CMPE352 - Milestone 2 Report

A. Furkan Varol
Aleyna Kara
Aslı Aykan
Burak Çetin
Furkan Nane
Hande Şirikçi
Hüseyin Seyyid Kaplan
Mehdi Saffar (Communicator)
Mehmet Umut Öksüz
Ömer Faruk Deniz
Özdeniz Dolu

May 27, 2020

Contents

1 Executive Summary

1.1 Introduction

Getflix is an online commerce platform that helps users find a plethora of products they need and purchase them in an easy and straightforward way. Getflix removes the friction of the shopping experience by providing an easy-to-use interface on all major desktop and mobile platforms. Shoppers are able to find products with the means of semantic search, view product details and decide whether a product is worth it by reading comments by verified buyers. When their shopping cart is finalized, they can easily buy their products with debit/credit card. Getflix also gives the opportunity for buyers to track price changes so that they can strike better deals. In Getflix there are four types of users: Guest, Customer, Vendor and Admins. Guest users are allowed to do basic operations such as searching products and viewing their details, as well as add products to their shopping cart. Customers are registered users that are allowed to continue with checking out process. They can also track their orders, ask for support from Admins and comment on products they have purchases. Vendors are allowed to setup an online shop where they can present their products on our platform. Admins maintain the system and make sure that operations go smoothly.

1.2 Work done so far

At the very beginning, we created a poll and decided to develop our API in the JavaScript ecosystem. One of our friends were quite experienced therefore he redirected us during the learning and development phase of the APIs. Before getting into the development phase, we decided which APIs will be developed and formed groups for the implementation. Each API is developed by two members and those members were also responsible for writing the tests. We then deployed our system to Amazon AWS and also created a description of our platform.

1.3 Road ahead

After getting our first team-wide API development experience, most people now believe that we should put considerable amount of effort to get better at the web development technologies. Also considering the current pandemic crisis, it was harder to work on the same code together without being in the same physical environment. Also reviewing many different works of groups, and sustaining a proper communication of the progresses of these works was hard. But we believe that these experiences will grow us as a team and make each one of us an excellent team-player. When it comes to future work on the platform, we will definitely be working on improving the design of our platform. We will be learning new technologies for efficient development phase and continue to improve API's of our platform.

1.4 Challenges we met as a group

There were some challenges we encountered as a group along the way. A lot of people in the group used git, react, JavaScript, jest unit testing etc. for the first time so there were some misunderstandings and hard times. Mehdi and couple of our group members helped our group to deal with most of the challenges and we thank them for their efforts.

2 List and status of deliverables and Evaluation of the status of deliverables

Deliverable	Delivery Date	Status
Milestone 2 Report	27.05.2020	Complete ✓
User Interface of our API	27.05.2020	Complete ✓
Route code of group API	27.05.2020	Complete ✓

2.1 Evaluation of Deliverables

- **Milestone 2 Report:** Preparation of this report is shared among some of the group members and overall design and responsibility is on Furkan Nane. This document is delievered via Moodle and can also be found at GitHub page of group 2.
- **User Interface of our API:** User interface of our group's API is prepared by collectively. Each member created their part of the user-interface. It seems that our group has done a good job of creating a pretty and effective front-end. Users can interact with this user interface easily. URL can be found at GitHub page of group 2.
- **Route codes of group API:** Our group has one API that consists of many routes. Each team member created their own route or created their route with their partner. During the implementation several branches were created and after necessary reviews they got merged into master. Route codes of our group API is stored at GitHub.

