**Task: Interactive Product Catalog with Search, Filtering, and User Roles**

**Objective**:

Develop an interactive product catalog application with user roles, allowing:

- Customers:
  - Browse a list of products with categories.
  - View detailed information for each product (image, title, description, price).
  - Search for products by name or category.
  - Optionally, filter products by price range (advanced).
- Administrators:
  - Add, edit, and delete products from the catalog.
  - Manage product categories (create, edit, delete).

**Benefits**:
- Full-Stack Development: Introduces the fresher to working on both frontend (Angular) and backend (Node.js) aspects of web development with multiple user roles.
- Database Interaction: Provides experience with storing, retrieving, and manipulating product data in a database based on user roles.
- User Interface Design: Focuses on creating an intuitive interface for product browsing, searching, and filtering, differentiated for customer and admin roles.
- User Authentication and Authorization: Emphasizes the importance of securing user data through login, password hashing, and role-based access control.
- Data Search and Filtering (Optional): Extends the previous version by introducing basic concepts of filtering based on user roles (customers can filter, admins can manage filters).

Frontend (Angular):

- Create an interface using Angular components:
  Customer View:
    - A product list view with thumbnails or images, titles, and optional prices.
    - A product detail view displaying a larger image, title, description, and optional price.
    - A search bar to allow users to search for products by name.
    - Optionally, implement filter controls (dropdown or slider) to filter products by price range.
    - Consider using Angular Material for pre-built UI components (optional).

  Admin View: (separate components or protected routes)
    - A product management section to:
      - List all products with details (editable fields).
      - Add new products with a form (image upload, title, description, price, category selection).
      - Edit existing product details.
      - Delete products.
    - A category management section to:
      - List all categories (editable names).
      - Add new categories.
      - Edit existing category names.
      - Delete categories (ensure no products are linked to deleted categories).

**Backend (Node.js):**

- Set up a Node.js server to handle API requests from the Angular frontend.
- Integrate with a chosen database (MySQL, MongoDB, PostgreSQL) to store product data and user information:
  - Create a schema for products (name, description, image URL, price - optional, category ID - foreign key).
  - Create a schema for users (username, password, role - customer or admin).
  - Implement user authentication using libraries like bcrypt for password hashing.
- Implement role-based authorization checks:
  - Customers can only access product data (search, filter, view details).
  - Administrators can access product data and management functionalities (add, edit, delete products and categories).
- Implement logic to:
  - Serve a list of all products with basic details (name, image, optional price) for customers.
  - Retrieve detailed product information based on product ID for customers.
  - Implement search functionality to filter products based on the search term (search product names) for customers.
  - Optionally, implement price range filtering on the backend for customers.
  - Provide product and category management functionalities for admins (CRUD operations).

**Database**:
- Choose a database management system (MySQL, MongoDB, PostgreSQL) and set up a local instance.
- Create a database and tables to store product data, user information, and categories as defined in the backend schema.