Data Fetching & API Routes Assignment

Mastering Next.js Techniques for Efficient Data Handling

Assignment: Data Fetching & API Routes

🎯 Objective

This assignment is designed to strengthen your understanding of data fetching strategies and API routes within Next.js. By the end of this task, you will be able to:

- Implement different data fetching strategies: Static Site Generation (SSG), Server-Side Rendering (SSR), and Client-Side Rendering (CSR).
- Utilize Next.js functions: getStaticProps, getServerSideProps, and Incremental Static Regeneration (ISR).
- Create and use API routes in Next.js.
- Practice client-side data fetching using useSWR.

📚 Prerequisites

Before beginning this assignment, ensure you have completed the following:

- Assignment 1: Getting Started & Routing
- Module 4: Data Fetching
- Module 9: Pages & File-Based Routing
- Module 10: Page Pre-Rendering and Data Fetching

🛠️ Tasks

1. Products Page (SSG)

- **Objective**: Create a /products page using Static Site Generation.
- **Steps**:
  - Use getStaticProps to fetch product data from the Fake Store API.
  - **Display**:
    - Product title
    - Price

        - Thumbnail image

2. Product Details Page (SSR)

- **Objective**: Implement a dynamic route /products/[id] for server-side rendering.
- **Steps**:
    - Use getServerSideProps to fetch single product details at runtime.
    - **Display**:
        - Product title
        - Description
        - Category
        - Price
        - Image

3. Dashboard Page (CSR with SWR)

- **Objective**: Create a /dashboard page using Client-Side Rendering with useSWR.
- **Steps**:
    - Fetch data from a public API, for example, JSONPlaceholder.
    - **Display**:
        - List of post titles
        - Implement auto-refresh every 10 seconds

4. Custom API Route

- **Objective**: Develop a custom API route.
- **Steps**:
    - Create /pages/api/hello.js that returns JSON:
    - export default function handler(req, res) {
      res.status(200).json({ message: "Hello from Next.js API Route!" });
      }
    - Fetch this API from the homepage and display the message.

5. Styling

- **Objective**: Apply consistent and responsive styling using Tailwind CSS.
- **Requirements**:
    - Style product cards with a grid layout.
    - Style the dashboard list.
    - Implement a loading state with a spinner or “Loading...” text.

📦 Deliverables

- **Pages & Features**:
    - /products page using SSG.
    - /products/[id] page using SSR.
    - /dashboard page using CSR with SWR.
    - Custom API route at /api/hello.

- **UI**: Ensure a fully responsive and clean user interface.

📤 Submission Guidelines (For Next.js)

1. **Project Name**
   - Name your project folder **my-nextjs-app** (or use a meaningful project name).
2. **Push to GitHub**
   - Upload the full project folder to GitHub (except node_modules, which is ignored by default using .gitignore).
   - Please make sure package.json, next.config.js, and all pages are included.
3. **Deploy on Vercel or Netlify**
   - **Preferred:** Deploy on **Vercel** (the official Next.js hosting platform).
     - Go to https://vercel.com
     - Import your GitHub repo
     - Click **Deploy**
   - **Alternative:** Deploy on **Netlify** (also supports Next.js).
     - Watch this guide: Deploy Next.js on Netlify
     - In Netlify, make sure the **build command** is next build and the **publish directory** is .next
4. **Submit the Following:**
   - **GitHub Repository Link**
   - **Live Deployed Link** (Vercel or Netlify)

By following these instructions, you will successfully complete the assignment and enhance your understanding of Next.js. Good luck!