***PRODUCT COMPARISON SERVICE: USER GUIDE***

***Steps to start the Application:***

Option 1:

There is docker-compose.yml present in root of the project ,If you wish to run it inside container, then :

* Go to the directory of docker-compose.yml
* Type command *docker-compose pull*
* Type command *docker-compose up*
* Visit the URL : <http://localhost:9000/swagger-ui.html>

For ease of use, PostgresDB will be set inside the container.

Option 2:

Build the jar using maven :

* Go to the root directory of the project.
* Type command *mvnw clean package*
* Execute jar created in the target directory with command :

*java –jar pcs-0.0.1-SNAPSHOT.jar*

* Visit the URL : <http://localhost:8080/swagger-ui.html>

**The Application provides following APIs for importing data into database:**

* */api/v1/pcs/save/products*

This API takes a file as input and imports the data from file into Service database.

Sample CSV file can be obtained from the API: */api/v1/pcs/save/products/download/csv*

Sample JSON file can be obtained from the API: */api/v1/pcs/save/products/download/json*

**NOTE: Currently only JSON/CSV file types are supported.**

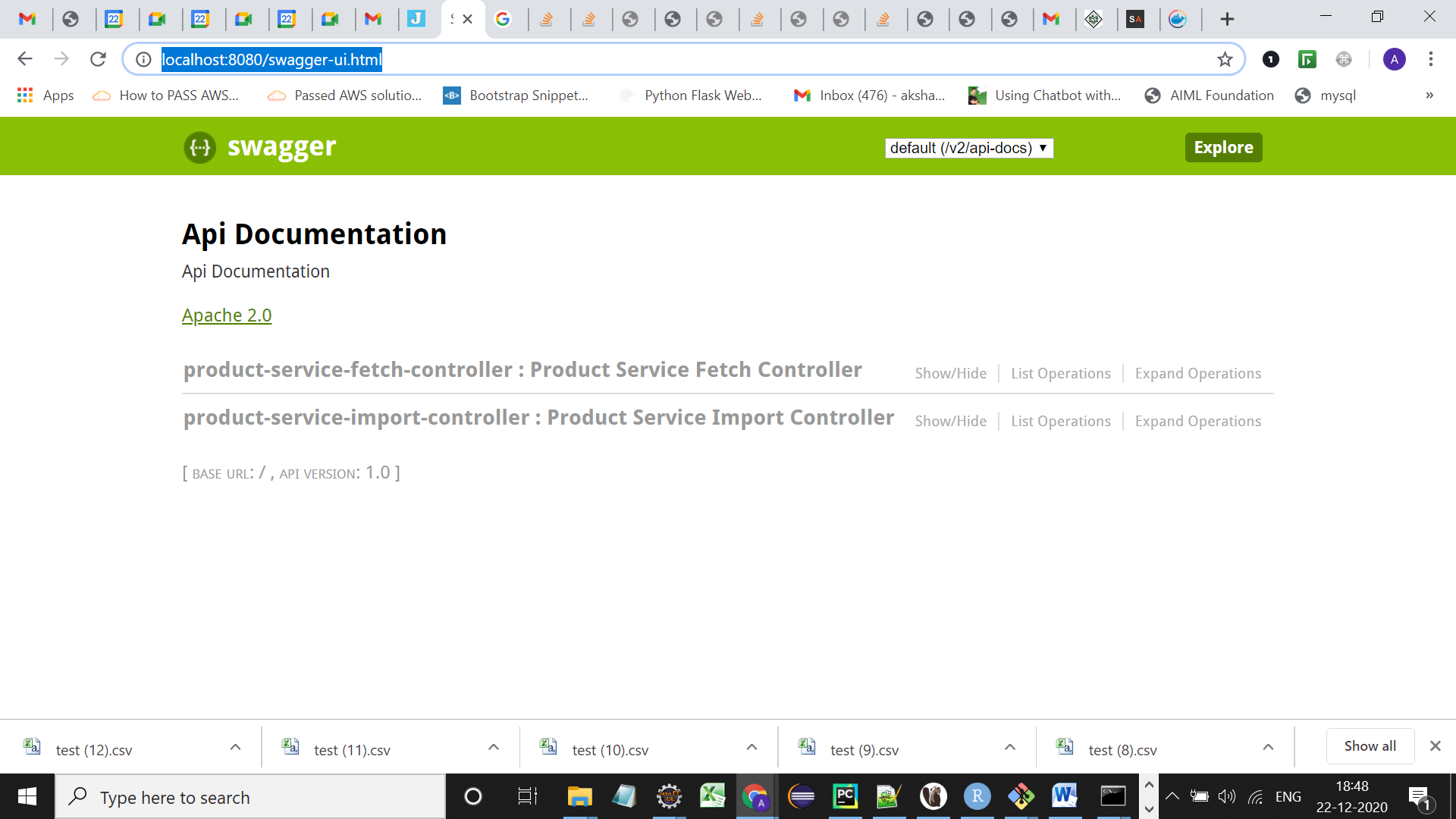
For example:

Once the server is up:

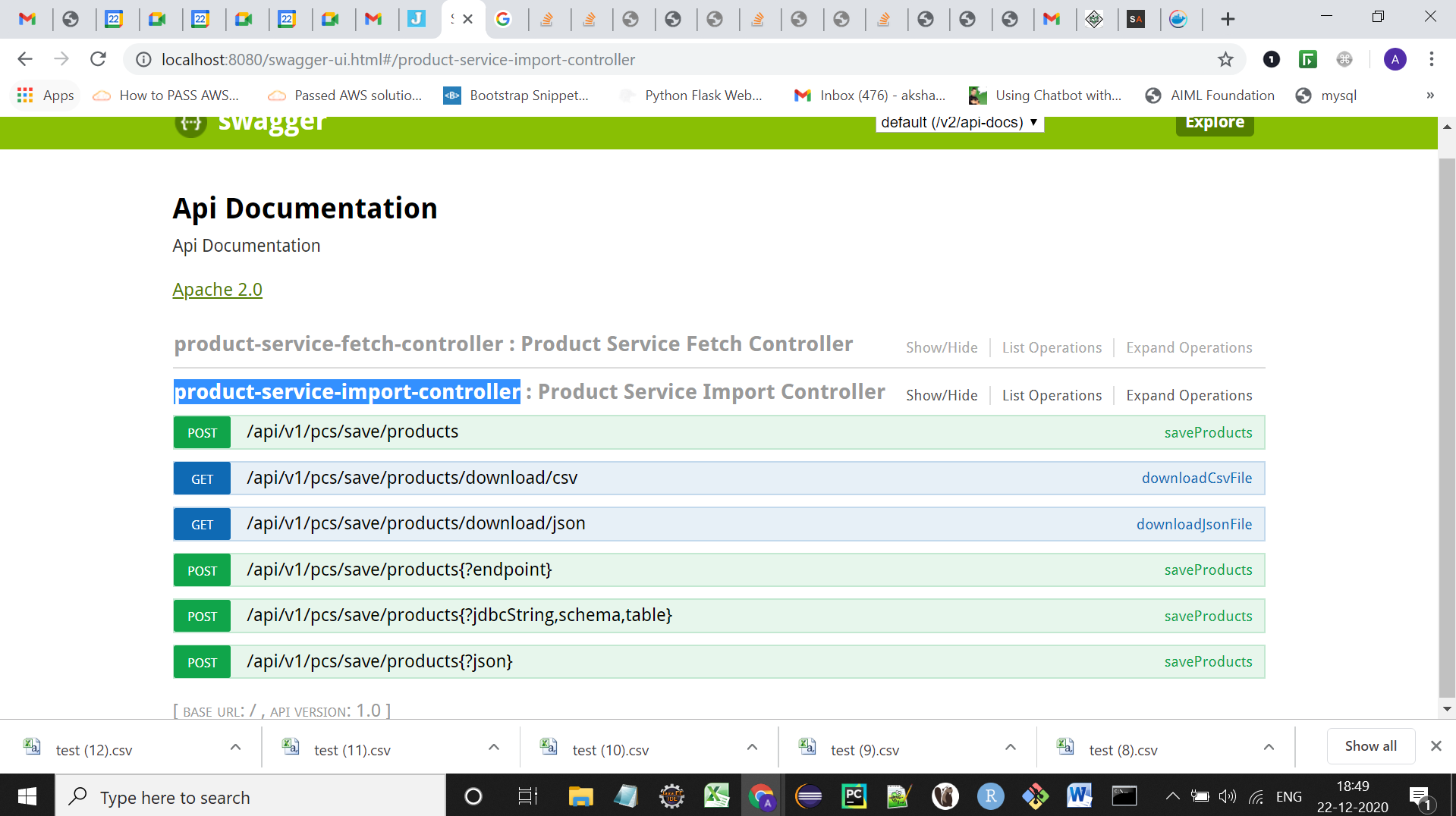
1. Visit the URL:

http://localhost:9000/swagger-ui.html for Option1 in the Application Setup

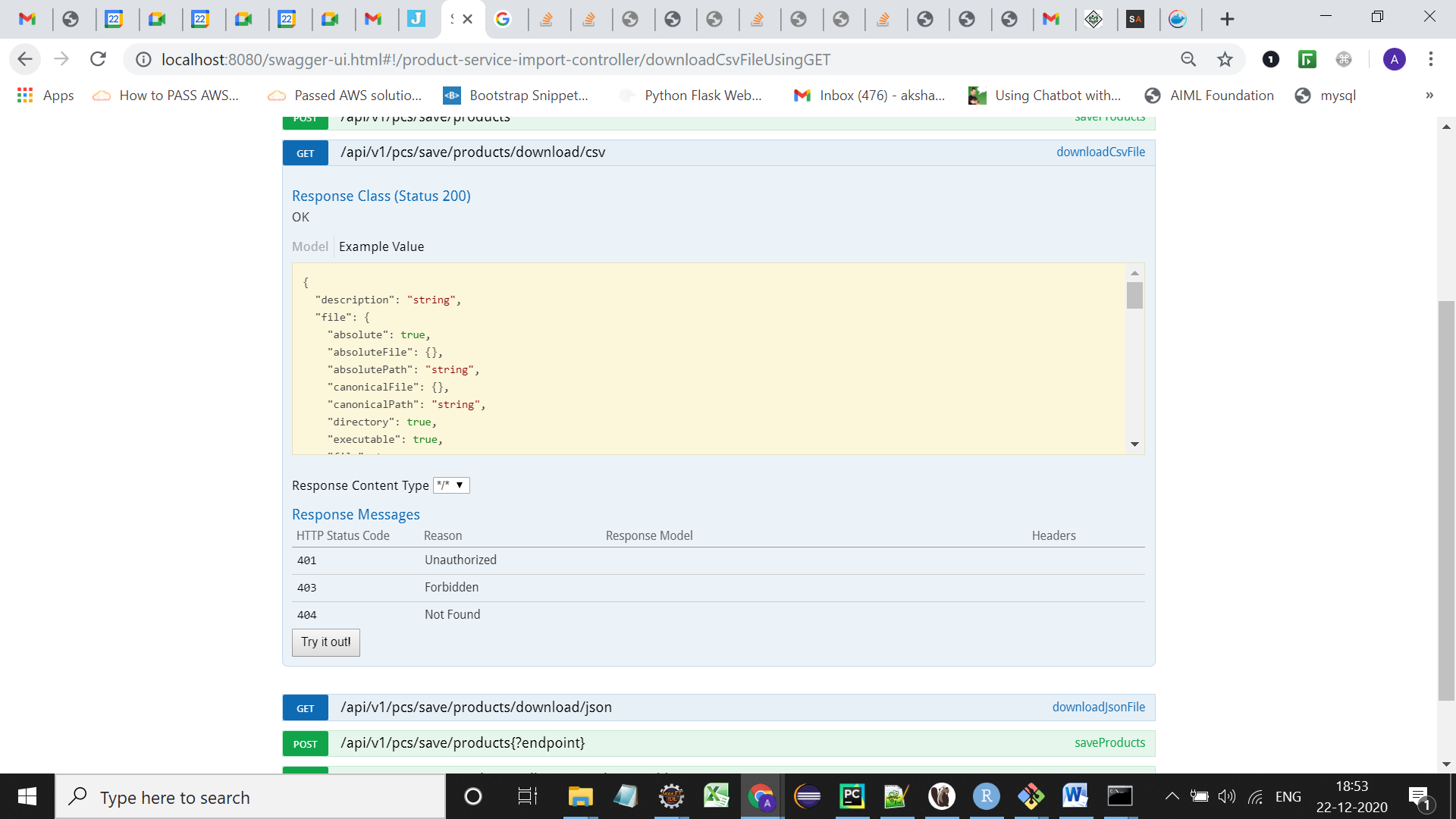
http://localhost:8080/swagger-ui.html for Option2 in the Application Setup