3 Summary of Work done

A. Furkan Varol	After the meeting about API tasks, I started working on the tranlation functionality of the project. I have searched on translation APIs for some time. Most viable and reliable option was Google Translate, and thus I decided to use that. I was not that much familiar with JavaScript and ReactJS, so I looked into them for a bit. First, I put some basic html to the front end in order to get input. Then, I created the backend for translation. The first version was taking input text from user with or without explicitly stating the source language and translate it to local language of the user's location. The location of the user was retrieved by using the IP address. However, after Maps API is implemented, I discussed with my group members and we decided that it would be much more convenient and this feature would be more 'flavorful' if the target language of the translation could be selected on the map. The necessary changes are done on the maps API and I implemented the selection of the target language. I have written unit tests to test corner cases. Also, I have reviewed pull requests of other group members and given feedback.
Aleyna Kara	I have utilized Google Product Search API with Aslı Aykan in order to show the similarity between the product that is given by the user and products that are stored in the system with applying machine learning algorithms of this API. To use this functionality of Google Product Search API, users can create their products, product sets and images. Furthermore, the update and delete operations of product and product sets are allowed to users. Therefore, I have written parts which respond to 'POST' request with the type of productSet, 'DELETE' request with type of 'product' and 'PATCH' request in `practice_app/pages/api/getProductCatalog.js`. I have worked on `practice_app/pages/api/getAllProducts.js` in order to get all products the product catalog currently has. I have committed `practice_app/pages/api/getProductsInSet.js` in order to retrieve products belonging to the product set that is given by end user. After the completion of the implementation of functions regarding to the product catalog, we have written `practice_app/pages/api/getSimilarProducts.js` in order to sort products with respect to similarities relatively to the image which end user provides. We have always been in contact with Aslı Aykan via zoom and practice pair programming during the implementation of the project. Since we have not enough experience on web development, we have gone down that road together and motivate each other. Overall, it took our two weeks to search next.js, learn the essentials of JavaScript, complete the project. In this two weeks, I have also written two tests in `practice_app/pages/api/getProductCatalog.test.js` which determines whether the update function of `practice_app/pages/api/getProductCatalog.js` works properly or not. Then, I have reviewed Reddit API and I have examined general operations of Covid API.

Asli Aykan	Me and Aleyna Kara worked on the Google Product Search and Vision API to add some basic functionalities both for vendors and customers to our e-commerce platform. Before starting, I had to study for Next.js because I had no prior knowledge on it, so we watched several tutorials about it with Aleyna Kara. After studying, we decided to add 11 functionalities to our API in total. My contribution was to add 5 functionalities which are to list all sets, to create product, to filter products by category in a product set, to delete set and to add reference image for a product. (Create product, delete set and add reference image parts are on `practice_app/pages/api/getProductCatalog.js`, filter products part is on `practice_app/pages/api/getProductsByCategory.js`, list all sets part is on `practice_app/pages/api/getSets.js`.) However, all functionalities should be compatible with each other and needed to be well tested before adding another functionality, so we needed to be always in touch. We corrected each other's mistakes and gave advices while coding via Zoom. We also had to understand how to use Google Cloud platform to add and save pictures. After completing the code, we studied Jest to write unit tests for our platform. I wrote two unit tests for creating product and product set, which are on `practice_app/pages/api/getProductCatalog.test.js`). I reviewed Burak's Reddit API and Furkan's Covid API and gave advices or solution ideas about some small errors.
Burak Çetin	After our group decided on using Next.js first of all we had a meeting to decide on the API's we will use. At first I was assigned to a group who will work on Twilio API (An SMS based service.) but after consideration we decided it can be problematic if we run out of the free trial of this API. Then I decided to switch to Reddit API and started reading about it. At the same time I learned basics of JavaScript because I had no prior knowledge on the language. I decided to implement the front-end demo for this API as a simple post collector from country based subreddits. Since other API's we are including in the project are also about information about countries I think this was a good decision. I implemented basic GET requests to the reddit API and it collects top 10 posts from a designated subreddit. And I took the selected country information from the index.js page as the designated subreddit. After making sure it did not create errors on corner cases I implemented unit tests using Jest as our team decided upon.
Furkan Nane	For the implementation i tried to learn JavaScript, HTML, git version management system, jest unit testing framework and many more things. I searched them on the web and asked questions to Mehdi Saffar and he was kind enough to help me. My partner was Ömer Faruk Deniz and i found the API we used on the web about Covid-19 and shared it with my partner. First i created my branch and after that i created my API route called getStatistics.js that finds certain statistics about a given country. I also wrote some unit test for it called getStatistics.test.js . Then i created a front-end for it. After all these i created a pull request for my branch. At this point Ömer Faruk had not started yet so i gave him a brief and clear explanation about the task he has to perform. After finishing my work early i tried organize my group with the help of Mehdi Saffar and took the responsibility of checking the assignments from Piazza and creating the Milestone 2 report using overleaf.com in latex as well as distributing tasks among the team members. My goal was to remind certain things to my team members constantly to make sure that nobody falls behind in these pandemic days. I reviewed the implementation of several members and at one point, while testing a branch I found out that one of our group members overwrote me and my partner's merged changes. I identified the overwritten lines and re-committed them to master branch and by doing so a disaster was prevented :)

