This function should handle a logic like a situation we are searching for an item in an online store. The function searches an item(s) among items in the database. It gets information of a user (userId) and 'criteria' for searching, then the function returns a list of items that match with the 'criteria'. The criteria can be every property in data type 'item' like name, description, or category of the item, so it has to search for all possible values of items. Since the function wants to return best match results, at the first the function searches among all names of all items if it could not find a match then go through matching among a description of items, if it could not find a match in description go through search among categories of all items in the store. It is possible that the user wants to only view items and does not login into the online store (in this case 'userId' is empty or null, the function does not need to be check for validity), so the function does not have the information about the user, but when the user logged in the system and tried to search items, the function should store a 'log' data type object from the items that the user searched, this log might be useful for future recommendations. If 'criteria' is empty or null, a 'TypeError' exception should be thrown with a description. The function may make use of a 3rd party persistence library.

Important notes: the endpoint address is "/searchItems". The URL query parameters are: "userId" , "criteria"

-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

A user might browse an item with the name of the item. It is very similar to search among item, but here the function is quicker and only searches among names of items to find the item. Iplementing the function for browsing an item. The function searches among items for finding the desired item. It is possible something be wrong and the function could not find the item, so the function must return meaningful message. It is possible that a user wants to only view items and does not login into the online store (in this case the 'userId' is null or empty, the function should not check the validity of userId), so we do not have the information about the user, but when the user already logged in the system and tried to browse items, the function should store a 'log' data type object from the items that the user browsed, this log might be useful for future recommendations.

If 'itemName' is empty or null, a 'TypeError' exception should be thrown with a description. The function may make use of a 3rd party persistence library.

Important notes: the endpoint address is "/browseItems", the URL query parameters are: "userId" , "itemName"

-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

A user may want to see his/her shopping cart, implementing the function for fetching the user's shopping cart. A shopping cart contains all items that the user added to shopping cart. The function search among shopping carts data types of all users and finds the user's shopping cart based on 'userId'. Each user has only one shopping cart, if the function finds more than one or zero shopping cart for the user it should return a meaningful response with a description. The structure of shopping cart object is available in Data Type section. The user logged in the system and tried to fetch shopping cart, the function should store a 'log' data type object from the items that the user has in the shopping cart, this log might be useful for future item recommendations. The function may make use of a 3rd party persistence library. If 'userId' is empty or null, a 'TypeError' exception should be thrown with a description.

Important notes: The endpoint address is "/fetchShoppingCart", the URL query parameter is: "userId"

-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Adding and removing items from the shopping cart are common actions for users, implementing the function to update the shopping cart based on the type of action. It finds item based on 'itemId' from database, then if the 'flagOfAction' is 'adding', the function fetches the user's shopping cart and adds the new item to the shopping cart, then save the shopping cart again in the database. It is possible user wants to remove an item from the shopping cart, If the 'flagOfAction' is 'deleting' an item, the function fetches the user's shopping cart and remove the item from the shopping cart, then save the shopping cart in the database again. if the 'flagOfAction' is not 'adding' or 'removing' the function should return a 'TypeError' exception with a description. Since the user already logged in the system and tried to update the shopping cart, the function should store a log from the items the user add or remove from the shopping cart, this log might be useful for future item recommendations. The function may make use of a 3rd party persistence library. If any of the input arguments are empty or null, a 'TypeError' exception should be thrown with a description.

Important notes: the endpoint address is "/updateShoppingCart", the URL query parameters are: "userId" , "itemId", and "flagOfAction". flagOfAction only has the values :"adding",and "removing".

-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

The last step in online purchasing is checking out the shopping cart. The information of a cart must be fetched from the database by the information of the user (each userId only has one cart). The function should calculate the summation of prices of items in the shopping cart. If the summation is zero it shows user does not have any item in the shopping cart, the function should return a meaningful response. All information of the user like the first name, last name is available in the database that function can fetch them based on 'userId'. The function for finalizing the order create an order data type object then store it in the database (FYI: a subroutine which is NOT part of this function reads order objects hourly and complete them). The function should call 3rd party persistence library for deleting the items in the shopping cart, because items in the shopping cart are purchased. The function is also responsible for checking the validity of all input arguments if one of them is invalid the function a 'TypeError' exception should be thrown with a description. Apparently the user already logged in the system then is trying to check out the shopping cart, the function should store a log from the items that the user added to the shopping cart, this log is useful for future item recommendations. The function may make use of a 3rd party persistence library.

Important notes: The endpoint address is "/placeOrder", the URL query parameters are: "userId" , "userAddress", "paymentCardNo", "CVV2" adsfadsf

-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

A user may compare a browsed item with other similar items, implementing the function to find similar items with a browsed item. The function should search among the items in the store that their name includes the name of the browsed item, if similar items are more than 20 items, The function only returns top 20 items that have rating higher rating (ex: there exist 35 similar items, The function should only return the 20 items which higher rating), if the function could not find any similar item it returns empty collection. The user already logged in the system and tried to comparing items, the function should store a log from the items that the user have seen, this can be useful for future item recommendations. The function may make use of a 3rd party persistence library. The function should also check the validity of input arguments, the function should check the information of the user, information of the item to not be empty or null, if it is invalid a 'TypeError' exception should be thrown with a description.

Important notes: The endpoint address is "/fetchTopMostSimilarItems", the URL query parameters are: "userId" , "itemName".

-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Users may want to view the list of items that they have previously viewed. All interaction of users like item they purchased, viewed, or searched are stored in database in the structure of 'log' Data Types. This function searches the items of log data type objects of all users, then refines and returns the items viewed by the user, the 'action' property of 'log' data type objects in the database for viewed items is 'viewed'. Each item in the log data type objects has a flag ('action' property) which indicates if the item was 'purchased', 'viewed', or 'searched' by the user. If there are no such items, it returns an empty collection. The function may make use of a 3rd party persistence library. The function should check the validity of the input arguments. If 'userId' is empty or nu, a 'TypeError' exception should be thrown with a description.

Important notes. The endpoint address should be "/recentlyViewedItems". The URL query parameter is "userId".

-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Users may want to view their purchase history. All interaction of users like item they purchased, viewed, or searched are stored in database in the structure of 'log' Data Types. This function searches the users' logs data types, then refines and returns items that the user purchased. The 'action' property of 'log' data type objects for purchase item is 'purchased'. Each item in the log data type objects has a 'action' property which indicates if the item was 'purchased', 'viewed', or 'searched' by the user. This function should return the items with the 'purchased' flag. If there are no such items, the function returns an empty collection. The function may interact with a 3rd party persistence library to retrieve data. If 'userId' is empty or null , a 'TypeError' exception should be thrown with a description.

Important notes. The endpoint address should be "/purchasesHistories". URL query parameter: "userId"

-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

A user may write a review of an item, implementing the function to add a new review for the item. The function should look up the item based on the id of an item, add the review to the list of reviews, and persist the updated item to the database. If the item cannot be found, a meaningful response should be returned with a description. The function interacts with a persistence library to store the data. The function should also check the validity of the input arguments. If any of the input arguments are empty or null, a 'TypeError' exception should be thrown with a description.

Important notes: The endpoint address is "/reviewAnItem". The URL query parameters are: "userId" , "itemId","comment", and "rating".

-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------