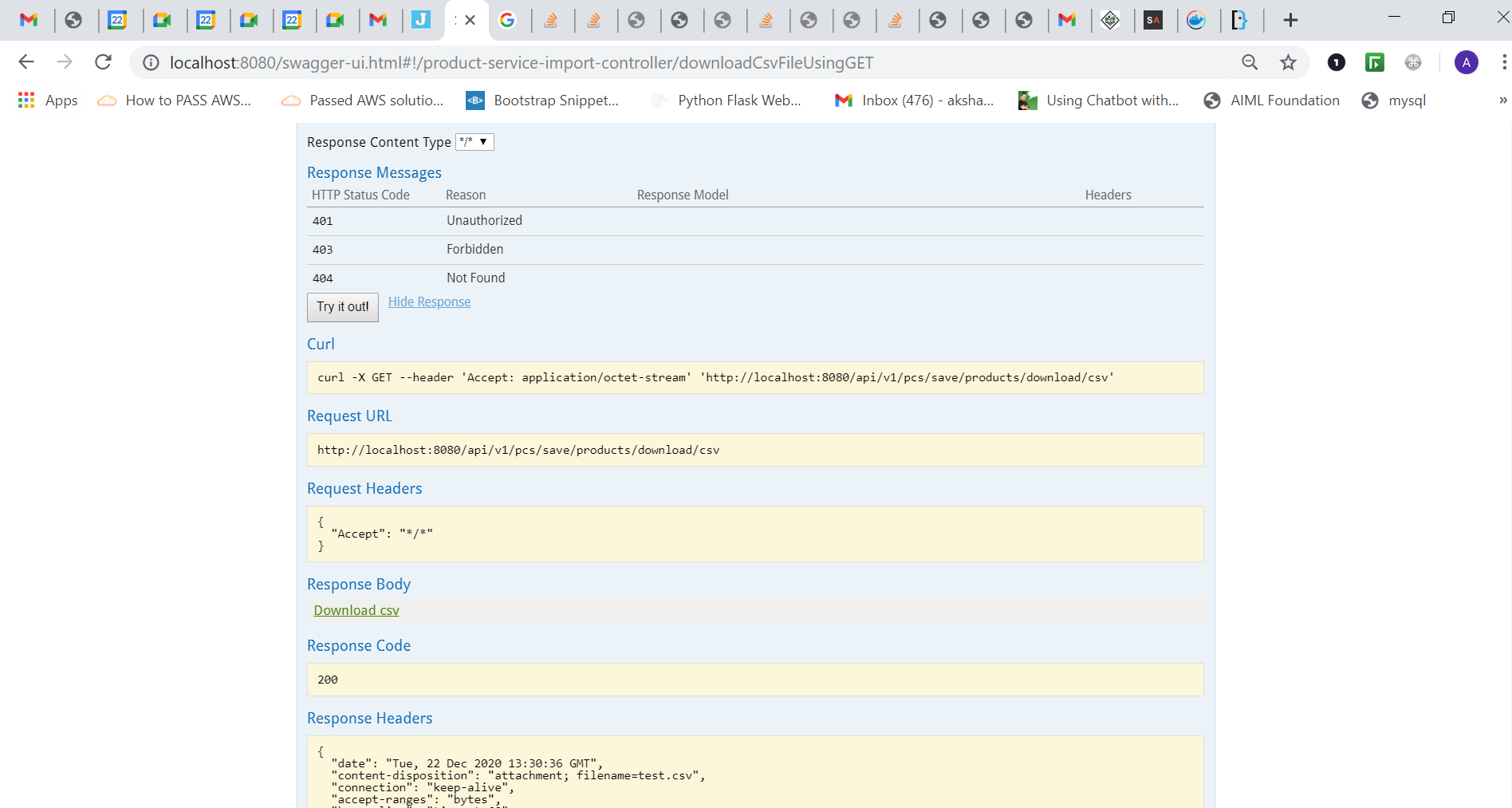
## 2. Click on the [product-service-import-controller](http://localhost:8080/swagger-ui.html#!/product-service-import-controller), you will see a list of APIs for importing data



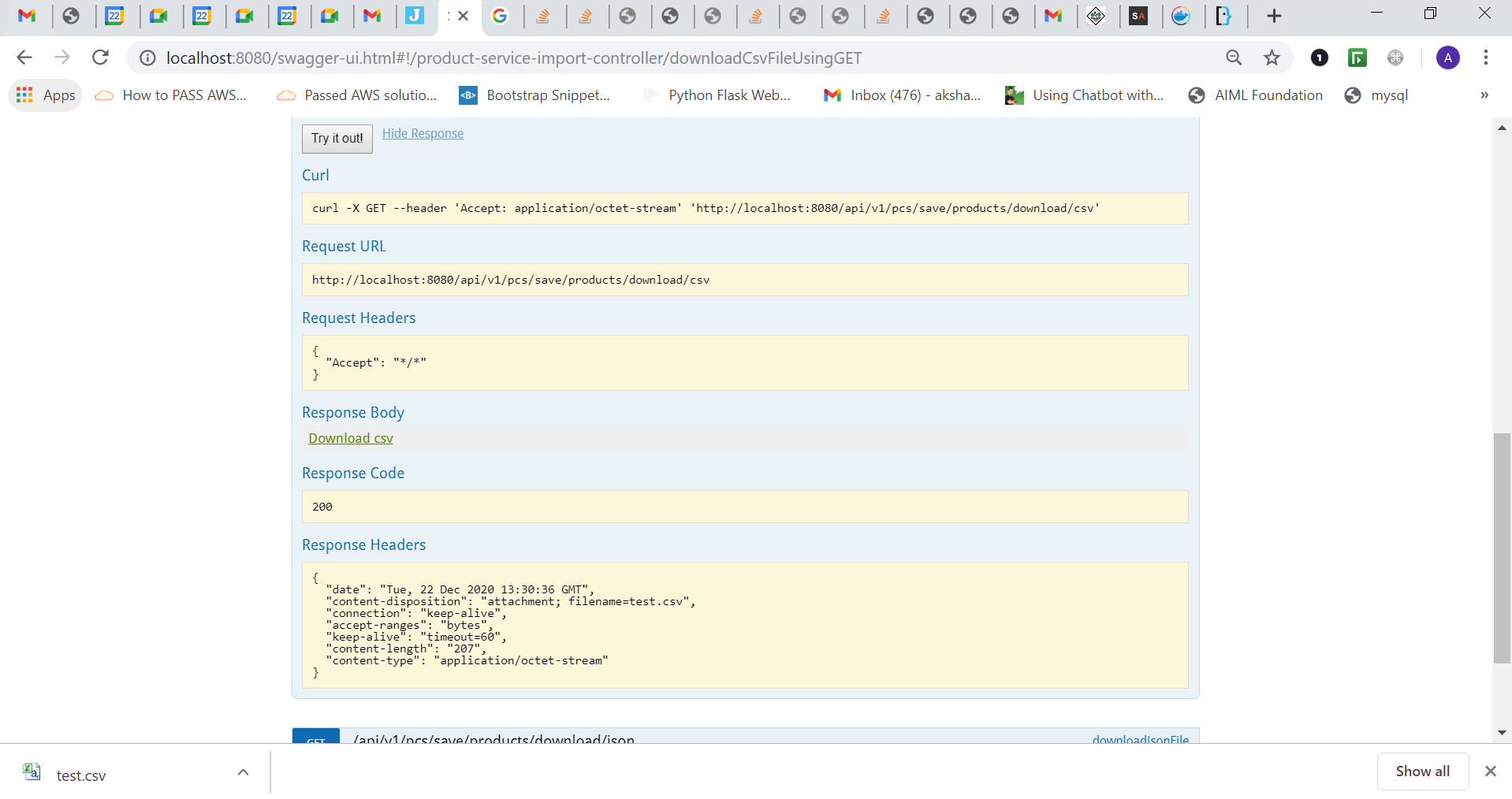
3. Click on */api/v1/pcs/save/products/download/csv*, you will see below screen shot:



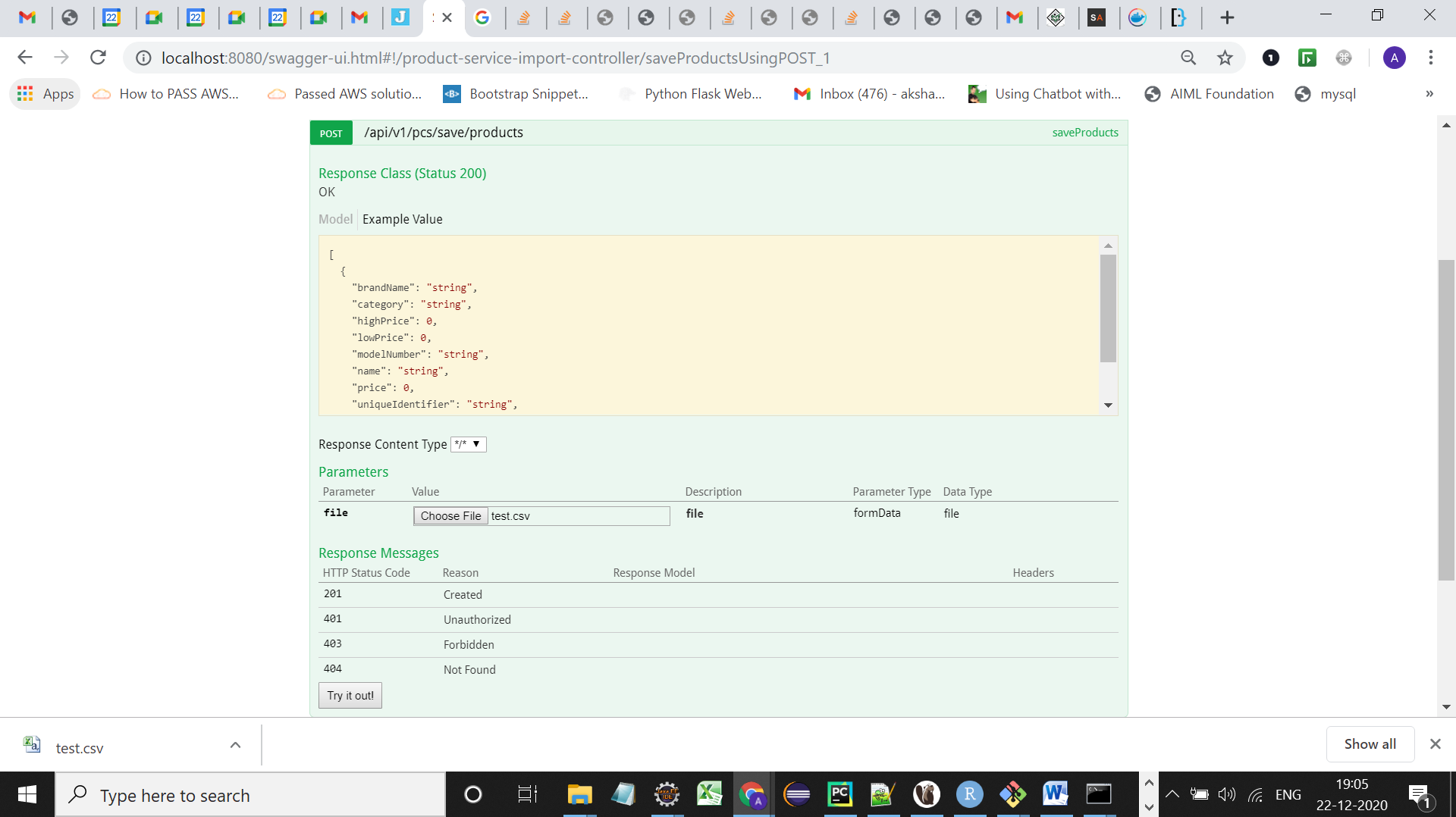
4. Click on *Try it out!*

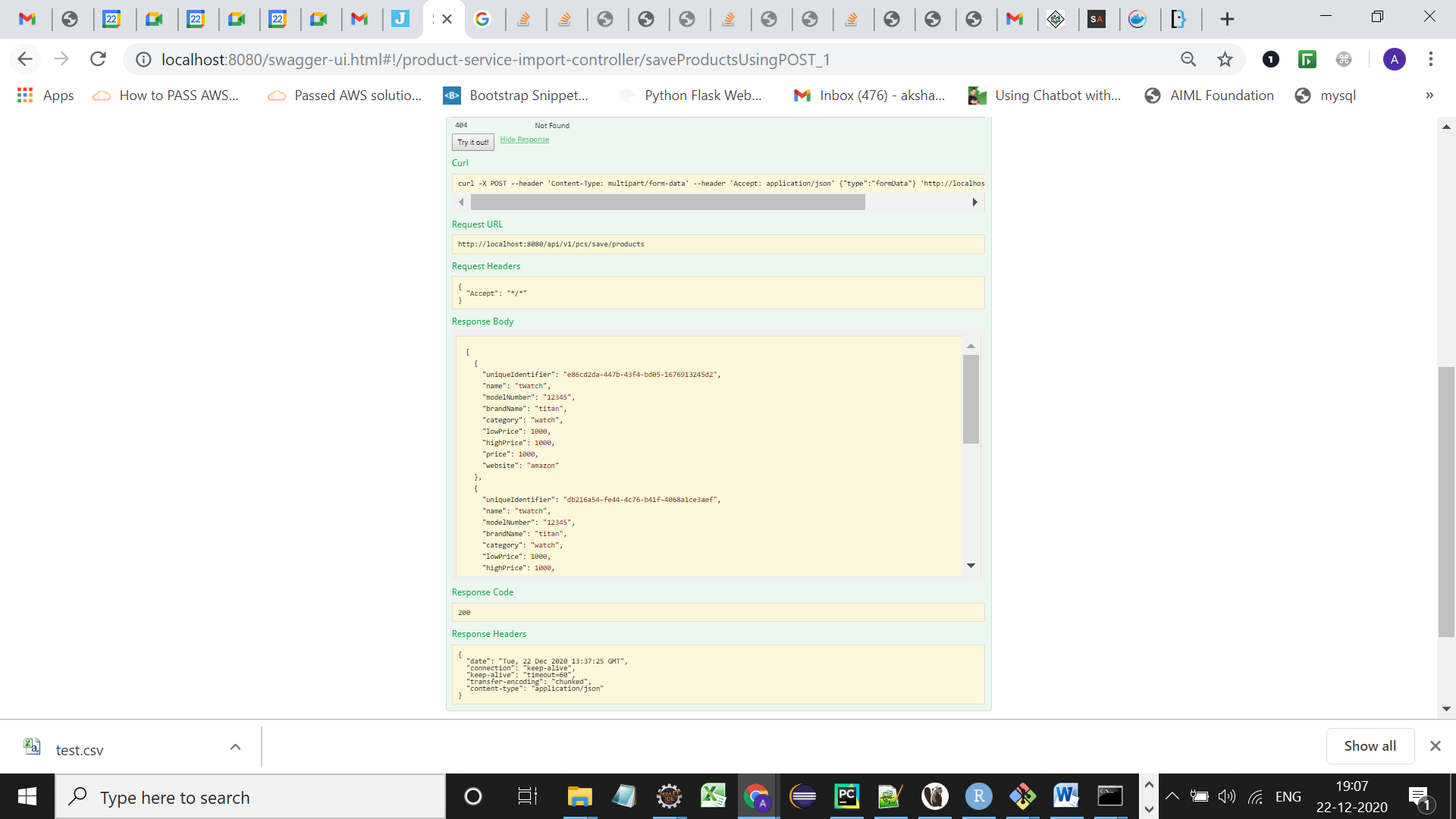
*5. You will see a download option:*

6. Click on it and a test.csv file will be downloaded.



7. Now go to the API */api/v1/pcs/save/products* visible at the top in 2nd screenshot and choose the recently downloaded file and click *Try it out!*



8. You will see the below screen shot:  


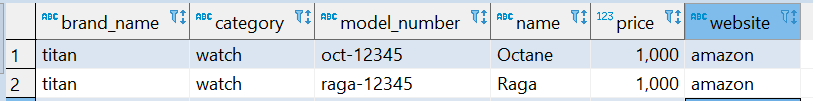
Which indicates that data in the csv file has been inserted into the database, You can also check your database and fire a simple select query to check the records.

**# Follow the same steps for JSON File as well (Only for downloading sample JSON file ,use */api/v1/pcs/save/products/download/json* instead of */api/v1/pcs/save/products/download/csv*).**

* /api/v1/pcs/save/products?jdbcString={jdbcString}&schema={schema}&table={table}

This API takes JDBC connection String along with schema name and table name to fetch data from the provided connection into our service database.