Hande Şirikçi	After the meeting that we have done to organize API tasks, we have decided on working with Ozdeniz Dolu and adding a price conversion route to our API by processing on two different third party API's .Because that was the first time that i am coding in JavaScript, i watched several tutorials and Ozdeniz also helped me to get familiarize with JS. Ozdeniz completed the part that takes ip address and returns the currency type of country that ip address belongs to. After deciding which API that i will use, i implemented the part that takes target currency type and price of the product in any other currency type then calculates the price of the product in target currency type by using up to date currency values taken from API response. Because we completed our task much earlier than deadline we have not decided on test procedure as a group yet. So i checked my code manually and made a pull request. After deciding on using Jest by the all group members, i watched a tutorial about it and we revised our code with Ozdeniz to make it testable. After revising the code, it has been modified much and we decided to delete first pull request and created a new branch. I added 7 unit tests to check my part of program. Ozdeniz made a pull request and i checked the part that he added to index.js file after my commit. I have also reviewed other group members' codes and given feedback.
Mehdi Saffar	I opened a poll for deciding the language to be used in the project. I volunteered to prepare a project template which comes with a README, comments about different parts and where things should go, examples of a route and how to call it from the frontend, and an example of unit test. I have also created a Husky pipeline to make sure that every commited code gets linted and formatted properly. I wrote and tested the Dockerfile. Through multiple announcements, and with the help of Furkan Nane, I described how the template is to be used, how pull requests should be etc. I asked any questions we had on Piazza. I coordinated with my subgroup teammate Umut to build our Location from map feature. To keep things short here, I have built the frontend which is made of `components/MapView.js` and some code in `index.js`. I have also wrote unit tests under `services/getLocationInfo.js`. I have extensively reviewed the pull requests of many of my peers, making sure that a certain standard of code quality is held. I have helped Asli and Aleyna on multiple occasions through collaboration sessions over Visual Studio Code LiveShare, as well as many other members. I have also made multiple commits to their branch to fix things myself since we were pressed by time. I have opened a Docker Hub and new AWS Account and did all the step required to deploy the app. Finally I announced the website URL to our Slack channel.
Mehmet Umut Öksüz	After we decide JavaScript as our development language, I and Mehdi have taken Maps api as a subgroup. We were supposed to implement a map that shows country, language and currency of selected location. First I have researched various third party API's to see which can help us. Then I have decided that we can use positionstack. I have created a positionstack account and got an api key for us. Then using that api key I have implemented our api request and handled response. I also have written 9 unit tests to test any corner case. I have also reviewed other group members' codes and given feedback.
Ömer Faruk Deniz	In this assignment, I was partner of Furkan Nane in Covid-19 API. We decided to use this API considering the need for fresh information about the Covid-19 in the public. We took the data from the apify.com's Covid-19 API. In my part, I tried to give the Top 10 Death Statistics in the world in a bar chart. Considering the fact that I had nearly no experience in JavaScript previously, I first tried to learn the basics of JavaScript. Then I analyzed the previously written ReactJs codes in the web. In the code, I took the data from the API and found the top 10 countries with maximum number of deaths. Then I plotted them in a bar chart using chartjs. In the unit tests, I checked wheter the number of countries and number of death numbers is equal to 10. Also, I reviewed the code of my friends and helped them overcome the problems they have during the development phase.

Özdeniz Dolu	I have partnered up with Hande Şirikçi and we decided to add a price conversion route to our API. Together we have implemented the /api/getConvertedPrice. We have created a branch called locationcurrencyapi and worked on it. I have created the file `practice_app/pages/api/getConvertedPrice.js`. I have written the part in the index.js where we demonstrate how getConvertedPrice works. In the file `practice_app/services/getConvertedPrice.js`, I have created the functions getCountry,getCurrency,processIPInfo,getInfoFromIP and first part of getConvertedPrice. In the file /`practice_app/pages/api/getConvertedPrice.spec.js`. I also gave a code review for the pull request 91.
Hüseyin Seyyid Kaplan	*No effort documentation was received from this team member*

4 Evaluation of tools and processes you have used to manage your team project

4.1 Next.js by Mehdi Saffar

After voting for using Javascript for the API assignment, I took the initiative of building the inital template for our project. We had to have a way to build frontend and backend easily. One way would be to create a server that serves only the frontend and another only for the backend. This setting would slow down the development as one would have to spin up two servers any time they would like to develop a new feature, and would have code spread over two projects. Instead, Next.js allows to have a unified environment where you could write client-side and server-side code easily. It also comes preconfigured with a testing framework so that adding unit tests is easy. Next.js uses React for the frontend which allows for declarative/functional approach for building user interfaces using a syntax very similar to that of HTML called JSX. Thanks to Next.js we were able to build a frontend and backend easily.

