NAME: Product Management SYSTEM

# Features:

This particular case study is about managing products, like viewing the list of products and manipulating them

**Functionalities**:

1. Following features will be available
   1. The UI must have a dashboard, which contains buttons/links to navigate to 3 different views: Home, Products and Add Product
   2. Upon clicking on a home button/link one should be redirected to Home view
   3. Upon clicking on Products button/link, one must be redirected to a view, where products are displayed
   4. On the same view there should be facility sort the products
   5. On the same view, facility to filter products by their name should be available
   6. Every product can be deleted through click of a Delete button/link available for every product view
   7. Upon clicking the image/name of the product, one should be redirected towards a view where complete details of the product can be seen
   8. On the same view, two buttons: Edit and Back should be available, so that user can move back to the list of products view as well as to a view where product data can be viewed respectively
   9. On the edit view, one should be able to edit the existing details of the product, except the unique id of the product, code and release date
   10. Also, by clicking the Add Product button/link one should be able to be redirected towards a view, where through input fields user can enter details of the new product and save the same

# Implementation:

Create an application **product-management-system** using **next cli.** The application should contain following folders and files:

* All methods to interact with RESTful API application must be created in a service class that should be placed inside “services” folder. Service class must follow singleton pattern.
* All components must be present in the “pages” folder
* Any images, local data (if any) should be placed inside “assets” folder
* Axios should be used to interact with RESTful API application
* Entire application should be Hook based. In case error handling, if Error Boundary pattern is used, then that component can be a Class component
* You can use an HOC for Error Boundary concept too
* The application must be designed using Material UI framework
* Application must use RWD
  + Must be responsive for at least 3 breakpoints (Small, Medium, Large)
  + Preferable Usage of Flex Layout (CSS)
* For form handling Formik used and for validation it can be merged with “Joi” or “Yup” package
* The application must be PWA ready
* Should use “boostrap” CSS for styling if necessary
* Usage of Sass / Scss for styling
* ESLint must be configured for the application
* Minimum 3 components must be tested using “jest/enzyme”
* The application must follow DDD
* The application must be tested using Lighthouse for the performance and PWA readiness (Lighthouse report >= 80 (SEO & Accessibility))
* The entire back-end should be created as RESTful API, using Nest
* Use PostgreSQL as the database
* Use TypeORM to interact with PostgreSQL from the RESTful API application