**Declaration**

I/We declare that this material, which I/We now submit for assessment, is entirely my/our own work and has not been taken from the work of others, except where otherwise stated. I/We have identified and included the source of all facts, ideas, opinions, and viewpoints of others in the assignment references. Direct quotations from books, journal articles, internet sources, module text, or any other source whatsoever are acknowledged and the source cited are identified in the assignment references.

I/We understand that plagiarism, collusion, and copying are grave and serious offences and accept the penalties that would be imposed should I/we engage in plagiarism, collusion or copying. I acknowledge that copying someone else’s assignment, or part of it, is wrong, and that submitting identical work to others constitutes a form of plagiarism. I/We have read and understood the colleges plagiarism policy 3AS08 (available [here](https://www.itb.ie/AboutITB/QualityAssurancePolicies/3AS08%20Institute%20Procedural%20Guidelines%20for%20Dealing%20with%20Instances%20of%20Plagiarism%20in%20Assignments%20and%20Examinations%20%2013%20June%202014.pdf)).

This material, or any part of it, has not been previously submitted for assessment for an academic purpose at this or any other academic institution.

I have not allowed anyone to copy my work with the intention of passing it off as their own work.

Name: Benas Bubulas\_\_\_\_\_\_ Dated: 08/04/2024\_\_\_\_\_

(Printing your name here will be taken as a digital signature)

WS-2024 Assignment 1

# Part 1- REST API

To create the REST-based API using Flask-Restful, the following endpoints will need to be created:

**/getProducts** – This endpoint will have a JSON list of our products that is returned to the user and is stored in a local MongoDB db.

**/getTitles** – getTitles will return a list of only the product titles. The requirement here is to make this endpoint communicate with a GraphQL server to retrieve information from out MongoDB database which contains the products.

**/insertProduct** – insertProduct should allow users to call this API endpoint using PostMan and send a product containing its id, title and cost, which will be stored on the database.

**/** - This is the root page and it should show a list of available API URLs with a brief description of how they work.

I began by opening the provided **sample\_api.py** file and added the **getProducts** class first and the imports along the way.

A screen shot of a computer program

Description automatically generated

The **GetProducts** class connects to the MongoDB database and enters the collection called sales\_data. It then takes all of the information and dumps it into “results”, which is then returned.

A screenshot of a computer

Description automatically generated

When entering [**http://127.0.0.1:5000/getProducts**](http://127.0.0.1:5000/getProducts), the Json list of products can be seen. Each product has the ProductId, P-name(product name), and cost.

Next I moved onto **/getTitles**.

A screen shot of a computer program

Description automatically generated

In order to use graphQL in the **GetTitles** class, I had to create the **TitleQuery** class first. In this class, a query is defined in order to fetch the product title from the “getProducts” page. I also needed to create the Product class which contains “title”, which is a graphene.String() type. Lastly, going back to the GetTitles class, the get request is sent which executes the graphql query which in turn sends a get request to “getProducts”. It then retrieves the product data and returns the titles of said products.

Then I created the insertProducts page. A customer API key needs to be passed in order to get access to the information. If the API key is incorrect, and error message should be returned.

A screenshot of a computer program

Description automatically generated

Here I have the custom api key set which is called “custom\_api\_key” along with error handling that will ask the user to provide a valid api key. Next is the request data parsing which parses the json data sent in the request to extract information about the product that will be inserted which consists of “ProductId”,”P-name” and “cost”. There is again an error handling response that will notify the user that there is missing required data. Finally, I’ve then added the connection to the mongodb database and inserted the product data into the collection called sales\_data.

Finally, I moved onto the root page (“/”). Here the requirement was to have a list of the API urls along with a brief description of how they work.

A black background with orange text

Description automatically generated

This part is self explanatory, the class APIDescription consists of the api descriptions and then returns jsonify (description). The jsonify method converts the description into a Json format.

Part 2- DevOps