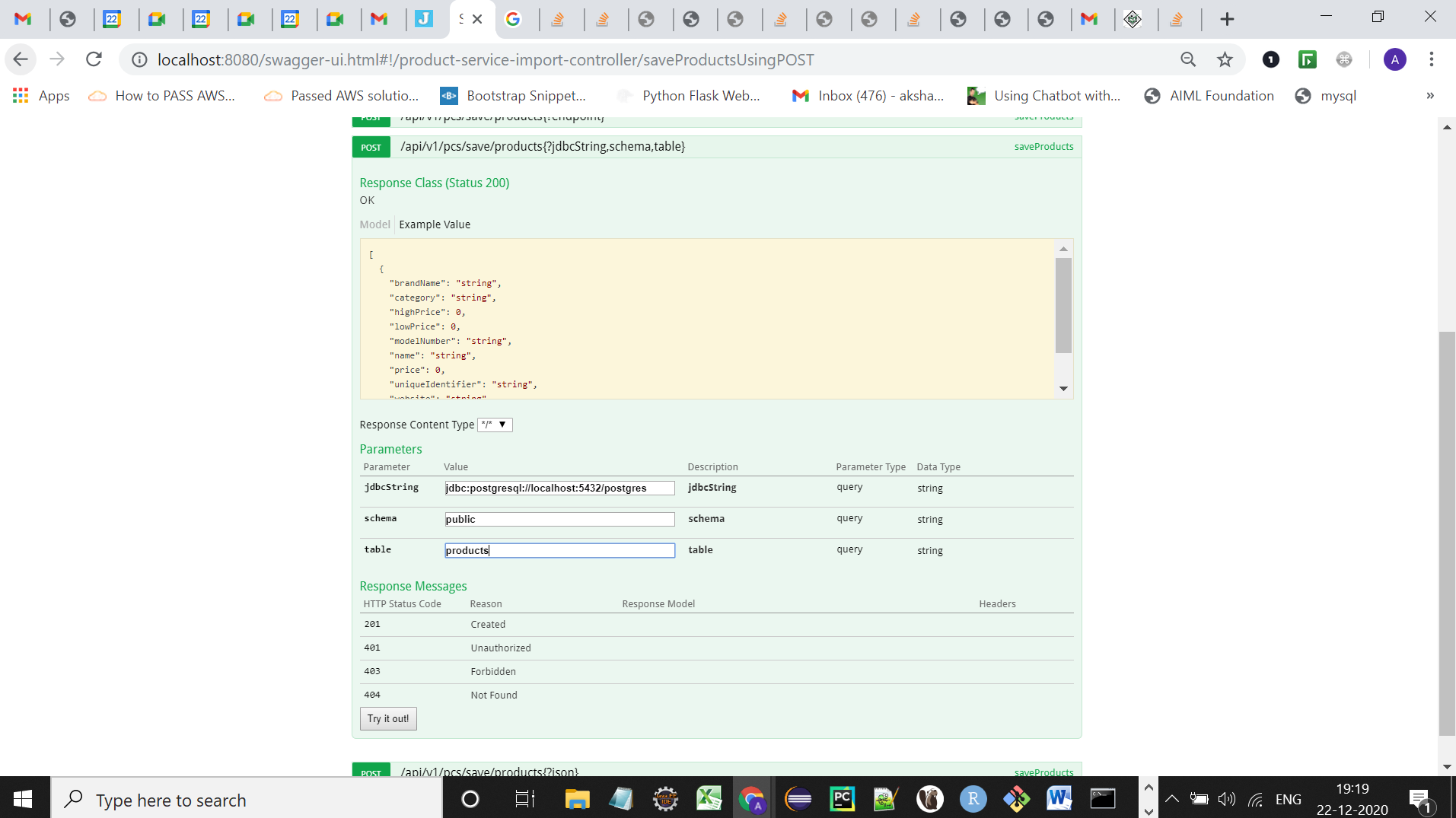
The Table schema should have following columns:



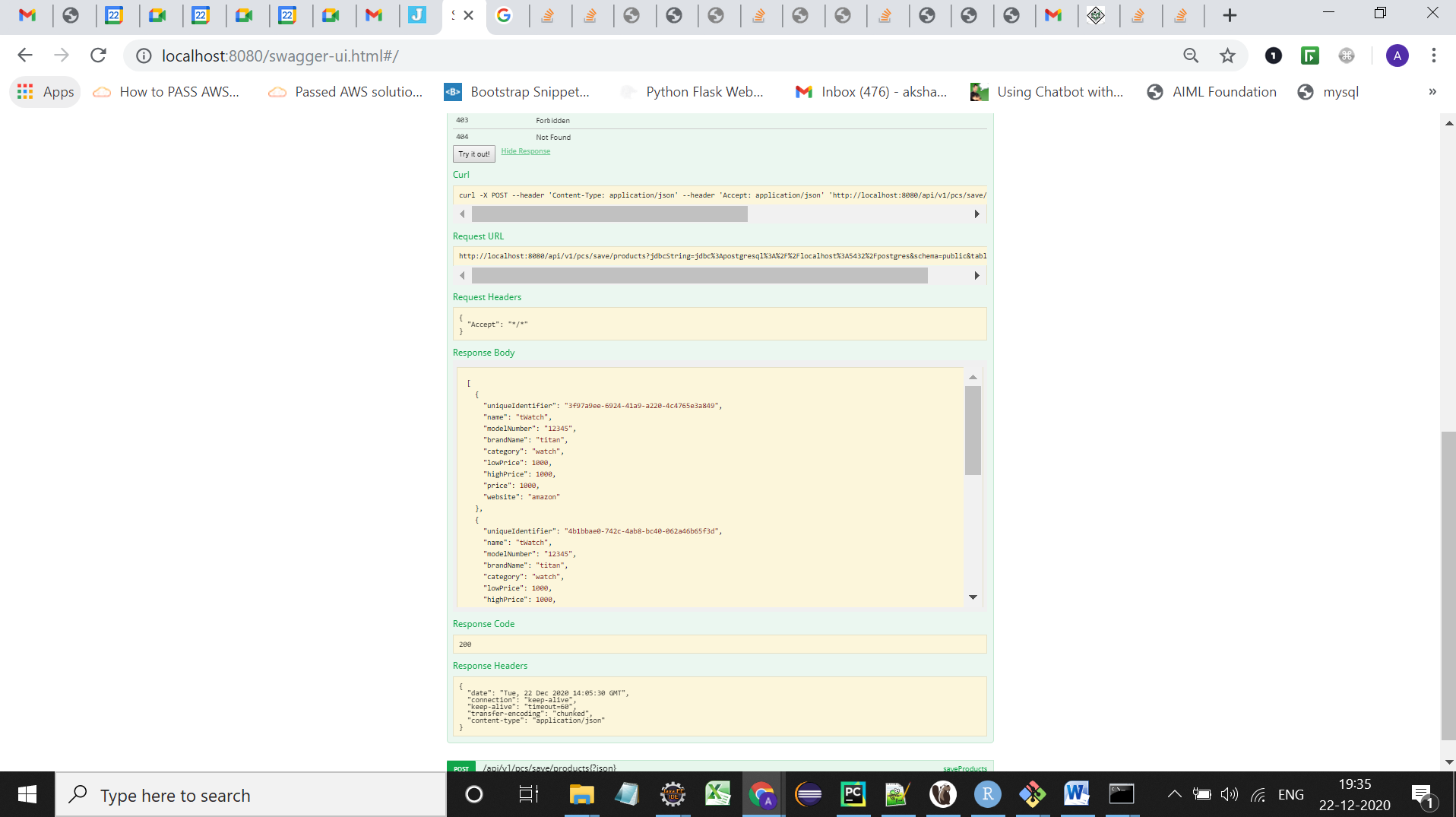
Currently only Postgres DB is supported.

