API URL Documentation

# Introduction:

This document is about explaining each of the URL paths that can be taken and where they may end up (redirects) and what might happen in each scenario.

# api/products/<str:sortType>/<str:field>

This will display the list of all products. It gives customisation in terms of how you want them to be sorted (ascending or descending) and the field (singular – one field only) that the sorting should be applied on. In the app, the options for fields are as follows:

## Relevance:

This is just the product ID.

## Price:

Can be sorted by ascending or descending.

Both the sort type entered and field are validated which means that if they’re invalid, nothing will be returned.

# api/products/<str:sortType>/<str:field>/<str:filterData>

This is the same as the above URL but it also allows the filtration of products according to the user’s liking. For example, they can filter the data by specifying price limits, specific colours and only showing products that are currently available. The filtered data comes in a custom form that is created using JavaScript functions so they cannot be tampered with by the user.

Filter data comes in the form specified below:

"price=<min-price>,<max-price>&colour=<colour>|<colour>|<colour>&available=Yes&type=<type>&new=Yes"

Each filter type can be omitted if necessary and the server that processes the filter data will pick up on that.

# api/product/<str:id>

This URL path displays the full information for a specific product. If the product ID is incorrect then the data returned will be empty.

# customers/<str:accessID>/<str:jwt>

This URL path is when users are stored or acknowledged that they have been logged in. When signing in using Google, they are authenticated in the front end. To authenticate in the back end, the server just checks that the user’s email is available on the database. If the user’s data is not available, it creates a new record storing the user’s data.

It also checks access to the database through an access ID. If the access ID is correct then the user can be stored (the access ID being correct means that the front end has communicated with the back end rather than unknown communication).

This specific URL path returns nothing (empty) to the frontend.

# api/watchlist/<str:jwt>/get

This URL path contains a JSON Web Token that contains the user’s email. This URL path retrieves all the user’s favourited products (stored in the watchlist). If the JWT is wrong, or the email contained is not correct, it returns nothing. It may also return nothing if the user hasn’t saved any products in their favourites.

# api/watchlist/<str:jwt>

This URL path contains a JSON web token which contains the user’s email, the product ID involved and whether the process should be to remove or to add the product to the user’s watchlist. If the JWT has incorrect keys then it will return nothing. If anything within the process is invalid, nothing is also returned.