**Problem: Gadgets Application**

**Objective:**

Develop a **Full Stack Application** using **Angular (preferred) or MVC + .NET** with a database (SQL Server preferred). The application will serve as an **inventory management system** for a **Gadget Store**, displaying available gadgets and manage their stock.

A gadget will have at least a name, date created and a quantity in stock. A gadget can be assigned to multiple categories like Mobile, TV, Wireless, etc.

The gadget name should be unique.

You can add whatever extra info you deem necessary.

**Backend components:**

* API Endpoints:
  + List all gadgets with search and pages (partial search by name or date range for date created)
  + Get gadget by Id
  + Create gadget
  + Update gadget by id (this will allow to update stock)
  + Increase gadget stock by id
  + Decrease gadget stock by id
  + Delete gadget by id
  + Delete gadgets in bulk
  + Authentication endpoints that you deem necessary
* Implement JWT authentication
* Use EF migrations (if possible) or provide the database creation script.
* API should do validations and error handling when executing all actions
* Use standard RESTFUL methods like GET, PUT, POST, DELETE, PATCH, etc. where you see fit.

### **Frontend components:**

* Responsive layout
* Side bar menu
* Implement simple JWT authentication with the login page (preferably store users in database but to simply you can use 2 hardcoded users).
* Display all gadget data using a table with paging and search.
  + When clicking any gadget, display a detailed view of the selected (clicked) gadget by fetching its data through API.
  + On the table we should also have actions to increase and decrease the stock.
* Add a create button to provide a form to create a gadget.
* Edit button on the overview page to edit a gadget and a button to delete a single gadget.
* For the bulk delete actions use a column with “checkboxes” in the table.
* The app should switch between overview and detailed views seamlessly.
* Create fallback pages to unauthorized access (401) and not found (404)

**Evaluation Criteria**

The solution will be evaluated based on:

* **Code Quality & Best Practices** (e.g., clean code, modular structure, error handling).
* **Functional API Implementation** (success/error responses, data validation).
* **UI/UX & Responsiveness** (proper rendering, smooth transitions).
* **Routing & Navigation** (proper linking between pages and error handling).
* **Bonus:** SignalR (or similar) for refreshing when data is changed across other browsers + unit tests.

**Environment Setup:**

* Frontend
  + Angular Version: (Choose the best to suit your needs)
    - Material UI
    - Tailwind
  + MVC you can use the default theme with bootstrap or use tailwind.
* .NET version: .net core 7.0 or above.

**Delivery:**

* Create a **Git repository**
  + Use **conventional commits** that reflects the commit history properly