For example:

1. Visit the /api/v1/pcs/save/products?jdbcString={jdbcString}&schema={schema}&table={table} API as shown in below screenshot and enter the required parameters .



1. The API will fetch the data from given JDBC connection into the current database as shown in below screenshot.



* /api/v1/pcs/save/products?endpoint={endpoint}

This API takes input from a rest endpoint and dumps the result into the application database.

Rest Endpoint Output must be in form of :

[

{

"brandName": "string",

"category": "string",

"highPrice": 0,

"lowPrice": 0,

"modelNumber": "string",

"name": "string",

"price": 0,

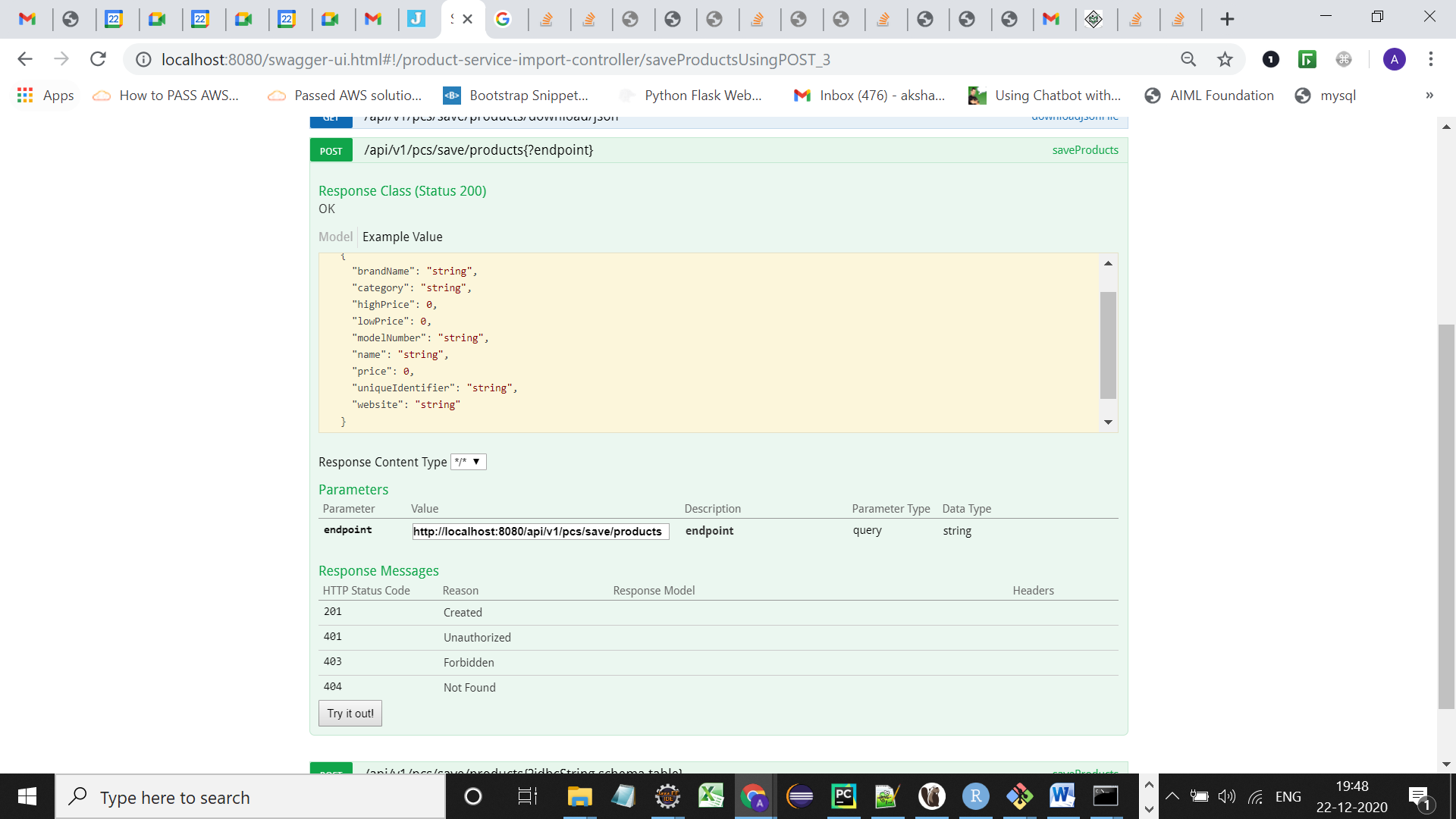
"website": "string"

}

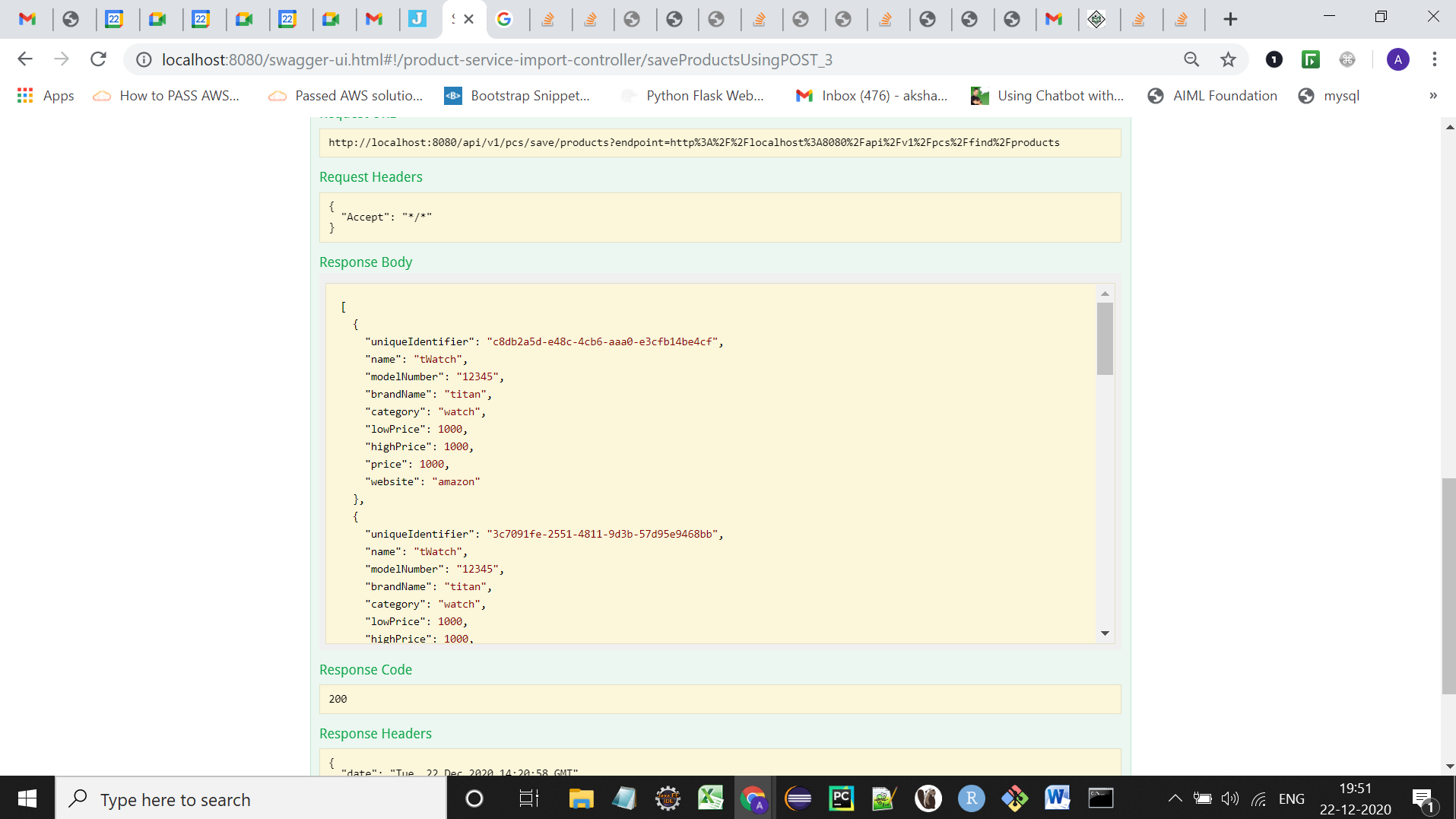
]

