```
public IEnumerable<Product> GetAllProducts()
{
    // Assuming a database context is injected into this class
    return _dbContext.Product.ToList();
}
----
[HttpGet]
public IActionResult GetAllProducts()
{
    var products = _productService.GetAllProducts();
    return Ok(products);
}
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
2.
public Product GetProductDetails(int product_id)
{
    // Assuming a database context is injected into this class
    return _dbContext.Products.FirstOrDefault(p => p.Id == product_id);
}
----------------------------------
[HttpGet("{id}")]
public IActionResult GetProductDetails(int id)
{
    var product = _productService.GetProductDetails(id);
    if (product == null)
    {
        return NotFound();
    }
    return Ok(product);
}
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++
3.
public ApplicationUser PlaceOrder(ApplicationUser user, int productId)
{
    // Assuming a database context is injected into this class
    var order = new Order
    {
        UserId = user.userId,
        ProductId = productId,

    };
    _dbContext.Orders.Add(order);
    _dbContext.SaveChanges();
return user;
}
----
[HttpPost]
public IActionResult PlaceOrder([FromBody] PlaceOrderRequest request)
{
    return _productService.PlaceOrder(request.UserId, request.ProductId);

}
----------

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++++++++++++++++=
4.
public IEnumerable<ProductOrder> GetOrderInfo(int userId)
{
    // Assuming a database context is injected into this class
    var orders = _dbContext.Orders
        .Where(o => o.UserId == userId)
        .Include(o => o.Product)
        .ToList();

     return orders;
}
-------
[HttpGet("{userId}")]
public IActionResult GetOrderInfo(int userId)
{
    var orders = _productService.GetOrderInfo(userId);
    if (orders == null || !orders.Any())
    {
        return NotFound();
    }
    return Ok(orders);
}
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Categories
5.
public IEnumerable<Menubar> GetAllCategories()
{
    // Assuming a database context is injected into this class
    var categories = _dbContext.Menubar.ToList();
    return categories;
}
------------
[HttpGet]
public IActionResult GetAllCategories()
{
    var categories = _productService.GetAllCategories();
    return Ok(categories);
}





+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
[REQUIRED-->NOT NULL]
[MINLENGTH(3)-->MINIMUM OF 3 Characters.] (CustomException)
public class InvalidDataException : Exception
{
    public InvalidDataException(string message) : base(message)
    {
    }
} if (string.IsNullOrEmpty(product.Name))
{
    throw new InvalidDataException("Product name cannot be null or empty.");
}
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++++++++++==
```