ENHANCING NEXT.JS PROJECTS

Implementing State Management, Middleware, and Optimization Techniques

Assignment – State Management, Middleware & Optimization

🎯 Objective

In this assignment, you will enhance your Next.js project by implementing global state management, securing routes with middleware, optimizing app performance, and exploring Incremental Static Regeneration (ISR). These tasks will build upon your existing knowledge and skills from previous modules and assignments.

Key Learning Outcomes:

- Implement global state management using Context API.
- Secure pages using Next.js Middleware.
- Optimize app performance with Next.js features like next/image and prefetching.
- Utilize ISR for data revalidation.

📚 Prerequisites

Before starting this assignment, ensure you have completed:

- Assignment 3 (Mutations & Authentication)
- Modules:
  - Module 6: Understanding and Configuring Caching
  - Module 7: Next.js App Optimization
  - Module 11: Adding Backend Code with API Routes
  - Module 12: Working with App-Wide State

🛠️ Tasks

1. Context API – Global State

- **UserContext Creation**:

- Create a UserContext that maintains the logged-in user's data, including their name and email.
- Implement a function to update the user state.
- Wrap your _app.js or root layout with UserContext.Provider to make the context available globally.
- Display the logged-in user's name in the header across your application.

2. Middleware – Route Protection

- **Middleware Setup**:
  - Create a middleware.js file in the root of your project.
  - Implement a check for a valid token in cookies or session.
  - If the token is missing, redirect the user to the /login page.
  - Protect at least one page, such as /dashboard, to ensure it requires authentication.

3. Image Optimization

- **Image Replacement**:
  - Replace all <img> tags with <Image> from next/image.
  - Specify different width and height properties to optimize images effectively.
  - Test the lazy-loading feature for images to enhance performance.

4. ISR (Incremental Static Regeneration)

- **ISR Implementation**:
  - Create a /news page that fetches data from a public API.
  - Use getStaticProps with the revalidate property set to 10, allowing the page to update every 10 seconds with new data.
  - Display the last updated timestamp on the page to inform users about the data freshness.

5. Performance Enhancements

- **Navigation Optimization**:
  - Utilize next/link with prefetch={true} for main navigation links to enhance loading speed.
  - Ensure your project achieves a Lighthouse performance score greater than 90 for desktop environments.

📦 Deliverables

To complete this assignment, ensure you deliver:

- A working global state implementation using Context API.
- Middleware-enabled route protection.
- Optimized images using next/image.
- An ISR-enabled /news page with automatic data revalidation.
- Prefetch-enabled navigation to improve user experience.

📤 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 completing these tasks, you will have a robust Next.js application that demonstrates effective state management, route protection, and performance optimization. Happy coding!