Task – Car Inventory Viewer with FastAPI Backend

Build a full-stack app for a car dealership using **FastAPI** (Python) and **React + TypeScript**. The app displays a list of cars and allows filtering based on their availability.

Backend (FastAPI)

Purpose: Serve a static list of cars via an API.

Requirements:

1. **FastAPI App**
    - Set up a FastAPI application.
    - Define a single endpoint: `GET /cars`
2. **Car definition**
    - Car properties:

```None
id - integer
make - string
model - string
year - integer
available: bool
```

3. **Static Data**
    - Return the following list of cars:

```JSON
[
    { "id": 1, "make": "Toyota", "model": "Camry", "year": 2021, "available": True },
    { "id": 2, "make": "Honda", "model": "Civic", "year": 2020, "available": False },
    { "id": 3, "make": "Ford", "model": "Mustang", "year": 2022, "available": True },
    { "id": 4, "make": "Chevrolet", "model": "Impala", "year": 2019, "available": False }
]
```

4. **Protection on all routes**

   - For all future routes in this FastAPI application, enforce a rule that checks the x-device-type HTTP header. If the header is missing or its value is anything other than "WebApp", respond with an HTTP 403 Forbidden error.

Frontend (React + TypeScript)

Purpose: Display and filter car inventory.

Requirements:

1. **Fetch Car Inventory**
   - Fetch the car list from `backend`.
2. **Display Cars**
   - Render each car showing:
     - Make
     - Model
     - Year
     - Availability
3. **Filter by Availability**
   - Add buttons:
     - **All**
     - **Available**

        - **Sold**
    - Filter the list accordingly.

4. **Highlight Availability**
    - Use styles (Tailwind, CSS, or a UI library):
        - Available: Green border or text
        - Sold: Gray or red
5. **Types**

- Define a `Car` interface in TypeScript

Bonus

- Use environment variables or a `.env` file to manage the backend API base URL.
- Structure your React components cleanly (e.g., separate FilterBar and CarList components).