For example:

1. Visit the API /api/v1/pcs/save/products?endpoint={endpoint} and enter the endpoint name as shown in below screenshot



1. Click on *Try it out!*
2. *You will see the below screenshot indicating data has been inserted.*



* /api/v1/pcs/save/products?json={json}

This API consumes input in the form of Json and loads the mentioned json into the application database.

Input must be in form of:

[

{

"brandName": "string",

"category": "string",

"highPrice": 0,

"lowPrice": 0,

"modelNumber": "string",

"name": "string",

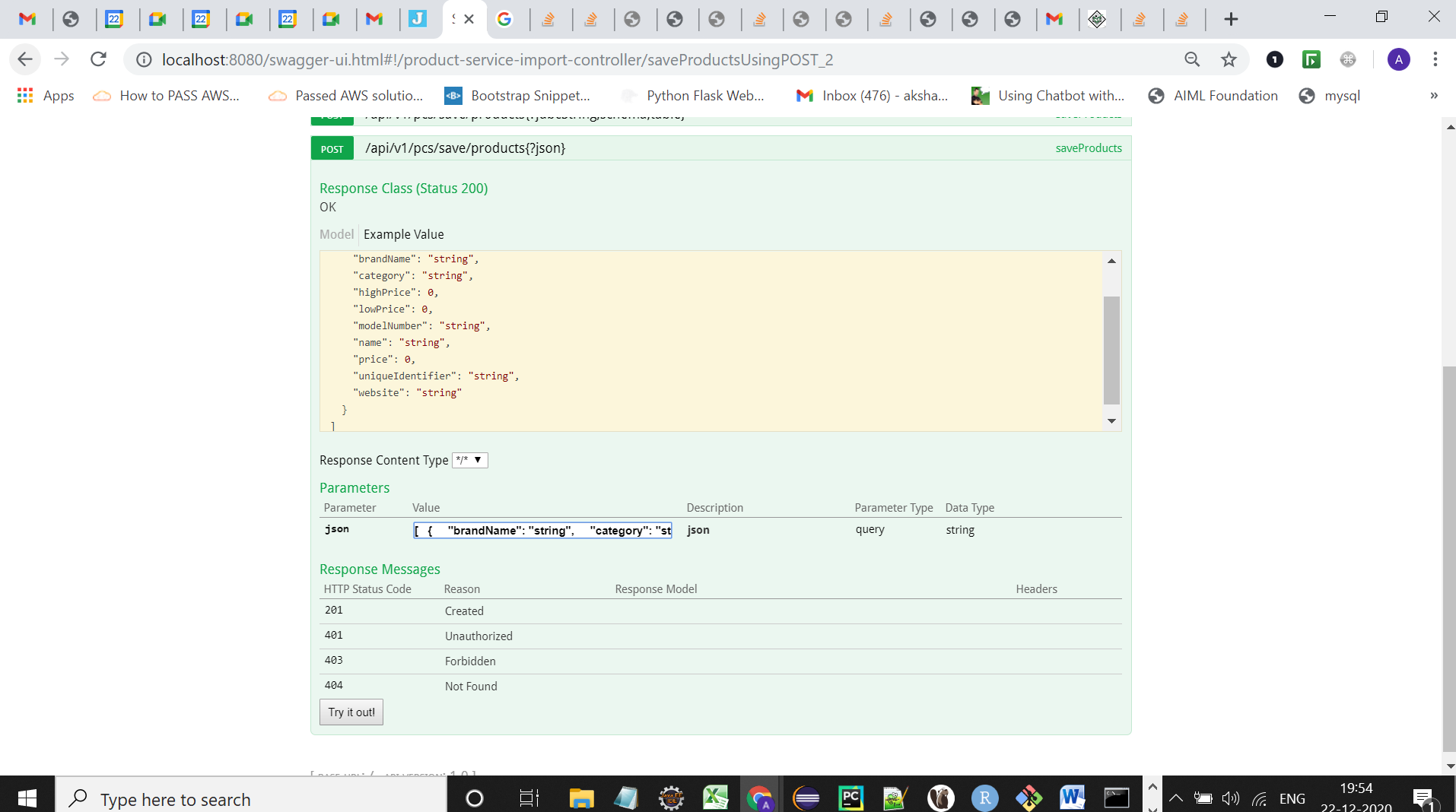
"price": 0,

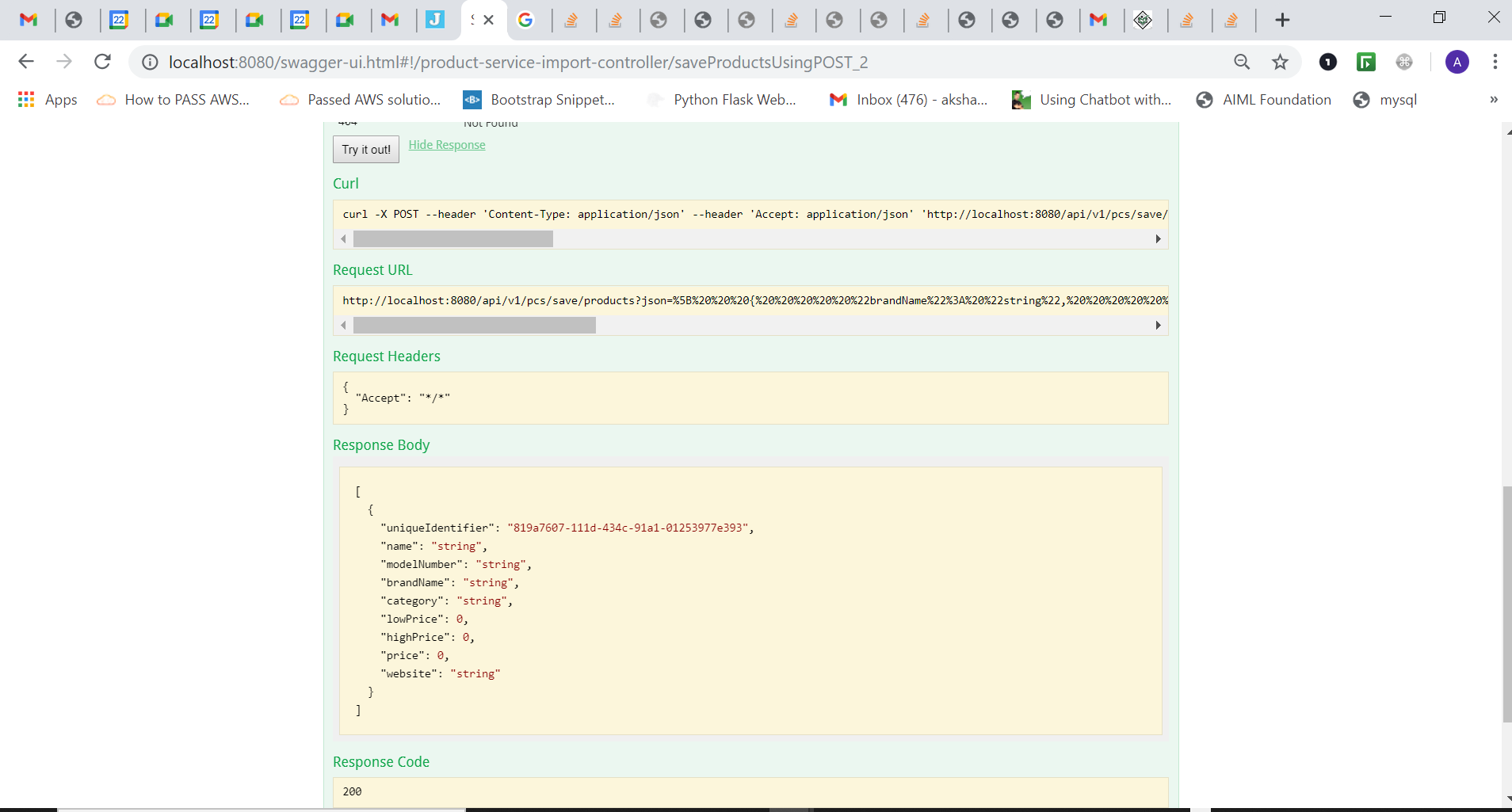
"website": "string"

}

]

For example:

1. Visit the API */api/v1/pcs/save/products?json={json}* and copy the above json into the blank json field and click *Try it out! As shown in below screenshot:*

2. You will see the data is inserted into the Database, as shown in below screenshot:  


**The Application provides following APIs for fetching data from database:**

* /api/v1/pcs/find/products?name={name}&category={category}&brand={brand}&website={website}&low={low}&high={high}&modelNumber={modelNumber}

