

PHONE CATALOG HOMEWORK

The Challenge

The challenge is to create a **NextJS-based phone catalog app** from scratch. You'll be working with a modern tech stack similar to the one Sherpa uses, and the goal is to simulate the development process you'd face in our team.

Minimum Requirements

In broad terms, the outcome should meet at least the following criteria:

1. Web App for Browsing the Phone Catalog

- Built using **NextJS** and **TailwindCSS** for the frontend.
- The design should be clean, responsive, and user-friendly, ensuring a seamless experience across all devices (desktop and mobile).
- The **Home page** should display a list of phones with images.
- Implement **server-side rendering (SSR)** for faster load times and better SEO.
- It should be possible to select a phone to view its details, which should include more information like manufacturer, price, and specifications.
- A **spinner or skeleton loader** should be shown while the REST API request is ongoing.

2. REST API Providing Phone Information

- Build the REST API using **Node.js** (preferably with **Prisma** for database management).
- The REST API should have at least one endpoint:
 - **Method:** GET
 - **Path:** `/phones`
 - **Response:** A JSON array containing phone details.

Example response:

```
[
  {
    "id": 0,
    "name": "iPhone 7",
    "manufacturer": "Apple",
    "description": "lorem ipsum dolor sit amet consectetur.",
    "color": "black",
    "price": 769,
    "imageFileName": "IPhone_7.png",
    "screen": "4.7 inch IPS",
    "processor": "A10 Fusion",
    "ram": 2
  }
]
```

- Use a **PostgreSQL** database with Prisma for managing the data (you can use **Supabase** if hosting the database).

3. GitHub Public Repository

- The repository should store all code, with a clear directory structure.
- Include a `README.md` with:
 - A project description.
 - Instructions on how to run the project locally.
 - A list of technologies used (e.g., Next.js, Tailwind, Prisma, etc.).
 - Deployment instructions for both the API and the frontend.

4. Basic DevOps and Deployment

- Include a **CI/CD pipeline** (for example, using **GitHub Actions**).
- Optionally, Dockerize both the frontend and backend apps.

Nice to Have

While the points above cover the basics, Sherpa encourages going beyond the core requirements to show your creativity and technical skills. Here are some suggestions:

1. Authentication and Authorization

- Implement a simple authentication system (e.g., **NextAuth.js**, **JWT**, etc.) to protect certain features.
- Only authenticated users can create, edit, or delete phones (basic CRUD operations).

2. Additional Libraries and Tools

- Feel free to integrate any useful libraries.

3. Full CRUD Functionality

- Extend the REST API to support **CRUD** operations:
 - **POST** for adding new phones.
 - **PUT** for updating existing phones.
 - **DELETE** for removing phones.
- The React app should include forms for creating and editing phones, with validation.

4. Testing

- Unit tests for React components.
- Integration tests for the REST API.
- End-to-end tests for the entire application.

5. Persistence Layer

- Store phone data using **Supabase** as a backend-as-a-service (with PostgreSQL) or a similar service.

6. Monitoring and Alerts

- Implement basic monitoring (e.g. **Betterstack**, **Sentry**, etc.) for error tracking.
- Set up basic alerting in case of downtime or errors.

7. Deployment and Continuous Integration

- Implement a **CI/CD pipeline** using **GitHub Actions** or any other CI service.
- Set up automatic deployment on a service like **Vercel** or **Heroku** (or any other platform that supports NextJS).