# Erfan

# DataHandlingLibrary

Data Handling Library is a class library project that contains all the logic for handling data.

## File Post

● Within the Post namespace, all the code required to deserialize JSON data into C# objects is available.  
● PostJsonDeserialize is a class for deserializing a list of JSON posts.  
● Post is a class for deserializing a single post.

## File Comment

● Within the CommentNamespace, all the code required to deserialize JSON data into C# objects is available.  
● CommentsJsonDeserialize is a class for deserializing a list of JSON posts.  
● Comment is a class for deserializing a single comment.  
● CommentsUsers is a class used to identify the user of the comments.

## File Product

● Within the Product namespace, all the code required to deserialize JSON data into C# objects is available.  
● ProductsJsonDeserialize is a class for deserializing a list of JSON posts.  
● Product is a class for deserializing a single product.

## File User

* Inom **UserNamespace** finns all kod som krävs för att deserialisera JSON-data till c# object
* **UsersJsonDeserialize** är en klass för att deserialisera en lista av JSON-inlägg.
* **User** är en klass för att deserialisera endast en User

# CacheProvider

It is a class that inherits from ICacheProvider (interface). In ICacheProvider, there are functions that act as helper methods for storing data retrieved from an endpoint in the cache memory.  
The purpose of using the cache is that the program doesn't need to fetch data from the API every time; instead, it can use the cached data, resulting in significantly faster access to the information.

# EndPoinDeserializer

The purpose of this class is to deserialize an 'AllEndPoints.json' file, which enables access to various endpoints.

# GetEndPoints

This class opens the file *AllEndPoints.json* and deserializes the data contained in it using the *EndPointDeserializer* class.  
The purpose of this class is to provide functions that facilitate the creation of various endpoints. All endpoints originate from a single location, making it easy to change them later if desired.

Some of the functions included in the class are designed to simplify the creation of endpoints with different parameters, including:

● **selectData**: specifies which data attributes should be included in the result set.  
● **id**: identifies the user, comment, or post.  
● **limit**: sets the maximum number of items to retrieve or process.  
● **skip**: determines the starting point by skipping a specified number of items.

By providing these functions, the class makes it easy to dynamically create endpoints with different parameters and needs.

# ResponseHandler

This class contains functions for performing GET (to retrieve data) and PATCH (to update part of the data) requests, both asynchronously and synchronously. Both methods support specifying a URL or `selectData`.

In addition, there is a function called `DeserializeJsonData`, which allows for deserializing different types of JSON data depending on the type (class) provided to the functions.

The functions are designed to be user-friendly and provide flexibility by supporting both asynchronous and synchronous calls, as well as the ability to handle different JSON structures through deserialization.

# Responses

This class inherits from ResponseHandler.

In this class, there are functions to perform GET requests, with the difference that these functions (using the functions of ResponseHandler) send GET requests that specify which data attributes to include, deserialize them, and return C# objects.

E-TrackPro

CommentCellDisplays comments. The information is retrieved from the API and includes title, body, and image.

ListItemCellDisplays a product. The information is retrieved from the API and includes title, price, image, rating, and brand.

ListPostComentCellDisplays a post from a user along with the various comments written by other users under the post. The information is retrieved from the API and includes title, body, image, and tags.

ListUserCellDisplays users. The information is retrieved from the API and includes username, first name, last name, age, gender, and image.

AddCommentPageUsed to allow users to write comments under products by entering a random ID and commenting.

# PriceMonitoringPage

On this page, there are charts showing how different categories have changed in price over time. There is also another chart that displays stock prices. The user can select different categories using a ComboBox.

# ProductsReviewPage

Displays various products using the ListItemCell.

# SocialMediaDashboardPage

Displays various posts and the comments associated with them.

# UpdatePage

On this page, the user can update various properties of a user by entering their ID. After sending a PATCH request, the user receives a message indicating whether the update was successful (the value does not always change).