4.2 Jest by Mehdi Saffar

Jest is a popular Javascript unit-testing framework that comes bundled with Next.js. It works both on the client-side (browser) and server-side (node) environment. It is very easy to use and has provides a rich terminal output with satisfying colors. One neat thing about it is that it has a watch mode, which executes the tests every time a change in code is made. This allows for fast feedback which is especially crucical for TDD (Test Driven Development). It is also smart enough to only run the tests whose results are most likely to be affected by the current code change.

4.3 Husky by Mehdi Saffar

Husky is a Javascript package that allows developers to insert git hooks into their Javascript project. With the power of Husky, it is possible to do some processing when a developer makes some changes to the code and commits it. In our case, I used Husky to enforce code quality and format with eslint and prettier .

4.4 Eslint by Mehdi Saffar

ESlint is a linting tool for Javascript. It prevents common code problems by analyzing the source code and pattern matching against its continously increasing database. Not only does it warn for code mistakes, by reporting what is wrong with and where it occured, it also allows to fixing mistakes automatically if it knows that it is safe to do so. I have used ESLint with its autofix mode to get rid of a certain class of problems, and have added it to the Husky pipeline.

4.5 Prettier by Mehdi Saffar

Prettier is an opinionated Javascript formatting tool. With a single command, it goes through all the files and formats them according to strict rules. This removes any trailing whitespaces, breaks down long lines into shorter neater lines and much more. It is part of our Husky pipeline which guarantees that everyone's code looks the same, improving readability of both our code and the git diffs, since it prevents commits that only contain formatting difference.

4.6 Docker by Mehdi Saffar

Docker is containerization tool that we use for deploying our app. With a simple Dockerfile description, we are able to create a clean, predictable, reproducible environment for our app. By publishing the image to Dockerhub, we are able to deploy our app to AWS EC2 instance. It is easy to use and we practically only had to use two commands, one for building an image and one for running it.

4.7 Github Pull requests by Mehdi Saffar

Github Pull requests is probably the best tool we have used for this project team. Once a member finishes developing a feature or subfeature and testing it, they can easily open a pull request to merge that branch into master. Through its slick and modern interface, the developer is able to request its peer to review their work. We have asked that every one who opens a PR must provide a clear title, description and a test report. I have reviewed many pull requests and I love the Github interface for doing so. By going into the Files Changed tab, I am able to see exactly which lines changed when compared to master. I am able to comment on a specific line or even a group of lines in one go, which makes the feedback very specific and localized. I also like that it provides a Review session workflow which considers all review comments as part of one atomic review session. Once review is done, it gives the choices between simply commenting, approving PR or requesting changes. This allowed us to keep a certain standard when it comes to our code quality.

4.8 Reddit API by Burak Çetin

Reddit API is a really good starting point for someone who is new to REST API's and similar concepts. It is doucumented well with an active community on /r/redditdev where one can ask questions and see example projects. Also, as long as your script do not add posts or comments to reddit forums there is no need for an API key so this also lowers the knowledge barrier for entry. Though I used a really small poriton of the data provided by

the API about posts one can parse or collect a lot of real world text data through reddit API without hassling with web crawlers.

4.9 PositionStack by Mehmet Umut Öksüz

To implement our map functionality we decided to use PositionStack API which is a Rest Api for forward geocoding and reverse geocoding. It can be used freely up to 25.000 requests per month. We got a free API key and used its reverse geocoding feature in our implementation. PositionStack's reverse geocoding function returns a response in json format containing various information about location specified with coordinates as a query parameter. To get detailed data like language, currency one needs to set another parameter in query that is countrymodule. When we set countrymodule = 1 than API returns all data that we need.

4.10 ChartJS by Ömer Faruk Deniz

To show the top 10 death statistics in the world, I used chartjs. It has a clean and tidy documentation and sample videos with a lot of easy-to-use features.