This API takes parameters of:

name(Name of Product),

category(Category which product belongs to),

brand(Brand name of product),

website(Which website this product is available on),

low(Lower price of products),

high(Higher price of products),

modelNumber(Model number of product)

The request might consist of any combination of these parameters and all the products which matches all the given parameters are fetched from database and presented to user in the format of:

[{

"brandName": "string",

"category": "string",

"highPrice": 0,

"lowPrice": 0,

"modelNumber": "string",

"name": "string",

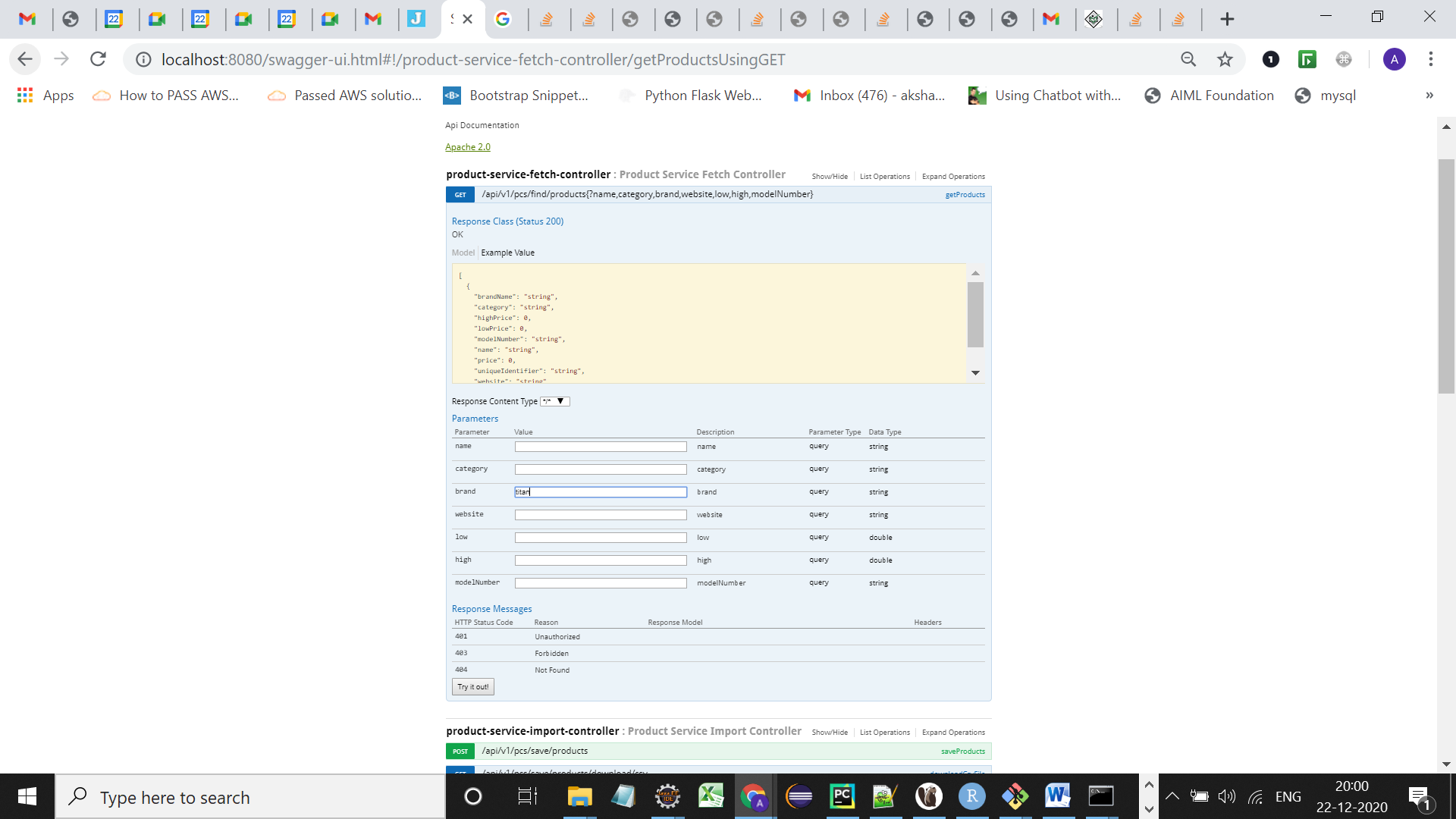
"price": 0,

"uniqueIdentifier": "string",

"website": "string"

}]

In case if none of the parameters are supplied then by default all the products are fetched.

For example:  
1. Visit the API */api/v1/pcs/find/products* and populate *brand* field and click *Try it out!* as shown in below screenshot:  
  


1. You will find all the products with brand Titan are fetched from Database.

