**System Overview:**

**ProductListing** is a module that’s manages Product and category listings for an online retail website. It has built in security features and built with keeping focus on possible business case and maintainability.

**Product**

Product has name, Description and price of the product.

Product is assigned to only one category.

User can enter price of the product in any currency but it will be converted to EUR and will be stored in the Database.

Product name must be unique.

Json input format for the product:

{    
   **"name"**: {name of product},  
   **"description"**: {description},  
   **"price"**: {price},  
   **"currency"**: {3 character code of currency},  
   **"category"**: {category}  
}

**Category**

Category has an hierarchical structure.

A Category can have multiple sub categories under it. Category have name and Description.

Category name is unique.

Json input format for the product:

{    
   **"name"**: {category Name},  
   **"description"**: {Category Description},  
   **"parentCategory"**: {exact name of parent category}  
}

**API Documentation:**

1. **Registration API**
2. **Category API**
3. **Product API**

**Registration API**

1. **POST /registration**

Don’t need admin level login.

Json structure of post request body:

{    
   **"userName"**: {user name},  
   **"password"**: {password},  
   **"role"**: {role one of predefined roles “Admin” or “User”}  
}

**Product API:**

1. **GET /product/{productName}**

Return JSON structure of the Product.

Example:

http://localhost:8080/product/Samsung S8

response:

{    
   **"status"**:"SUCCESS",  
   **"message"**:"Sucess",  
   **"data"**:{    
      **"productId"**:3,  
      **"name"**:"Samsung S8",  
      **"description"**:"Change1",  
      **"price"**:844.27,  
      **"category"**:{    
         **"categoryId"**:6,  
         **"name"**:"Mobiles",  
         **"description"**:"A to Z mobiles",  
         **"parentCategory"**:{    
            **"categoryId"**:2,  
            **"name"**:"Electronics",  
            **"description"**:"Electronic Gadgets",  
            **"parentCategory"**:null  
         }  
      }  
   }  
}

Returns ID, Name, Description and category hierarchy of the product.

Category hierarchy is in reverse order from closest category to parent category.

1. **GET /product/category/{categoryName} (Products under a category)**

Example:

<http://localhost:8080/product/category/Electronics>

Returns all products present under the specified category and its child categories too.

For example there is structure:

Clothing -> Men’s Clothing -> Men’s Tshirt

<http://localhost:8080/product/category/Clothing> will return all products under all 3 categories. As product “Black Tshirt” listed under “Men’s Tshirt” is also a part of “Men’s Clothing” and “Clothing”.

No admin privileges needed.

1. **POST /product (Creates new Product)**

Needs basic HTTP authentication. With admin level user.

Product needs to be specified in below JSON format in post body:

{    
   **"name"**:{name of product},  
   **"description"**: {description},  
   **"price"**: {price},  
   **"currency"**: {3 character code of currency},  
   **"category"**: {category}  
}

1. **PUT /product (updates existing Product)**

Needs basic HTTP authentication. With admin level user.

Product needs to be specified in below JSON format in post body:

{    
   **"name"**:{name of product},  
   **"description"**: {description},  
   **"price"**: {price},  
   **"currency"**: {3 character code of currency},  
   **"category"**: {category}  
}

1. **DELETE /product/{productName}**

Needs basic HTTP authentication. With admin level user.

Deletes product with specified product name.

**Category API:**

1. **GET /category/{categoryName} (get category details)**

Doesn’t need admin level login.

Return category details of specified category name in JSON.

Example response:

{    
   **"status"**:"SUCCESS",  
   **"message"**:"Sucess",  
   **"data"**:{    
      **"categoryId"**:7,  
      **"name"**:"Smartphones",  
      **"description"**:"A to Z Smartphones",  
      **"parentCategory"**:{    
         **"categoryId"**:6,  
         **"name"**:"Mobiles",  
         **"description"**:"A to Z mobiles",  
         **"parentCategory"**:{    
            **"categoryId"**:2,  
            **"name"**:"Electronics",  
            **"description"**:"Electronic Gadgets",  
            **"parentCategory"**:null  
         }  
      }  
   }  
}

Provides ID, name, description and hierarchy of parent category from bottom up style.

1. **GET /category/allChild/{categoryName}**

Doesn’t need admin privilages.

Returns all sub categories under specified category

Example response:

{    
   **"status"**:"SUCCESS",  
   **"message"**:"Sucess",  
   **"data"**:[    
      {    
         **"categoryId"**:7,  
         **"name"**:"Smartphones",  
         **"description"**:"A to Z Smartphones",  
         **"parentCategory"**:{    
            **"categoryId"**:6,  
            **"name"**:"Mobiles",  
            **"description"**:"A to Z mobiles",  
            **"parentCategory"**:{    
               **"categoryId"**:2,  
               **"name"**:"Electronics",  
               **"description"**:"Electronic Gadgets",  
               **"parentCategory"**:null  
            }  
         }  
      }  
   ]  
}

1. **POST /category (create new category)**

Needs Admin level user credentials.

Json for post request data:

{    
   **"name"**: {category Name},  
   **"description"**: {Category Description},  
   **"parentCategory"**: {exact name of parent category}  
}

1. **PUT /category (updates existing category)**

Needs admin level user credentials.

Input json format needed for put request:

{    
   **"name"**: {category Name},  
   **"description"**: {Category Description},  
   **"parentCategory"**: {exact name of parent category}  
}

1. **DELETE /category/{categoryName} (deletes category with provided name)**

Needs admin level user authentication.

Deletes the category only if there is no subcategory/product present under the specified category.

**Possible alternatives that might be an improvement:**

1. **As category structure for a website will be frequently used. This entire category structure can be cached at the server startup. We will need to maintain this cache after add/update/delete operation on category. There we can make use of spring AOP feature to update cache after each successful add/update/delete on category. (which is not likely to occur much often).**
2. **Auto reassignment of child products/child categories on deletion of category. If a category is being deleted we might think of reassigning all its child products and categories to its parent category.**