Assignment Overview

You are required to build a small **e-commerce-style web application**, using **Next.js** . The application should include both **frontend pages** and **backend API routes**, and demonstrate the use of **different rendering methods** across different sections of the app.

The project must include:

- A **frontend** with multiple pages using different rendering strategies.
- A **backend** using Next.js API routes.
- Integration with a **mock or real database** (MongoDB, JSON file, or any data source).
- A short **report or README file** explaining the rendering choices.

Project Description

You need to develop an online product catalog where users can browse products, view details, and administrators can manage product inventory. The system should demonstrate multiple Next.js rendering approaches as described below.

Core Requirements

1. Home Page

- Route: `/`
- Display a list of products.
- Data must be fetched at **build time**
- Include basic filtering or search (handled on the **client side**).

Purpose:
Demonstrate static rendering and show how SSG can improve performance for frequently viewed content.

---

2. Product Detail Page –

- Route: `/products/[slug]`
- Pre-generate product pages at build time, but automatically update outdated pages (e.g., every 60 seconds).
- Fetch product details from your API or database.

Purpose:
Show understanding of how ISR combines static performance with periodic regeneration for dynamic data like prices or stock.

---

3. Inventory Dashboard

- Route: `/dashboard`
- Fetch live product inventory from the database **on every request**.
- Show real-time statistics such as “low stock” or “total products”.

Purpose:
Show that you understand when SSR is needed for always-fresh or protected data.

---

4. Admin Panel

- Route: `/admin`
- Build an admin page that allows product creation or updates.
- Use **client-side fetching** to get data from your API.
- Include forms for adding and editing products.

Purpose:
Demonstrate interactive and dynamic front-end development.

---

5. Server Components (Optional – Bonus)

- Create a page (e.g., `/recommendations`) where you use **React Server Components** to fetch data on the server and render partially on the client.
- Example: Fetch recommended products server-side and render them with a client-side “Add to Wishlist” button.

Purpose:
Show knowledge of the modern App Router and hybrid server/client component architecture.

Backend Requirements

API Routes

Create API routes under `/api` with the following endpoints:

- `GET /api/products` – Fetch all products.
- `GET /api/products/[slug]` – Fetch a single product.
- `POST /api/products` – Add a new product.
- `PUT /api/products/[id]` – Update an existing product (e.g., price or inventory).

Admin routes (POST/PUT) should be **protected** with a simple key-based check or mock authentication.

Data Model

Use the following model for products:

```
{
  "id": "string",
  "name": "string",
  "slug": "string",
  "description": "string",
  "price": number,
  "category": "string",
  "inventory": number,
  "lastUpdated": "string (ISO datetime)"
}
```

You may store data in a simple `data/products.json` file or use MongoDB (recommended).

Deliverables

1. **Source Code**
    - Organized Next.js project folder with all routes and components.
    - Include `.env.example` for environment variables (if used).
2. **README.md File**
    - Include instructions to run the project (`npm run dev`, etc.).
    - Mention which rendering strategy is used on each page and why.
    - Include database setup instructions (if any).
3. **Short Report (1–2 Pages)**
    - Explain the rationale behind rendering choices for each page.
    - Describe how data flows between frontend and backend.
    - Mention challenges faced and how you solved them.
    - Include screenshots of all pages.
4. **Optional (Bonus)**
    - Include **basic authentication** for admin dashboard.

Submission Instructions

- Submit your completed project as a **GitHub repository link**.
- Ensure the project runs locally using `npm install && npm run dev`.
- Include your name and date in the README.
- Deadline: *4 Days*

Bonus Points

- Use TypeScript in your project.
- Deploy the project to **Vercel** or another hosting platform and share the live URL.
- Demonstrate **revalidation on update (ISR on-demand)**.
- Add **unit tests** for API endpoints.
- Use **modern Next.js App Router features** like layout.js and Server Components.
