Project Milestone 3 - Front-end Applications

Overview

In this milestone project, you will apply your Next.js skills to build a full-stack e-commerce platform. This project will reinforce your understanding of Next.js fundamentals, data fetching, authentication, and state management while preparing you for real-world development scenarios.

Goals

The goal of this project is to evaluate your ability to:

- Implement dynamic routing and navigation using Next.js.
- Use different data-fetching strategies: CSR, SSR, SSG, and ISR.
- Implement authentication using NextAuth.js.
- Manage state effectively using Context API or Zustand.
- Develop a full-stack CRUD system for managing products.
- Write unit tests for components and key functionalities.

Expected Output

Your application should be similar in functionality and design to:

- CaliStore Example
- Noel's Fake Store Example

Important Notice!

The assignments are designed to enhance your problem-solving abilities and reinforce your foundational knowledge in Next.js development. While you may use AI tools for exploration and ideation, ensure you understand the logic and implementation behind your code. Misuse or unethical use of AI, such as plagiarism or blindly copying solutions, will have consequences. Engage actively with the material to ensure that your submissions reflect your own learning and comprehension.

Your journey as a developer is about both the **process** and the **final product**. Embrace challenges and take the time to learn!

✅ Module 4 Assignment

(Deadline: Friday, 16 May 2025 at 23:59 WIB)

🎥 Recorded Brief

📜 Scenario

You have been hired as a web developer to build an online store for a fictional company named **RevoShop**. Your first task is to develop the **product listing page** and implement routing for the product detail page. This module will focus on **Next.js fundamentals** and **data fetching**.

Audience: The website is intended for customers looking to browse and purchase products online, as well as administrators managing product listings.

Purpose:

- **For Customers:** Provide a seamless and interactive shopping experience with product browsing, cart management, and checkout functionality.
- **For Administrators:** Enable efficient management of product listings, including adding, editing, and deleting products through an admin dashboard.

📁 Requirements

You are required to implement:

1. **Home Page (Product Listing):**
    - Fetch products from FakeStoreAPI.
    - Use **Static Site Generation (SSG)** for static page like promotion, FAQ & ETC.
    - Display a grid of products with images, names, and prices.
    - Clicking a product should navigate to the **Product Detail Page**.
2. **Product Detail Page (SSR & Routing):**

- Fetch product data dynamically using **Server-Side Rendering (SSR)**.
- Display detailed product information (name, description, price, image).
- Implement a simple **Add to Cart** button.

🚀 Deliverables

For this checkpoint, you are required to submit:

1. **Deployed Website**: A hosted version of your e-commerce store.
2. **Source Code Repository**: The complete Next.js project on GitHub.
3. **Project Documentation (README file)** including:
    - Overview of the project
    - Features implemented
    - Technologies used
    - Screenshots or demo links

🔍 Grading Component

Your project will be evaluated based on the following criteria:

✅ Next.js Fundamentals

- Uses Next.js file-based routing for navigation.
- Implements pages and components following best practices.
- Uses JS syntax correctly for structuring UI components.
- Passes and handles props effectively in components.
- Manages component state using `useState` and `useEffect`.

✅ Routing & Navigation

- Implements dynamic routing with `[id].js` or similar approach.
- Uses `Link` from `next/link` for client-side navigation.
- Correctly handles route parameters in dynamic pages.
- Navigates between pages without unnecessary reloads.

✅ Data Fetching

- Implements SSG for static pages.
- Implements SSR for dynamic pages.
- Uses `useEffect()` with fetch API for client-side fetching.
- Handles API errors and loading states properly.

✅ Module 5 Assignment

(Deadline: Friday, 30 May 2025 at 23:59 WIB)

🎥 Recorded Brief

📜 Scenario

Now that the product listing and detail pages are set up, you will implement **authentication, CRUD operations, and an admin dashboard** to manage products. This module will focus on **advanced Next.js concepts**, including **middleware, state management, and unit testing.**

📁 Requirements

You are required to implement:

1. **Authentication & Middleware:**
    - Fetch users from **FakeStoreAPI**.
    - Implement **NextAuth.js** for the login process.
    - Restrict the **Checkout Page** to authenticated users using **middleware**.
2. **Shopping Cart & Checkout Page:**
    - Implement cart functionality using **Context API, Zustand or Redux**.
    - Persist cart data using **localStorage**.
    - Display cart summary and a checkout button.
3. **Admin Dashboard (CRUD with API Routes & ISR):**
    - Create an admin panel where users can **view, add, edit, and delete** products.
    - Implement **API Routes** for CRUD operations.
    - Use **Incremental Static Regeneration (ISR)** to update products dynamically.
4. **Unit Testing & Performance Optimization:**
    - Write unit tests for components using **Jest & React Testing Library**.
    - Optimize performance with **lazy loading, caching, and custom hooks**.

🚀 Deliverables

For this checkpoint, you are required to submit:

1. **Deployed Website:** An updated version of your e-commerce store with full functionality.

2. **Source Code Repository:** The complete Next.js project on GitHub.
3. **Project Documentation (README file) including:**
    - Overview of the project
    - Features implemented (including authentication & admin panel)
    - Technologies used
    - Screenshots or demo links

🔍 Grading Component

Your project will be evaluated based on the following criteria:

✅ Authentication & Middleware

- Implements NextAuth.js for the login process.
- Uses middleware to restrict access to protected routes.
- Manages user authentication state correctly.
- Redirects unauthorized users to the login page.

✅ Advanced State Management

- Uses Context API, Zustand or Redux for global state handling.
- Implements persistent state using `localStorage / cookies`.
- Updates cart state correctly based on user actions.
- Handles state updates efficiently without unnecessary re-renders.

✅ CRUD & API Fetching

- Implements API routes for product management (`GET`, `POST`, `PUT`, `DELETE`).
- Ensures secure handling of API requests and responses.
- Uses form validation to prevent invalid submissions.
- Updates UI dynamically when CRUD actions are performed.

✅ ISR & Performance Optimization

- Implements Incremental Static Regeneration (ISR) for dynamic updates.
- Uses lazy loading for images and heavy components.
- Optimizes fetch requests with caching strategies.
- Ensures good performance through custom hooks or split loading.

✅ Front-end Unit Testing

- Writes unit tests using Jest & React Testing Library with minimum coverage 50%.
- Tests key functionalities like authentication and API calls.

- Ensures no major errors in production build.
