Understand and apply state management techniques.

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

5. 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 like:

- o   Product Name

- o   Price

- o   Image

- o   Stock Status

- Example layout: cards displaying product details.

2. Product Detail Component:

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

- Include detailed fields such as:

- o   Product Description

- o   Price

- o   Available Sizes or Colors

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.

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:

- o   Billing and shipping address

- o   Payment details (mock implementation)

8. User Profile Component:

- Display user-specific details like:

- o Name, email, and saved addresses.

- o Order history with links to individual orders.

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:

  - o Price range sliders

  - o Brand selection

  - o 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. (If you don't know how to do it, just add the UI; you may make it dynamic later when you find a way.)

- Use charts and graphs for visual representation.

16. Product Comparison Component:

- Allow users to select and compare multiple products side by side.

- Include details like price, features, and ratings.

17. Multi-Language Support Component:

- Add a language switcher to enable multilingual support for the marketplace. (If you don't know how to do it, just add the UI; you may make it dynamic later when you find a way.)

- Fetch translations dynamically for different languages.

18. Order Tracking Component:

- Provide users with real-time updates on their order status.

- Include details like estimated delivery time and current location.

19. FAQ and Help Center Component:

- Include a searchable FAQ section.

- Add contact forms or chatbot integrations for support.

20. Subscription Management Component:

- Allow users to subscribe to products or services. (If you don't know how to do it, just add the UI; you may make it dynamic later when you find a way.)

- Manage subscription details and renewal dates.

21. Admin Dashboard Component:

- Build an interface for managing products, orders, users, and categories.

- Include tools for analytics and business insights.

22. Discount and Promotion Component:

- Allow admins to create and manage discounts, coupons, and promotional campaigns.

- Display discounts dynamically on relevant product listings.

23. Social Media Sharing Component:

- Enable users to share products or promotions directly to social media platforms.

- Integrate APIs for platforms like Facebook, Twitter, and WhatsApp. (If you don't know how to do it, just add the UI; you may make it dynamic later when you find a way.)

24. Bulk Upload Component:

- Allow admin users to upload large amounts of product data via CSV or Excel files.

- Validate data before inserting it into the database.

25. AI Recommendations Component:

- Use a basic LLM model to recommend products based on user preferences or purchase history. (If you don't know how to do it, just add the UI; you may make it dynamic later when you find a way.)

- Display recommendations dynamically on the home or product detail pages.

26. Gift Card and Voucher Component:

- Allow users to purchase and redeem gift cards or vouchers.

- Track gift card balances and usage.

27. Customer Feedback Component:

- Create a form for users to submit feedback about the marketplace or specific products.

- Display aggregated feedback for admins to review.

28. Advanced Search Component:

- Implement faceted search to combine multiple filters and provide precise results.

- Include filters like price, ratings, brands, and more.

Frontend Best Practices:

1. **Reusable Components:**
   - Build modular components like ProductCard and CategoryFilter to reuse across pages.
   - Pass data via props to maintain flexibility.

2. **State Management:**
   - Use React state or context to manage data across components.
   - Example: Use useState for local component state and useContext for global state.

3. **Styling:**

- o Use modern styling libraries like Tailwind CSS or styled-components.
- o Ensure designs are responsive for mobile and desktop views.

4. **Performance Optimization:**

- o Implement lazy loading for images.
- o Use pagination or infinite scrolling for large datasets.

Steps for Implementation:

1. Setup:

- Ensure your Next.js project is connected to Sanity CMS or your chosen API.
- Test data fetching to confirm availability.

2. Build Components:

Expected Output:

By the end of Day 4, students should have:

1. A fully functional product listing page displaying dynamic data from Sanity CMS or APIs.
2. Individual product detail pages implemented using dynamic routing.
3. Advanced category filters to refine and segment product views dynamically.
4. A search bar that effectively filters products by name or tags.
5. Additional features like pagination and related products on detail pages.
6. Components styled to ensure responsiveness and a professional look across devices.
7. Modular and reusable components for future scalability.

Submission Requirements:

- **Document Title:** "Day 4 - Dynamic Frontend Components - [Your Marketplace Name]"
- **What to Submit:**
  - 1. **Functional Deliverables:**
    - ▪ Screenshots or screen recordings showcasing:
      - ▪ The product listing page with dynamic data.

▪ Individual product detail pages with accurate routing and data rendering.

▪ Working category filters, search bar, and pagination.

▪ Any additional features implemented, such as related products or user profile components.

2. **Code Deliverables:**

▪ Code snippets for key components (e.g., ProductCard, ProductList, SearchBar).

▪ Scripts or logic for API integration and dynamic routing.

3. **Documentation:**

▪ A technical report summarizing:

▪ Steps taken to build and integrate components.

▪ Challenges faced and solutions implemented.

▪ Best practices followed during development.

▪ Ensure the document is submitted in a professional format (e.g., PDF or Markdown).

4. **Repository Submission:**

▪ Upload all files to your designated GitHub repository, under a well-structured folder hierarchy for easier navigation.

FAQs:

1. Do I need to use all the components listed in "Key Components to Build"?

No, you should focus on building the components that are essential for your marketplace template. However, adding extra components like wishlist or notifications can enhance your project.

2. Can I use libraries or frameworks like ShadCN or Tailwind for styling?

Yes, you can use any modern CSS library or framework. Ensure that your design remains responsive and professional.

3. What should I do if I encounter errors while fetching data from Sanity CMS?

• Check your API keys and dataset configurations.

• Use the Sanity dashboard to verify that your data exists.

- Log errors to identify issues in your code.

4. How do I test my dynamic routing?

- Navigate directly to routes like /product/[id] with a valid ID in your browser.

- Use placeholder IDs to ensure the routes are functional before data integration.

5. Is it mandatory to submit documentation for Day 4?

Yes, a professional and detailed submission document is required. This document should include your process, challenges, and solutions. It demonstrates your ability to communicate technical work effectively.

6. Can I use mock data if my API integration isn't working?

Yes, you can use mock data as a temporary solution but focus on resolving integration issues for a complete project.

7. What format should the submission document be in?

Submit your document in a professional format such as PDF or Markdown. Ensure it includes screenshots, code snippets, and a summary of your work.

8. What tools can I use to debug my components?

- Use browser developer tools to inspect elements and network requests.

- Utilize console logs for debugging JavaScript code.

- Consider using React DevTools for inspecting component state and props.

Checklist for Day 4:

Self-Validation Checklist:

Frontend Component Development:

    ✓ ✗

Styling and Responsiveness:

    ✓ ✗

Code Quality:

    ✓ ✗

Documentation and Submission:

    ✓ ✗

Final Review:

✓ ✗

Professional Practices Emphasized:

- Modular and reusable component design.

- State management and dynamic data binding.

- Responsive and user-friendly UI design.

- Thorough documentation for code and processes.

This structure ensures students build scalable, professional frontend solutions applicable in real-world projects. Let me know if you need further refinements!