4.11 Google Translate by Alaaddin Furkan Varol

After intense searching for a translation API, I can say that Google Translation is the most reliable for short translations. It is certainly useful in a pinch to translate a few words or phrases. It is fast, has auto-detect source feature, and accurate if you are not translating a whole page.

4.12 Google Product Search and Vision API by Aleyna Kara and Aslı Aykan

Description by Aleyna Kara: Google Product Search & Vision API is a very useful API for our e-commerce project. It has lots of functionalities and a well-written documentation in which you can descriptions, possible errors, features of parameters and examples of each function. This API consists of products and product sets. We thought that each brand should have its own set and be able to add a product, delete a product, update a feature of a product, create a product set and delete a product set. On the other hand, customers should be able to reach products by listing all products, products in a specific set, all sets and products in a category with filtering feature. Lastly, it has also a very unique functionality for customers. A customer just enters the file path of a product picture on his/her own computer and similar products on Getflix will be listed by applying machine learning and comparing the pictures of products. With the help of this unique feature, Getflix can easily stand out among other e-commerce platforms and be preferred by a wide range of customers.

Functionality by Aslı Aykan: Here are the brief descriptions of basic functionalities from the Google Product and Vision API which we also used in our API.

Functionalities for Vendors:

- **Create product set**: Vendor can create its own set by entering unique set id and set name, and if a set with that id doesn't exist, a new set will be created.

- **Delete product set**: Vendor enters a set id and if a set with that id exists, it will be deleted.
- **Add product to a product set**: Vendor enters unique product id, set id, product name, product category, product description, color and style to add that product to that product set.
- **Update product**: Vendor enters product id, key name and value for that key of the product to update the key value.
- **Delete product**: Vendor enters the unique id of the product s/he wants to delete and product will be deleted.
- **Add reference image for a product**: Vendor adds a picture from Google Cloud storage for a product by entering unique product id and reference image uri. All pictures are stored in Google Cloud storage.

Functionalities for Customers:

- **List all products**: Customer can see the list of all products.
- **List sets**: Customer can see the list of all sets.
- **List products in product set**: Customer can see the list of all products in a set by entering unique set id.
- **Filter products by category in a product set**: Customer can see filtered products of a specific set by entering unique set id and category name.
- **List similar products**: Customer can see whether there exists a similar product for a product that s/he wants to buy or look for by giving a product picture url(file path) from his/her computer, its category name, filter (ex. style=men), and list for similar products and similarity scores will be shown as a sorted list by their scores.

5 API documentation and URL

URL: http://ec2-18-208-120-34.compute-1.amazonaws.com/

5.1 GET /api/getConvertedPrice

Finds the currency of the country that given IP belongs to, converts the given price to that currency using exchange rates at the moment of the request.
Parameters:

- *ip*: any valid IP address
- *price*: a number representing the price
- *srcCurrency*: a currency code representing the currency of the price

Response: Response in JSON. With the following information:

- *price*: a number representing the converted price
- *currency*: a currency code representing the currency of the price

5.2 GET /api/getLocationInfo

Given longitude and latitude information, returns currency, language and country information corresponding to that location.
Parameters:

- *lat*: latitude
- *lng*: longitude

Response: Response in JSON. With the following information:

- *valid*: validity of the response
- *country*: name of the country that the given location belongs to
- *currency*: a dictionary containing: symbol: symbol of the currency of the country that the given location belongs to. code: code of the currency of the country that the given location belongs to. name: currency of the country that the given location belongs to.
- *language*: a dictionary containing: name: language of the country that the given location belongs to. code: language code of the of the country that the given location belongs to.

5.3 GET /api/getStatistics

Returns recent statistics related to COVID-19 pandemic for the given country.
Parameters:

- *country*: name of the country

Response: Response in JSON. With the following information:

- *answered*: validity of the response
- *deathToll*: death toll for the given country
- *recovery*: total number of recovered for the given country
- *infection*: total number of infected for the given country

5.4 GET /api/getCountriesWithMaxDeaths

Returns names and death tolls of the ten countries with the highest death toll of Covid-19 in the world.

