The Blitz Machine API documentation

General notes

This is the documentation for the **Blitz-API**.

API detail

Products

To query about the products present in our database.

Get products

This fetches the products using certain parameters: *type* and *color*. This is also **paginated**, hence we use *skip* and *limit* too.

Curl HTTP

```
curl -X GET "https://blitz-db-service.herokuapp.com/products"
```

Add product

We use certain properties to add products to our database. > If **is_tagged** is set to `true` , **type**, **sub_type** and **color** are needed to be sent along with it. And if **is_tagged** is set to `false` , the backend would call the **tagging API** to get the suggested tags.

Curl HTTP

```
curl -X POST -d '{
    "name" : "bingokingo",
    "image_url" : "https://source.unsplash.com/random/200x200",
    "type": "shoes",
    "sub_type": "sports",
    "color": "white",
    "is_tagged": true
}' "https://blitz-db-service.herokuapp.com/product"
```

Frequencies

This is basically our **order** database collection. It records the frequencies of combinations of items bought together and then further helps us recommend the suggested product to the next user.

Buy

We send product ids of two products, the backend automatically fetches the tags and adds a suitable entry in the database (if required).

Curl HTTP

```
curl -X POST -d '{
    "top": "61811fcc5c0c5c94449b4c50",
    "bottom": "61811fcc5c0c5c94449b4c50"
}' "https://blitz-db-service.herokuapp.com/buy"
```

Add

The is for manually entering data into the database, can be used for a fashion expert involvement or just entering an entry for the first time. > If `count` is not sent, the backend automatically sets it to 1, else the value is overwritten using the new value.

Curl HTTP

```
curl -X POST -d '{
    "type_1": "shirt",
    "sub_type_1": "formal",
    "color_1": "black",
    "type_2": "pant",
    "sub_type_2": "formal",
    "color_2": "blue"
}' "https://blitz-db-service.herokuapp.com/add"
```

Recommendations

We send the **id** of the product for the server to find suitable recommendations using the **frequencies** database.

Curl | HTTP

```
curl -X GET "https://blitz-db-service.herokuapp.com/recommendations?id=6186dd92cb55c9cff3ec7e7c"
```

Recommendations using Tags

Here instead of product id, we directly send tags to get the recommendations

Curl | HTTP

```
curl -X GET "https://blitz-db-service.herokuapp.com/recommendationsUsingTags?type=pant&sub_type=informa
l&color=black"
```

Cloudinary

We use cloudinary to store our images.

Upload

We send the `base64` encoding of our image in the body so that the server can use it to regenerate the image and upload it to `cloudinary` .

Curl HTTP

```
curl -X POST -d '{
    "image": "{base64}"
}' "https://blitz-db-service.herokuapp.com/upload"
```

Upload and get recommendations

Similar to the `\upload` but it also sends recommendations by extracting the tags of the image and calling `\recommendations` in one go.

Curl HTTP

```
curl -X POST -d '{
    "image": "{base64}"
}' "https://blitz-db-service.herokuapp.com/upload-and-recommendation"
```

The Tag Detection Server

Now this server is an entirely **independent** server. The frontend provides the facility to use this. This was made using `flask` rather than `nodejs` because of the preprocessing of the image that had to be done in `python` .

Get Tags

Curl HTTP

```
curl -X GET "https://blitz-tf.herokuapp.com/get-tags?url=https://source.unsplash.com/random"
```