Response: Response in JSON. With the following information:

- *keys*: an ordered list of country names
- *values*: an ordered list of death tolls

5.5 GET /api/getRedditData

Collects titles and permalinks of top 10 posts from a given subreddit name.
Parameters:

- *country*: name of the country(or any subreddit)

Response: Response in JSON. With the following information:

- *answered*: validity of the response
- *posts*: a list of top 10 posts where each element of the list is a dictionary with the keys: title: title of the post. permalink: permalink of the post

5.6 GET /api/getTranslation

Using a country code, translates the given text to the language spoken in that country.
Parameters:

- *text*: text to be translated
- *sl*: name of language that the text is written in
- *countryCode*: country code of the country(iso code of target language of the translation)

Response: Response in JSON. With the following information:

- *answered*: validity of the response
- *sourceLanguage*: name of language of the input text
- *translation*: translated text

5.7 /api/getProductCatalog

5.7.1 POST /api/getProductCatalog

Creating a new product and then adding it to the product set that is given.

Parameters:

- *requestType*: A string representing the type of POST request
- *productId*: The id of the product
- *productDisplayName*: Display name of the product
- *productCategory*: Category of the product
- *productSetId*: The set which the product will be added
- *productDesc*: The description of the product
- *color*: Color of the product

Response: Response in JSON. With the following information:

- *statusCode*: The status code of the response
- *statusText*: The status message of the response

Creating a product set.

Parameters:

- *requestType*: A string representing the type of POST request
- *productSetId*: The id of the product set that will be created
- *productSetDisplayName*: The name of the product set that usually used for specifying it

Response: Response in JSON. With the following information:

- *statusCode*: The status code of the response
- *statusText*: The status message of the response

Creating a unique reference image of the product and linked it to the product.

Parameters:

- *requestType*: A string representing the type of POST request
- *productId*: The id of the product that the image will be added

Response: Response in HTML. With the following information:

- *statusCode*: The status code of the response
- *statusText*: The status message of the response

5.7.2 PATCH /api/getProductCatalog

Updating the labels of the product.

Parameters:

- *productId*: The id of the product whose label will be changed
- *key*: The key of the label which is linked to the value that will be changed
- *value*: The field of the label

Response: Response in JSON. With the following information:

- *statusCode*: The status code of the response
- *statusText*: The status message of the response

5.7.3 DELETE /api/getProductCatalog

Deleting a product from the product catalog that stores all products and product sets.

Parameters:

- *requestType*: A string representing the type of DELETE request
- *productId*: The id of the product that will be deleted

Response:
Response in JSON. With the following information:

- *statusCode*: The status code of the response
- *statusText*: The status message of the response

Deleting a product set from the product catalog.

Parameters:

- *requestType*: A string representing the type of DELETE request
- *productSetId*: The id of the product set that will be deleted

Response:
Response in JSON. With the following information:

- *statusCode*: The status code of the response
- *statusText*: The status message of the response

5.8 GET /api/getProductsInSet

Getting the products of a specific set.
Parameters:

- *productSetId*: The id of the product set whose products will be listed

Response: Response in HTML. With the following information:

- *productId*: The id of the product
- *productDisplayName*: The display name of the product
- *productDesc*: The description of the product
- *productCategory*: The category of the product

5.9 GET /api/getAllProducts

Getting the all products.
Parameters:

- This function does not take any parameter.

Response: Response in HTML. With the following information:

- *productId*: The id of the product
- *productDisplayName*: The display name of the product
- *productCategory*: The category of the product
- *productSetId*: The set which the product is stored
- *productDesc*: The description of the product

5.10 GET /api/getProductsByCategory

Getting the products of a specific category in a product set.
Parameters:

- *productSetId*: The set which the product is stored
- *productCategory*: The category of the product

Response: Response in HTML. With the following information:

- *productDisplayName*: The display name of the product
- *productCategory*: The category of the product
- *productSetId*: The set which the product is stored
- *productDesc*: The description of the product

5.11 GET /api/getSets

Getting whole product sets
Parameters:

- *productSetId*: The set which the product is stored

Response: Response in HTML. With the following information:

- *productDisplayName*: The display name of the product
- *productCategory*: The category of the product
- *productSetId*: The set which the product is stored
- *productDesc*: The description of the product

5.12 POST /api/getSimilarProducts

Listing similar products by their similarity scores relatively to the image that is given by the user.
Parameters:

- *productSetId*: The product set whose products will be matched with the image
- *productCategory*: The category of the product
- *imageBase64*: The base64 text representation of the image
- *filter*: The filter for the labels of the products

Response: Response in JSON. With the following information:

- *id*: The id of the product
- *name*: The display name of the product
- *description*: The description of the product
- *category*: The category of the product
- *score*: Similarity percentage relatively to the image that is given